EMNLP 2013

2013 Conference on Empirical Methods in Natural Language Processing

Proceedings of the Conference

18-21 October 2013
Grand Hyatt Seattle
Seattle, Washington, USA

We would like to thank our sponsors:

Platinum:

Gold:

Bronze:

Supporter:

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-937284-97-8

Preface

Welcome to the 2013 Conference on Empirical Methods in Natural Language Processing.

EMNLP has grown to be one of the largest and most competitive conferences in computational linguistics. Organized by ACL SIGDAT (the Association for Computational Linguistics special interest group for linguistic data and corpus-based approaches to natural language processing), it features papers on all areas of interest to the SIGDAT community and aligned fields. It is being held this year as a standalone conference at the Grand Hyatt Seattle, in the heart of downtown Seattle, USA over the period October 18–21, 2013.

This year we introduced short papers to EMNLP for the first time, in an attempt to encourage submission of papers reporting smaller, more focused contributions and work in progress. We also put a lot of time and energy into "closing the loop" in the author response phase, in getting reviewers to explicitly acknowledge author responses and update their reviews where appropriate. We sincerely hope that this contributed towards further improvement in the quality of reviews and the decision-making process.

We received a record number of 772 valid submissions (not including co-submitted papers that were withdrawn from the conference), made up of 539 long papers and 233 short papers. These papers were reviewed across a total of 15 areas, of which Machine Translation (98 submissions), Semantics (87 submissions) and NLP-related Machine Learning (76) were the largest. The submissions were managed by 30 area chairs (two per area) and evaluated by a combined programme committee of 505 reviewers.

28% of the long paper submissions and 24% of the short paper submissions were accepted for publication at the conference. Five long papers were shortlisted for the best paper award, based on input from the reviewers and area chairs, and have been scheduled for presentation in a plenary session at the end of the conference, culminating in the presentation of the best paper award.

We would like to acknowledge all the hard work of the submitting authors, without whom there would, of course, be no conference. To the authors of accepted papers, we offer congratulations; to the authors of rejected papers, we offer our sincere commiserations, and dearly hope that the hard work of the programme committee provided you with valuable feedback on your research. We are eternally indebted to our dedicated and hard-working area chairs, and to the reviewers for their attention to detail and engagement with the author response/discussion phase, which was tremendously helpful in gauging the relative merits of each paper and being able to send out the notifications on time.

We are very grateful to our two invited speakers: Fernando Pereira (Research Director at Google) who will draw on his considerable experience and wisdom in presenting "Meaning in the Wild", focusing on machine understanding; and Andrew Ng (Co-CEO and Co-founder of Coursera) who will discuss the challenges and opportunities associated with the delivery of Massively Open Online Courses (MOOCs) in a talk titled "The Online Revolution: Education for Everyone", from the perspective of a true world leader in MOOC provision and Machine Learning/NLP research.

We would also like to thank the inimitable Priscilla Rasmussen who single-handedly looked after the local organisation of EMNLP 2013. We also wish to acknowledge the considerable efforts of Steven Berthard who put this volume together with peerless efficiency, Francesco Figari who took excellent

care of the conference website, and Rich Gerber from Softconf.com, who responded to any questions regarding START — the submission management system used for EMNLP 2013 — instantaneously and uncomplainingly, and helped us manage the large number of submissions smoothly. Additionally, we would like to thank Eugene Charniak, Mark Johnson and Noah Smith for serving on the best paper award committee, and providing characteristically probing and insightful critiques of the best paper award nominees.

Special thanks go to David Yarowsky, the general chair for the conference, who has provided us with much valuable advice, encouragement and assistance over the past six months. We would also like to thank the members of the SIGDAT board who advised us on various matters, and our predecessors James Henderson and Marius Pasca for nudging us in the right direction on a number of occasions.

On behalf of all attendees at the conference, we would also like to acknowledge the generosity of our sponsors/supports: Amazon, Google, the Allen Institute for Artificial Intelligence, Inome, IBM Research, Microsoft Research, Nuance and John Hopkins University.

It has been an honour to serve as Programme Chairs of EMNLP 2013. We sincerely hope that you — in equal measure — enjoy and are intellectually-stimulated by the conference, and have a pleasant stay in beautiful Seattle.

Timothy Baldwin and Anna Korhonen
EMNLP 2013 Programme Chairs

Organizers

General Chair:

David Yarowsky (John Hopkins University, USA)

Programme Chairs:

Timothy Baldwin (University of Melbourne, Australia)
Anna Korhonen (University of Cambridge, UK)

Workshops Chair:

Karen Livescu (Toyota Technological Institute at Chicago, USA)

Publication Chair:

Steven Bethard (University of Alabama at Birmingham, USA)

Area Chairs:

Phonology, Morphology, Tagging, Chunking and Segmentation:

Kemal Oflazer (Carnegie Mellon University Qatar)
Anna Feldman (Montclair State University, USA)

Syntax and Parsing:

Jennifer Foster (Dublin City University, Ireland)
Yoav Goldberg (Bar Ilan University, Israel)

Semantics:

Mark Stevenson (University of Sheffield, UK)
Luke Zettlemoyer (University of Washington, USA)

Discourse, Dialogue, and Pragmatics:

Carolyn Rose (Carnegie Mellon University, USA)
Matt Purver (Queen Mary, University of London, UK)

Language resources:

Emily M. Bender (University of Washington, USA)
Aline Villavicencio (Federal University of Rio Grande do Sul, Brazil)

Summarization and Generation:

Dragomir Radev (University of Michigan, USA)
Yang Liu (University of Texas at Dallas, USA)

NLP-related Machine Learning: theory, methods and algorithms:

Amir Globerson (The Hebrew University of Jerusalem, Israel)
Antal van den Bosch (Radboud University Nijmegen, Netherlands)

Machine Translation:

Taro Watanabe (NICT, Japan)
Kevin Knight (Information Sciences Institute, USA)

Information Retrieval and Question Answering:

Bernardo Magnini (Fondazione Bruno Kessler, Italy)
Soumen Chakrabarti (Indian Institute of Technology, India)

Information Extraction:

Mausam (Indian Institute of Technology, India)
Heng Ji (City University of New York, USA)

Spoken Language Processing:

Haizhou Li (Institute for Infocomm Research, Singapore)
Amanda Stent (AT&T Labs, USA)

Text Mining and Natural Language Processing Applications:

Hang Li (Huawei Noah's Ark Lab, Hong Kong)
Kevin Cohen (University of Colorado School of Medicine, USA)

Sentiment Analysis and Opinion Mining:

Janyce Weibe (University of Pittsburgh, USA)
Bing Liu (University of Illinois at Chicago, USA)

NLP for the Web and Social Media:

Miles Osborne (University of Edinburgh, UK)
Chin-Yew Lin (Microsoft Research Asia, China)

Computational Models of Human Language Acquisition and Processing:

Alessandro Lenci (University of Pisa, Italy)
Afra Alishahi (Tilburg University, Netherlands)

Table of Contents

Event-Based Time Label Propagation for Automatic Dating of News Articles
Tao Ge, Baobao Chang, Sujian Li and Zhifang Sui 1

Exploiting Discourse Analysis for Article-Wide Temporal Classification
Jun-Ping Ng, Min-Yen Kan, Ziheng Lin, Wei Feng, Bin Chen, Jian Su and Chew Lim Tan 12

Combining Generative and Discriminative Model Scores for Distant Supervision
Benjamin Roth and Dietrich Klakow 24

Exploring the Utility of Joint Morphological and Syntactic Learning from Child-directed Speech
Stella Frank, Frank Keller and Sharon Goldwater 30

A Joint Learning Model of Word Segmentation, Lexical Acquisition, and Phonetic Variability
Micha Elsner, Sharon Goldwater, Naomi Feldman and Frank Wood 42

Animacy Detection with Voting Models
Joshua Moore, Christopher J.C. Burges, Erin Renshaw and Wen-tau Yih 55

A Log-Linear Model for Unsupervised Text Normalization
Yi Yang and Jacob Eisenstein 61

Paraphrasing 4 Microblog Normalization
Wang Ling, Chris Dyer, Alan W Black and Isabel Trancoso 73

Question Difficulty Estimation in Community Question Answering Services
Jing Liu, Quan Wang, Chin-Yew Lin and Hsiao-Wuen Hon 85

Measuring Ideological Proportions in Political Speeches
Yanchuan Sim, Brice D. L. Acree, Justin H. Gross and Noah A. Smith 91

Learning to Freestyle: Hip Hop Challenge-Response Induction via Transduction Rule Segmentation
Dekai Wu, Karteek Addanki, Markus Saers and Meriem Beloucif 102

Modeling Scientific Impact with Topical Influence Regression
James Foulds and Padhraic Smyth 113

Joint Parsing and Disfluency Detection in Linear Time
Mohammad Sadegh Rasooli and Joel Tetreault 124

Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks
Masashi Tsubaki, Kevin Duh, Masashi Shimbo and Yuji Matsumoto 130

Studying the Recursive Behaviour of Adjectival Modification with Compositional Distributional Semantics
Eva Maria Vecchi, Roberto Zamparelli and Marco Baroni 141

Learning Latent Word Representations for Domain Adaptation using Supervised Word Clustering
Min Xiao, Feipeng Zhao and Yuhong Guo 152

Appropriately Incorporating Statistical Significance in PMI
Om P. Damani and Shweta Ghonge 163

Growing Multi-Domain Glossaries from a Few Seeds using Probabilistic Topic Models
Stefano Faralli and Roberto Navigli 170

Joint Learning of Phonetic Units and Word Pronunciations for ASR
Chia-ying Lee, Yu Zhang and James Glass 182

MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text
Matthew Richardson, Christopher J.C. Burges and Erin Renshaw 193

Noise-Aware Character Alignment for Bootstrapping Statistical Machine Transliteration from Bilingual Corpora
Katsuhito Sudoh, Shinsuke Mori and Masaaki Nagata 204

Optimal Beam Search for Machine Translation
Alexander Rush, Yin-Wen Chang and Michael Collins 210

An Efficient Language Model Using Double-Array Structures
Makoto Yasuhara, Toru Tanaka, Jun-ya Norimatsu and Mikio Yamamoto 222

Structured Penalties for Log-Linear Language Models
Anil Kumar Nelakanti, Cedric Archambeau, Julien Mairal, Francis Bach and Guillaume Bouchard 233

Interactive Machine Translation using Hierarchical Translation Models
Jesús González-Rubio, Daniel Ortíz-Martinez, José-Miguel Benedí and Francisco Casacuberta 244

Max-Margin Synchronous Grammar Induction for Machine Translation
Xinyan Xiao and Deyi Xiong 255

Error-Driven Analysis of Challenges in Coreference Resolution
Jonathan K. Kummerfeld and Dan Klein 265

Exploiting Zero Pronouns to Improve Chinese Coreference Resolution
Fang Kong and Hwee Tou Ng 278

Joint Coreference Resolution and Named-Entity Linking with Multi-Pass Sieves
Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld and Luke Zettlemoyer 289

Interpreting Anaphoric Shell Nouns using Antecedents of Cataphoric Shell Nouns as Training Data
Varada Kolhatkar, Heike Zinsmeister and Graeme Hirst 300

Exploring Representations from Unlabeled Data with Co-training for Chinese Word Segmentation
Longkai Zhang, Houfeng Wang, Xu Sun and Mairgup Mansur 311

Efficient Higher-Order CRFs for Morphological Tagging
Thomas Mueller, Helmut Schmid and Hinrich Schütze 322

The Effects of Syntactic Features in Automatic Prediction of Morphology
Wolfgang Seeker and Jonas Kuhn 333

Adaptor Grammars for Learning Non-Concatenative Morphology
Jan A. Botha and Phil Blunsom 345

Grounding Strategic Conversation: Using Negotiation Dialogues to Predict Trades in a Win-Lose Game
Anais Cadilhac, Nicholas Asher, Farah Benamara and Alex Lascarides 357

Unsupervised Induction of Contingent Event Pairs from Film Scenes
Zhichao Hu, Elahe Rahimtoroghi, Larissa Munishkina, Reid Swanson and Marilyn A. Walker369

Latent Anaphora Resolution for Cross-Lingual Pronoun Prediction
Christian Hardmeier, Jörg Tiedemann and Joakim Nivre 380

Towards Situated Dialogue: Revisiting Referring Expression Generation
Rui Fang, Changsong Liu, Lanbo She and Joyce Y. Chai 392

Open-Domain Fine-Grained Class Extraction from Web Search Queries
Marius Pasca 403

Unsupervised Relation Extraction with General Domain Knowledge
Oier Lopez de Lacalle and Mirella Lapata 415

Efficient Collective Entity Linking with Stacking
Zhengyan He, Shujie Liu, Yang Song, Mu Li, Ming Zhou and Houfeng Wang 426

Joint Bootstrapping of Corpus Annotations and Entity Types
Hrushikesh Mohapatra, Siddhanth Jain and Soumen Chakrabarti 436

Effectiveness and Efficiency of Open Relation Extraction
Filipe Mesquita, Jordan Schmidek and Denilson Barbosa 447

Automatic Feature Engineering for Answer Selection and Extraction
Aliaksei Severyn and Alessandro Moschitti 458

Improving Web Search Ranking by Incorporating Structured Annotation of Queries
Xiao Ding, Zhicheng Dou, Bing Qin, Ting Liu and Ji-rong Wen 468

Building Specialized Bilingual Lexicons Using Large Scale Background Knowledge
Dhouha Bouamor, Adrian Popescu, Nasredine Semmar and Pierre Zweigenbaum 479

Document Summarization via Guided Sentence Compression
Chen Li, Fei Liu, Fuliang Weng and Yang Liu 490

Anchor Graph: Global Reordering Contexts for Statistical Machine Translation
Hendra Setiawan, Bowen Zhou and Bing Xiang 501

Source-Side Classifier Preordering for Machine Translation
Uri Lerner and Slav Petrov 513

Improving Pivot-Based Statistical Machine Translation Using Random Walk
Xiaoning Zhu, Zhongjun He, Hua Wu, Haifeng Wang, Conghui Zhu and Tiejun Zhao 524

Improving Alignment of System Combination by Using Multi-objective Optimization
Tian Xia, Zongcheng Ji, Shaodan Zhai, Yidong Chen, Qun Liu and Shaojun Wang 535

Flexible and Efficient Hypergraph Interactions for Joint Hierarchical and Forest-to-String Decoding
Martin Cmejrek, Haitao Mi and Bowen Zhou 545

Factored Soft Source Syntactic Constraints for Hierarchical Machine Translation
Zhongqiang Huang, Jacob Devlin and Rabih Zbib 556

Recursive Autoencoders for ITG-Based Translation
Peng Li, Yang Liu and Maosong Sun 567

Automatically Classifying Edit Categories in Wikipedia Revisions
Johannes Daxenberger and Iryna Gurevych 578

Semi-Markov Phrase-Based Monolingual Alignment
Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch and Peter Clark 590

A Constrained Latent Variable Model for Coreference Resolution
Kai-Wei Chang, Rajhans Samdani and Dan Roth 601

Centering Similarity Measures to Reduce Hubs
Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, Marco Saerens and Kenji Fukumizu 613

Unsupervised Spectral Learning of WCFG as Low-rank Matrix Completion
Raphaël Bailly, Xavier Carreras, Franco M. Luque and Ariadna Quattoni 624

Identifying Phrasal Verbs Using Many Bilingual Corpora
Karl Pichotta and John DeNero 636

Deep Learning for Chinese Word Segmentation and POS Tagging
Xiaoqing Zheng, Hanyang Chen and Tianyu Xu 647

Joint Chinese Word Segmentation and POS Tagging on Heterogeneous Annotated Corpora with Multiple Task Learning
Xipeng Qiu, Jiayi Zhao and Xuanjing Huang 658

The Topology of Semantic Knowledge
Jimmy Dubuisson, Jean-Pierre Eckmann, Christian Scheible and Hinrich Schütze 669

Unsupervised Induction of Cross-Lingual Semantic Relations
Mike Lewis and Mark Steedman 681

Two-Stage Method for Large-Scale Acquisition of Contradiction Pattern Pairs using Entailment
Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, Motoki Sano and Kiyonori Ohtake 693

Sarcasm as Contrast between a Positive Sentiment and Negative Situation
Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert and Ruihong Huang 704

Collective Personal Profile Summarization with Social Networks
Zhongqing Wang, Shoushan LI, Fang Kong and Guodong Zhou 715

Optimized Event Storyline Generation based on Mixture-Event-Aspect Model
Lifu Huang and Lian'en Huang 726

Automatically Determining a Proper Length for Multi-Document Summarization: A Bayesian Non-parametric Approach
Tengfei Ma and Hiroshi Nakagawa 736

A Discourse-Driven Content Model for Summarising Scientific Articles Evaluated in a Complex Question Answering Task
Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor and Dietrich Rebholz-Schuhmann 747

Optimal Incremental Parsing via Best-First Dynamic Programming
Kai Zhao, James Cross and Liang Huang 758

Exploiting Language Models for Visual Recognition
Dieu-Thu Le, Jasper Uijlings and Raffaella Bernardi 769

Mining Scientific Terms and their Definitions: A Study of the ACL Anthology
Yiping Jin, Min-Yen Kan, Jun-Ping Ng and Xiangnan He 780

Joint Learning and Inference for Grammatical Error Correction
Alla Rozovskaya and Dan Roth 791

With Blinkers on: Robust Prediction of Eye Movements across Readers
Franz Matthies and Anders Søgaard 803

Using Paraphrases and Lexical Semantics to Improve the Accuracy and the Robustness of Supervised Models in Situated Dialogue Systems
Claire Gardent and Lina M. Rojas Barahona 808

Cascading Collective Classification for Bridging Anaphora Recognition using a Rich Linguistic Feature Set
Yufang Hou, Katja Markert and Michael Strube 814

A Synchronous Context Free Grammar for Time Normalization
Steven Bethard 821

Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!
Laura Chiticariu, Yunyao Li and Frederick R. Reiss 827

Improving Learning and Inference in a Large Knowledge-Base using Latent Syntactic Cues
Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel and Tom Mitchell 833

What is Hidden among Translation Rules
Libin Shen and Bowen Zhou 839

Converting Continuous-Space Language Models into N-Gram Language Models for Statistical Machine Translation
Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao and Bao-Liang Lu 845

A Corpus Level MIRA Tuning Strategy for Machine Translation
Ming Tan, Tian Xia, Shaojun Wang and Bowen Zhou 851

Word Level Language Identification in Online Multilingual Communication
Dong Nguyen and A. Seza Dogruoz 857

Microblog Entity Linking by Leveraging Extra Posts
Yuhang Guo, Bing Qin, Ting Liu and Sheng Li 863

Automatic Domain Partitioning for Multi-Domain Learning
Di Wang, Chenyan Xiong and William Yang Wang 869

Decipherment with a Million Random Restarts
Taylor Berg-Kirkpatrick and Dan Klein 874

Russian Stress Prediction using Maximum Entropy Ranking
Keith Hall and Richard Sproat 879

Scaling to Large^3 Data: An Efficient and Effective Method to Compute Distributional Thesauri
Martin Riedl and Chris Biemann 884

Discriminative Improvements to Distributional Sentence Similarity
Yangfeng Ji and Jacob Eisenstein 891

Is Twitter A Better Corpus for Measuring Sentiment Similarity?
Shi Feng, Le Zhang, Binyang Li, Daling Wang, Ge Yu and Kam-Fai Wong 897

Implicit Feature Detection via a Constrained Topic Model and SVM
Wei Wang, Hua Xu and Xiaoqiu Huang 903

Online Learning for Inexact Hypergraph Search
Hao Zhang, Liang Huang, Kai Zhao and Ryan McDonald 908

Predicting the Presence of Discourse Connectives
Gary Patterson and Andrew Kehler 914

Japanese Zero Reference Resolution Considering Exophora and Author/Reader Mentions
Masatsugu Hangyo, Daisuke Kawahara and Sadao Kurohashi 924

A Dataset for Research on Short-Text Conversations
Hao Wang, Zhengdong Lu, Hang Li and Enhong Chen 935

Discourse Level Explanatory Relation Extraction from Product Reviews Using First-Order Logic
Qi Zhang, Jin Qian, Huan Chen, Jihua Kang and Xuanjing Huang 946

Building Event Threads out of Multiple News Articles
Xavier Tannier and Véronique Moriceau 958

Tree Kernel-based Negation and Speculation Scope Detection with Structured Syntactic Parse Features
Bowei Zou, Guodong Zhou and Qiaoming Zhu 968

A temporal model of text periodicities using Gaussian Processes
Daniel Preoţiuc-Pietro and Trevor Cohn 977

Automatically Detecting and Attributing Indirect Quotations
Silvia Pareti, Tim O'Keefe, Ioannis Konstas, James R. Curran and Irena Koprinska 989

Identifying Web Search Query Reformulation using Concept based Matching
Ahmed Hassan 1000

The Answer is at your Fingertips: Improving Passage Retrieval for Web Question Answering with Search Behavior Data
Mikhail Ageev, Dmitry Lagun and Eugene Agichtein 1011

Assembling the Kazakh Language Corpus
Olzhas Makhambetov, Aibek Makazhanov, Zhandos Yessenbayev, Bakhyt Matkarimov, Islam Sabyrgaliyev and Anuar Sharafudinov 1022

Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora
Ramy Eskander, Nizar Habash and Owen Rambow 1032

Joint Language and Translation Modeling with Recurrent Neural Networks
Michael Auli, Michel Galley, Chris Quirk and Geoffrey Zweig 1044

Multi-Domain Adaptation for SMT Using Multi-Task Learning
Lei Cui, Xilun Chen, Dongdong Zhang, Shujie Liu, Mu Li and Ming Zhou 1055

Translation with Source Constituency and Dependency Trees
Fandong Meng, Jun Xie, Linfeng Song, Yajuan Lü and Qun Liu 1066

Monolingual Marginal Matching for Translation Model Adaptation
Ann Irvine, Chris Quirk and Hal Daumé III 1077

Efficient Left-to-Right Hierarchical Phrase-Based Translation with Improved Reordering
Maryam Siahbani, Baskaran Sankaran and Anoop Sarkar 1089

A Systematic Exploration of Diversity in Machine Translation
Kevin Gimpel, Dhruv Batra, Chris Dyer and Gregory Shakhnarovich 1100

Max-Violation Perceptron and Forced Decoding for Scalable MT Training
Heng Yu, Liang Huang, Haitao Mi and Kai Zhao 1112

Identifying Multiple Userids of the Same Author
Tieyun Qian and Bing Liu 1124

Gender Inference of Twitter Users in Non-English Contexts
Morgane Ciot, Morgan Sonderegger and Derek Ruths 1136

A Multimodal LDA Model integrating Textual, Cognitive and Visual Modalities
Stephen Roller and Sabine Schulte im Walde 1146

Combining PCFG-LA Models with Dual Decomposition: A Case Study with Function Labels and Binarization
Joseph Le Roux, Antoine Rozenknop and Jennifer Foster 1158

Feature Noising for Log-Linear Structured Prediction
Sida Wang, Mengqiu Wang, Stefan Wager, Percy Liang and Christopher D. Manning 1170

Improvements to the Bayesian Topic N-Gram Models
Hiroshi Noji, Daichi Mochihashi and Yusuke Miyao 1180

An Empirical Study Of Semi-Supervised Chinese Word Segmentation Using Co-Training
Fan Yang and Paul Vozila 1191

Ubertagging: Joint Segmentation and Supertagging for English
Rebecca Dridan 1201

Automatic Knowledge Acquisition for Case Alternation between the Passive and Active Voices in Japanese
Ryohei Sasano, Daisuke Kawahara, Sadao Kurohashi and Manabu Okumura 1213

Exploiting Multiple Sources for Open-Domain Hypernym Discovery
Ruiji Fu, Bing Qin and Ting Liu 1224

A Semantically Enhanced Approach to Determine Textual Similarity
Eduardo Blanco and Dan Moldovan 1235

Understanding and Quantifying Creativity in Lexical Composition
Polina Kuznetsova, Jianfu Chen and Yejin Choi 1246

Sentiment Analysis: How to Derive Prior Polarities from SentiWordNet
Marco Guerini, Lorenzo Gatti and Marco Turchi 1259

Simulating Early-Termination Search for Verbose Spoken Queries
Jerome White, Douglas W. Oard, Nitendra Rajput and Marion Zalk 1270

Summarizing Complex Events: a Cross-Modal Solution of Storylines Extraction and Reconstruction
Shize Xu, Shanshan Wang and Yan Zhang 1281

Image Description using Visual Dependency Representations
Desmond Elliott and Frank Keller 1292

Semi-Supervised Feature Transformation for Dependency Parsing
Wenliang Chen, Min Zhang and Yue Zhang 1303

Leveraging Lexical Cohesion and Disruption for Topic Segmentation
Anca-Roxana Simon, Guillaume Gravier and Pascale Sébillot 1314

This Text Has the Scent of Starbucks: A Laplacian Structured Sparsity Model for Computational Branding Analytics
William Yang Wang, Edward Lin and John Kominek 1325

Mining New Business Opportunities: Identifying Trend related Products by Leveraging Commercial Intents from Microblogs
Jinpeng Wang, Wayne Xin Zhao, Haitian Wei, Hongfei Yan and Xiaoming Li 1337

Using Topic Modeling to Improve Prediction of Neuroticism and Depression in College Students
Philip Resnik, Anderson Garron and Rebecca Resnik 1348

Predicting the Resolution of Referring Expressions from User Behavior
Nikos Engonopoulos, Martin Villalba, Ivan Titov and Alexander Koller 1354

Chinese Zero Pronoun Resolution: Some Recent Advances
Chen Chen and Vincent Ng 1360

Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction
Jason Weston, Antoine Bordes, Oksana Yakhnenko and Nicolas Usunier 1366

Simple Customization of Recursive Neural Networks for Semantic Relation Classification
Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka and Takashi Chikayama 1372

Improving Statistical Machine Translation with Word Class Models
Joern Wuebker, Stephan Peitz, Felix Rietig and Hermann Ney 1377

Shift-Reduce Word Reordering for Machine Translation
Katsuhiko Hayashi, Katsuhito Sudoh, Hajime Tsukada, Jun Suzuki and Masaaki Nagata . . . 1382

Decoding with Large-Scale Neural Language Models Improves Translation
Ashish Vaswani, Yinggong Zhao, Victoria Fossum and David Chiang 1387

Bilingual Word Embeddings for Phrase-Based Machine Translation
Will Y. Zou, Richard Socher, Daniel Cer and Christopher D. Manning 1393

Application of Localized Similarity for Web Documents
Peter Reberšek and Mateja Verlic 1399

Dependency Language Models for Sentence Completion
Joseph Gubbins and Andreas Vlachos 1405

A Walk-Based Semantically Enriched Tree Kernel Over Distributed Word Representations
Shashank Srivastava, Dirk Hovy and Eduard Hovy 1411

Automatic Idiom Identification in Wiktionary
Grace Muzny and Luke Zettlemoyer 1417

Elephant: Sequence Labeling for Word and Sentence Segmentation
Kilian Evang, Valerio Basile, Grzegorz Chrupała and Johan Bos 1422

Detecting Compositionality of Multi-Word Expressions using Nearest Neighbours in Vector Space Models
Douwe Kiela and Stephen Clark 1427

Naive Bayes Word Sense Induction
Do Kook Choe and Eugene Charniak 1433

The VerbCorner Project: Toward an Empirically-Based Semantic Decomposition of Verbs
Joshua K. Hartshorne, Claire Bonial and Martha Palmer 1438

Where Not to Eat? Improving Public Policy by Predicting Hygiene Inspections Using Online Reviews
Jun Seok Kang, Polina Kuznetsova, Michael Luca and Yejin Choi 1443

Automatically Identifying Pseudepigraphic Texts
Moshe Koppel and Shachar Seidman 1449

Dynamic Feature Selection for Dependency Parsing
He He, Hal Daumé III and Jason Eisner 1455

Semi-Supervised Representation Learning for Cross-Lingual Text Classification
Min Xiao and Yuhong Guo 1465

Using Crowdsourcing to get Representations based on Regular Expressions
Anders Søgaard, Hector Martinez, Jakob Elming and Anders Johannsen 1476

Overcoming the Lack of Parallel Data in Sentence Compression
Katja Filippova and Yasemin Altun 1481

Fast Joint Compression and Summarization via Graph Cuts
Xian Qian and Yang Liu 1492

Inducing Document Plans for Concept-to-Text Generation
Ioannis Konstas and Mirella Lapata 1503

Single-Document Summarization as a Tree Knapsack Problem
Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda and Masaaki Nagata . 1515

A Hierarchical Entity-Based Approach to Structuralize User Generated Content in Social Media: A Case of Yahoo! Answers
Baichuan Li, Jing Liu, Chin-Yew Lin, Irwin King and Michael R. Lyu 1521

Semantic Parsing on Freebase from Question-Answer Pairs
Jonathan Berant, Andrew Chou, Roy Frostig and Percy Liang 1533

Scaling Semantic Parsers with On-the-Fly Ontology Matching
Tom Kwiatkowski, Eunsol Choi, Yoav Artzi and Luke Zettlemoyer 1545

Classifying Message Board Posts with an Extracted Lexicon of Patient Attributes
Ruihong Huang and Ellen Riloff 1557

Lexical Chain Based Cohesion Models for Document-Level Statistical Machine Translation
Deyi Xiong, Yang Ding, Min Zhang and Chew Lim Tan 1563

A Convex Alternative to IBM Model 2
Andrei Simion, Michael Collins and Cliff Stein 1574

Pair Language Models for Deriving Alternative Pronunciations and Spellings from Pronunciation Dictionaries
Russell Beckley and Brian Roark 1584

Prior Disambiguation of Word Tensors for Constructing Sentence Vectors
Dimitri Kartsaklis and Mehrnoosh Sadrzadeh 1590

Multi-Relational Latent Semantic Analysis
Kai-Wei Chang, Wen-tau Yih and Christopher Meek 1602

A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else)
Ivan Vulić and Marie-Francine Moens 1613

Deriving Adjectival Scales from Continuous Space Word Representations
Joo-Kyung Kim and Marie-Catherine de Marneffe 1625

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank
Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng and Christopher Potts 1631

Open Domain Targeted Sentiment
Margaret Mitchell, Jacqui Aguilar, Theresa Wilson and Benjamin Van Durme 1643

Exploiting Domain Knowledge in Aspect Extraction
Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos and Riddhiman Ghosh 1655

Dependency-Based Decipherment for Resource-Limited Machine Translation
Qing Dou and Kevin Knight 1668

Translating into Morphologically Rich Languages with Synthetic Phrases
Victor Chahuneau, Eva Schlinger, Noah A. Smith and Chris Dyer 1677

Boosting Cross-Language Retrieval by Learning Bilingual Phrase Associations from Relevance Rankings
Artem Sokokov, Laura Jehl, Felix Hieber and Stefan Riezler 1688

Recurrent Continuous Translation Models
Nal Kalchbrenner and Phil Blunsom 1700

Learning Biological Processes with Global Constraints
Aju Thalappillil Scaria, Jonathan Berant, Mengqiu Wang, Peter Clark, Justin Lewis, Brittany Harding and Christopher D. Manning 1710

Generating Coherent Event Schemas at Scale
Niranjan Balasubramanian, Stephen Soderland, Mausam and Oren Etzioni 1721

Orthonormal Explicit Topic Analysis for Cross-Lingual Document Matching
John Philip McCrae, Philipp Cimiano and Roman Klinger 1732

Automated Essay Scoring by Maximizing Human-Machine Agreement
Hongbo Chen and Ben He 1741

Success with Style: Using Writing Style to Predict the Success of Novels
Vikas Ganjigunte Ashok, Song Feng and Yejin Choi 1753

A Generative Joint, Additive, Sequential Model of Topics and Speech Acts in Patient-Doctor Communication
Byron C. Wallace, Thomas A Trikalinos, M. Barton Laws, Ira B. Wilson and Eugene Charniak 1765

Harvesting Parallel News Streams to Generate Paraphrases of Event Relations
Congle Zhang and Daniel S. Weld 1776

Relational Inference for Wikification
Xiao Cheng and Dan Roth 1787

Event Schema Induction with a Probabilistic Entity-Driven Model
Nathanael Chambers 1797

Using Soft Constraints in Joint Inference for Clinical Concept Recognition
Prateek Jindal and Dan Roth 1808

Exploring Demographic Language Variations to Improve Multilingual Sentiment Analysis in Social Media
Svitlana Volkova, Theresa Wilson and David Yarowsky 1815

Opinion Mining in Newspaper Articles by Entropy-Based Word Connections
Thomas Scholz and Stefan Conrad 1828

Collective Opinion Target Extraction in Chinese Microblogs
Xinjie Zhou, Xiaojun Wan and Jianguo Xiao 1840

Detecting Promotional Content in Wikipedia
Shruti Bhosale, Heath Vinicombe and Raymond Mooney 1851

Learning Topics and Positions from Debatepedia
Swapna Gottipati, Minghui Qiu, Yanchuan Sim, Jing Jiang and Noah A. Smith 1858

A Unified Model for Topics, Events and Users on Twitter
Qiming Diao and Jing Jiang 1869

Authorship Attribution of Micro-Messages
Roy Schwartz, Oren Tsur, Ari Rappoport and Moshe Koppel 1880

Detection of Product Comparisons - How Far Does an Out-of-the-Box Semantic Role Labeling System Take You?
Wiltrud Kessler and Jonas Kuhn 1892

A Multi-Teraflop Constituency Parser using GPUs
John Canny, David Hall and Dan Klein 1898

Fish Transporters and Miracle Homes: How Compositional Distributional Semantics can Help NP Parsing
Angeliki Lazaridou, Eva Maria Vecchi and Marco Baroni 1908

Learning Distributions over Logical Forms for Referring Expression Generation
Nicholas FitzGerald, Yoav Artzi and Luke Zettlemoyer 1914

Learning to Rank Lexical Substitutions
György Szarvas, Róbert Busa-Fekete and Eyke Hüllermeier 1926

Identifying Manipulated Offerings on Review Portals
Jiwei Li, Myle Ott and Claire Cardie 1933

Well-Argued Recommendation: Adaptive Models Based on Words in Recommender Systems
Julien Gaillard, Marc El-Beze, Eitan Altman and Emmanuel Ethis 1943

Regularized Minimum Error Rate Training
Michel Galley, Chris Quirk, Colin Cherry and Kristina Toutanova 1948

Of Words, Eyes and Brains: Correlating Image-Based Distributional Semantic Models with Neural Representations of Concepts
Andrew J. Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio and Marco Baroni 1960

Easy Victories and Uphill Battles in Coreference Resolution
Greg Durrett and Dan Klein 1971

Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction
Valentin I. Spitkovsky, Hiyan Alshawi and Daniel Jurafsky 1983

Cross-Lingual Discriminative Learning of Sequence Models with Posterior Regularization
Kuzman Ganchev and Dipanjan Das 1996

Conference Program

Saturday, October 19, 2013

(7:30-9:00) Breakfast

(9:00-9:15) Opening Remarks

(9:20-10:30) Information Extraction I

9:20–9:45 *Event-Based Time Label Propagation for Automatic Dating of News Articles*
Tao Ge, Baobao Chang, Sujian Li and Zhifang Sui

9:45–10:10 *Exploiting Discourse Analysis for Article-Wide Temporal Classification*
Jun-Ping Ng, Min-Yen Kan, Ziheng Lin, Wei Feng, Bin Chen, Jian Su and Chew Lim Tan

10:10–10:30 *Combining Generative and Discriminative Model Scores for Distant Supervision*
Benjamin Roth and Dietrich Klakow

(9:20-10:30) Language Acquisition and Processing

9:20–9:45 *Exploring the Utility of Joint Morphological and Syntactic Learning from Child-directed Speech*
Stella Frank, Frank Keller and Sharon Goldwater

9:45–10:10 *A Joint Learning Model of Word Segmentation, Lexical Acquisition, and Phonetic Variability*
Micha Elsner, Sharon Goldwater, Naomi Feldman and Frank Wood

10:10–10:30 *Animacy Detection with Voting Models*
Joshua Moore, Christopher J.C. Burges, Erin Renshaw and Wen-tau Yih

Saturday, October 19, 2013 (continued)

(9:20-10:30) NLP for Social Media I

9:20–9:45 *A Log-Linear Model for Unsupervised Text Normalization*
Yi Yang and Jacob Eisenstein

9:45–10:10 *Paraphrasing 4 Microblog Normalization*
Wang Ling, Chris Dyer, Alan W Black and Isabel Trancoso

10:10–10:30 *Question Difficulty Estimation in Community Question Answering Services*
Jing Liu, Quan Wang, Chin-Yew Lin and Hsiao-Wuen Hon

(10:30-11:00) Break

(11:00-12:35) NLP Applications I

11:00–11:25 *Measuring Ideological Proportions in Political Speeches*
Yanchuan Sim, Brice D. L. Acree, Justin H. Gross and Noah A. Smith

11:25–11:50 *Learning to Freestyle: Hip Hop Challenge-Response Induction via Transduction Rule Segmentation*
Dekai Wu, Karteek Addanki, Markus Saers and Meriem Beloucif

11:50–12:15 *Modeling Scientific Impact with Topical Influence Regression*
James Foulds and Padhraic Smyth

12:15–12:35 *Joint Parsing and Disfluency Detection in Linear Time*
Mohammad Sadegh Rasooli and Joel Tetreault

Saturday, October 19, 2013 (continued)

(11:00-12:35) Semantics I

11:00–11:25 *Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks*
Masashi Tsubaki, Kevin Duh, Masashi Shimbo and Yuji Matsumoto

11:25–11:50 *Studying the Recursive Behaviour of Adjectival Modification with Compositional Distributional Semantics*
Eva Maria Vecchi, Roberto Zamparelli and Marco Baroni

11:50–12:15 *Learning Latent Word Representations for Domain Adaptation using Supervised Word Clustering*
Min Xiao, Feipeng Zhao and Yuhong Guo

12:15–12:35 *Appropriately Incorporating Statistical Significance in PMI*
Om P. Damani and Shweta Ghonge

(11:00-12:35) Language Resources

11:00–11:25 *Growing Multi-Domain Glossaries from a Few Seeds using Probabilistic Topic Models*
Stefano Faralli and Roberto Navigli

11:25–11:50 *Joint Learning of Phonetic Units and Word Pronunciations for ASR*
Chia-ying Lee, Yu Zhang and James Glass

11:50–12:15 *MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text*
Matthew Richardson, Christopher J.C. Burges and Erin Renshaw

12:15–12:35 *Noise-Aware Character Alignment for Bootstrapping Statistical Machine Transliteration from Bilingual Corpora*
Katsuhito Sudoh, Shinsuke Mori and Masaaki Nagata

(12:35–2:00) Lunch

(2:00–3:15) First invited talk: Andrew Ng

(3:15–3:45) Break

Saturday, October 19, 2013 (continued)

(3:45-5:50) Machine Translation I

3:45–4:10 *Optimal Beam Search for Machine Translation*
Alexander Rush, Yin-Wen Chang and Michael Collins

4:10–4:35 *An Efficient Language Model Using Double-Array Structures*
Makoto Yasuhara, Toru Tanaka, Jun-ya Norimatsu and Mikio Yamamoto

4:35–5:00 *Structured Penalties for Log-Linear Language Models*
Anil Kumar Nelakanti, Cedric Archambeau, Julien Mairal, Francis Bach and Guillaume Bouchard

5:00–5:25 *Interactive Machine Translation using Hierarchical Translation Models*
Jesús González-Rubio, Daniel Ortíz-Martinez, José-Miguel Benedí and Francisco Casacuberta

5:25–5:50 *Max-Margin Synchronous Grammar Induction for Machine Translation*
Xinyan Xiao and Deyi Xiong

(3:45-5:50) Dialogue and Discourse

3:45–4:10 *Error-Driven Analysis of Challenges in Coreference Resolution*
Jonathan K. Kummerfeld and Dan Klein

4:10–4:35 *Exploiting Zero Pronouns to Improve Chinese Coreference Resolution*
Fang Kong and Hwee Tou Ng

4:35–5:00 *Joint Coreference Resolution and Named-Entity Linking with Multi-Pass Sieves*
Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld and Luke Zettlemoyer

5:00–5:25 *Interpreting Anaphoric Shell Nouns using Antecedents of Cataphoric Shell Nouns as Training Data*
Varada Kolhatkar, Heike Zinsmeister and Graeme Hirst

Saturday, October 19, 2013 (continued)

(3:45-5:50) Morphology and Phonology

3:45–4:10 *Exploring Representations from Unlabeled Data with Co-training for Chinese Word Segmentation*
Longkai Zhang, Houfeng Wang, Xu Sun and Mairgup Mansur

4:10–4:35 *Efficient Higher-Order CRFs for Morphological Tagging*
Thomas Mueller, Helmut Schmid and Hinrich Schütze

4:35–5:00 *The Effects of Syntactic Features in Automatic Prediction of Morphology*
Wolfgang Seeker and Jonas Kuhn

5:00–5:25 *Adaptor Grammars for Learning Non-Concatenative Morphology*
Jan A. Botha and Phil Blunsom

(6:00-7:30) Poster Session A

Grounding Strategic Conversation: Using Negotiation Dialogues to Predict Trades in a Win-Lose Game
Anais Cadilhac, Nicholas Asher, Farah Benamara and Alex Lascarides

Unsupervised Induction of Contingent Event Pairs from Film Scenes
Zhichao Hu, Elahe Rahimtoroghi, Larissa Munishkina, Reid Swanson and Marilyn A. Walker

Latent Anaphora Resolution for Cross-Lingual Pronoun Prediction
Christian Hardmeier, Jörg Tiedemann and Joakim Nivre

Towards Situated Dialogue: Revisiting Referring Expression Generation
Rui Fang, Changsong Liu, Lanbo She and Joyce Y. Chai

Open-Domain Fine-Grained Class Extraction from Web Search Queries
Marius Pasca

Unsupervised Relation Extraction with General Domain Knowledge
Oier Lopez de Lacalle and Mirella Lapata

Efficient Collective Entity Linking with Stacking
Zhengyan He, Shujie Liu, Yang Song, Mu Li, Ming Zhou and Houfeng Wang

Saturday, October 19, 2013 (continued)

Joint Bootstrapping of Corpus Annotations and Entity Types
Hrushikesh Mohapatra, Siddhanth Jain and Soumen Chakrabarti

Effectiveness and Efficiency of Open Relation Extraction
Filipe Mesquita, Jordan Schmidek and Denilson Barbosa

Automatic Feature Engineering for Answer Selection and Extraction
Aliaksei Severyn and Alessandro Moschitti

Improving Web Search Ranking by Incorporating Structured Annotation of Queries
Xiao Ding, Zhicheng Dou, Bing Qin, Ting Liu and Ji-rong Wen

Building Specialized Bilingual Lexicons Using Large Scale Background Knowledge
Dhouha Bouamor, Adrian Popescu, Nasredine Semmar and Pierre Zweigenbaum

Document Summarization via Guided Sentence Compression
Chen Li, Fei Liu, Fuliang Weng and Yang Liu

Anchor Graph: Global Reordering Contexts for Statistical Machine Translation
Hendra Setiawan, Bowen Zhou and Bing Xiang

Source-Side Classifier Preordering for Machine Translation
Uri Lerner and Slav Petrov

Improving Pivot-Based Statistical Machine Translation Using Random Walk
Xiaoning Zhu, Zhongjun He, Hua Wu, Haifeng Wang, Conghui Zhu and Tiejun Zhao

Improving Alignment of System Combination by Using Multi-objective Optimization
Tian Xia, Zongcheng Ji, Shaodan Zhai, Yidong Chen, Qun Liu and Shaojun Wang

Flexible and Efficient Hypergraph Interactions for Joint Hierarchical and Forest-to-String Decoding
Martin Cmejrek, Haitao Mi and Bowen Zhou

Factored Soft Source Syntactic Constraints for Hierarchical Machine Translation
Zhongqiang Huang, Jacob Devlin and Rabih Zbib

Recursive Autoencoders for ITG-Based Translation
Peng Li, Yang Liu and Maosong Sun

Automatically Classifying Edit Categories in Wikipedia Revisions
Johannes Daxenberger and Iryna Gurevych

Semi-Markov Phrase-Based Monolingual Alignment
Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch and Peter Clark

A Constrained Latent Variable Model for Coreference Resolution
Kai-Wei Chang, Rajhans Samdani and Dan Roth

Centering Similarity Measures to Reduce Hubs
Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, Marco Saerens and Kenji Fukumizu

Unsupervised Spectral Learning of WCFG as Low-rank Matrix Completion
Raphaël Bailly, Xavier Carreras, Franco M. Luque and Ariadna Quattoni

Identifying Phrasal Verbs Using Many Bilingual Corpora
Karl Pichotta and John DeNero

Deep Learning for Chinese Word Segmentation and POS Tagging
Xiaoqing Zheng, Hanyang Chen and Tianyu Xu

Joint Chinese Word Segmentation and POS Tagging on Heterogeneous Annotated Corpora with Multiple Task Learning
Xipeng Qiu, Jiayi Zhao and Xuanjing Huang

The Topology of Semantic Knowledge
Jimmy Dubuisson, Jean-Pierre Eckmann, Christian Scheible and Hinrich Schütze

Unsupervised Induction of Cross-Lingual Semantic Relations
Mike Lewis and Mark Steedman

Two-Stage Method for Large-Scale Acquisition of Contradiction Pattern Pairs using Entailment
Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, Motoki Sano and Kiyonori Ohtake

Saturday, October 19, 2013 (continued)

Sarcasm as Contrast between a Positive Sentiment and Negative Situation
Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert and Ruihong Huang

Collective Personal Profile Summarization with Social Networks
Zhongqing Wang, Shoushan LI, Fang Kong and Guodong Zhou

Optimized Event Storyline Generation based on Mixture-Event-Aspect Model
Lifu Huang and Lian'en Huang

Automatically Determining a Proper Length for Multi-Document Summarization: A Bayesian Nonparametric Approach
Tengfei Ma and Hiroshi Nakagawa

A Discourse-Driven Content Model for Summarising Scientific Articles Evaluated in a Complex Question Answering Task
Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor and Dietrich Rebholz-Schuhmann

Optimal Incremental Parsing via Best-First Dynamic Programming
Kai Zhao, James Cross and Liang Huang

Exploiting Language Models for Visual Recognition
Dieu-Thu Le, Jasper Uijlings and Raffaella Bernardi

Mining Scientific Terms and their Definitions: A Study of the ACL Anthology
Yiping Jin, Min-Yen Kan, Jun-Ping Ng and Xiangnan He

Joint Learning and Inference for Grammatical Error Correction
Alla Rozovskaya and Dan Roth

With Blinkers on: Robust Prediction of Eye Movements across Readers
Franz Matthies and Anders Søgaard

Using Paraphrases and Lexical Semantics to Improve the Accuracy and the Robustness of Supervised Models in Situated Dialogue Systems
Claire Gardent and Lina M. Rojas Barahona

Cascading Collective Classification for Bridging Anaphora Recognition using a Rich Linguistic Feature Set
Yufang Hou, Katja Markert and Michael Strube

A Synchronous Context Free Grammar for Time Normalization
Steven Bethard

Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!
Laura Chiticariu, Yunyao Li and Frederick R. Reiss

Improving Learning and Inference in a Large Knowledge-Base using Latent Syntactic Cues
Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel and Tom Mitchell

What is Hidden among Translation Rules
Libin Shen and Bowen Zhou

Converting Continuous-Space Language Models into N-Gram Language Models for Statistical Machine Translation
Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao and Bao-Liang Lu

A Corpus Level MIRA Tuning Strategy for Machine Translation
Ming Tan, Tian Xia, Shaojun Wang and Bowen Zhou

Word Level Language Identification in Online Multilingual Communication
Dong Nguyen and A. Seza Dogruoz

Microblog Entity Linking by Leveraging Extra Posts
Yuhang Guo, Bing Qin, Ting Liu and Sheng Li

Automatic Domain Partitioning for Multi-Domain Learning
Di Wang, Chenyan Xiong and William Yang Wang

Decipherment with a Million Random Restarts
Taylor Berg-Kirkpatrick and Dan Klein

Russian Stress Prediction using Maximum Entropy Ranking
Keith Hall and Richard Sproat

Scaling to Large^3 Data: An Efficient and Effective Method to Compute Distributional Thesauri
Martin Riedl and Chris Biemann

Discriminative Improvements to Distributional Sentence Similarity
Yangfeng Ji and Jacob Eisenstein

Is Twitter A Better Corpus for Measuring Sentiment Similarity?
Shi Feng, Le Zhang, Binyang Li, Daling Wang, Ge Yu and Kam-Fai Wong

Implicit Feature Detection via a Constrained Topic Model and SVM
Wei Wang, Hua Xu and Xiaoqiu Huang

Online Learning for Inexact Hypergraph Search
Hao Zhang, Liang Huang, Kai Zhao and Ryan McDonald

(7:30-9:00) Poster Session B

Predicting the Presence of Discourse Connectives
Gary Patterson and Andrew Kehler

Japanese Zero Reference Resolution Considering Exophora and Author/Reader Mentions
Masatsugu Hangyo, Daisuke Kawahara and Sadao Kurohashi

A Dataset for Research on Short-Text Conversations
Hao Wang, Zhengdong Lu, Hang Li and Enhong Chen

Discourse Level Explanatory Relation Extraction from Product Reviews Using First-Order Logic
Qi Zhang, Jin Qian, Huan Chen, Jihua Kang and Xuanjing Huang

Building Event Threads out of Multiple News Articles
Xavier Tannier and Véronique Moriceau

Tree Kernel-based Negation and Speculation Scope Detection with Structured Syntactic Parse Features
Bowei Zou, Guodong Zhou and Qiaoming Zhu

A temporal model of text periodicities using Gaussian Processes
Daniel Preoţiuc-Pietro and Trevor Cohn

Automatically Detecting and Attributing Indirect Quotations
Silvia Pareti, Tim O'Keefe, Ioannis Konstas, James R. Curran and Irena Koprinska

Saturday, October 19, 2013 (continued)

Identifying Web Search Query Reformulation using Concept based Matching
Ahmed Hassan

The Answer is at your Fingertips: Improving Passage Retrieval for Web Question Answering with Search Behavior Data
Mikhail Ageev, Dmitry Lagun and Eugene Agichtein

Assembling the Kazakh Language Corpus
Olzhas Makhambetov, Aibek Makazhanov, Zhandos Yessenbayev, Bakhyt Matkarimov, Islam Sabyrgaliyev and Anuar Sharafudinov

Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora
Ramy Eskander, Nizar Habash and Owen Rambow

Joint Language and Translation Modeling with Recurrent Neural Networks
Michael Auli, Michel Galley, Chris Quirk and Geoffrey Zweig

Multi-Domain Adaptation for SMT Using Multi-Task Learning
Lei Cui, Xilun Chen, Dongdong Zhang, Shujie Liu, Mu Li and Ming Zhou

Translation with Source Constituency and Dependency Trees
Fandong Meng, Jun Xie, Linfeng Song, Yajuan Lü and Qun Liu

Monolingual Marginal Matching for Translation Model Adaptation
Ann Irvine, Chris Quirk and Hal Daumé III

Efficient Left-to-Right Hierarchical Phrase-Based Translation with Improved Reordering
Maryam Siahbani, Baskaran Sankaran and Anoop Sarkar

A Systematic Exploration of Diversity in Machine Translation
Kevin Gimpel, Dhruv Batra, Chris Dyer and Gregory Shakhnarovich

Max-Violation Perceptron and Forced Decoding for Scalable MT Training
Heng Yu, Liang Huang, Haitao Mi and Kai Zhao

Identifying Multiple Userids of the Same Author
Tieyun Qian and Bing Liu

Saturday, October 19, 2013 (continued)

Gender Inference of Twitter Users in Non-English Contexts
Morgane Ciot, Morgan Sonderegger and Derek Ruths

A Multimodal LDA Model integrating Textual, Cognitive and Visual Modalities
Stephen Roller and Sabine Schulte im Walde

Combining PCFG-LA Models with Dual Decomposition: A Case Study with Function Labels and Binarization
Joseph Le Roux, Antoine Rozenknop and Jennifer Foster

Feature Noising for Log-Linear Structured Prediction
Sida Wang, Mengqiu Wang, Stefan Wager, Percy Liang and Christopher D. Manning

Improvements to the Bayesian Topic N-Gram Models
Hiroshi Noji, Daichi Mochihashi and Yusuke Miyao

An Empirical Study Of Semi-Supervised Chinese Word Segmentation Using Co-Training
Fan Yang and Paul Vozila

Ubertagging: Joint Segmentation and Supertagging for English
Rebecca Dridan

Automatic Knowledge Acquisition for Case Alternation between the Passive and Active Voices in Japanese
Ryohei Sasano, Daisuke Kawahara, Sadao Kurohashi and Manabu Okumura

Exploiting Multiple Sources for Open-Domain Hypernym Discovery
Ruiji Fu, Bing Qin and Ting Liu

A Semantically Enhanced Approach to Determine Textual Similarity
Eduardo Blanco and Dan Moldovan

Understanding and Quantifying Creativity in Lexical Composition
Polina Kuznetsova, Jianfu Chen and Yejin Choi

Sentiment Analysis: How to Derive Prior Polarities from SentiWordNet
Marco Guerini, Lorenzo Gatti and Marco Turchi

Simulating Early-Termination Search for Verbose Spoken Queries
Jerome White, Douglas W. Oard, Nitendra Rajput and Marion Zalk

Summarizing Complex Events: a Cross-Modal Solution of Storylines Extraction and Reconstruction
Shize Xu, Shanshan Wang and Yan Zhang

Image Description using Visual Dependency Representations
Desmond Elliott and Frank Keller

Semi-Supervised Feature Transformation for Dependency Parsing
Wenliang Chen, Min Zhang and Yue Zhang

Leveraging Lexical Cohesion and Disruption for Topic Segmentation
Anca-Roxana Simon, Guillaume Gravier and Pascale Sébillot

This Text Has the Scent of Starbucks: A Laplacian Structured Sparsity Model for Computational Branding Analytics
William Yang Wang, Edward Lin and John Kominek

Mining New Business Opportunities: Identifying Trend related Products by Leveraging Commercial Intents from Microblogs
Jinpeng Wang, Wayne Xin Zhao, Haitian Wei, Hongfei Yan and Xiaoming Li

Using Topic Modeling to Improve Prediction of Neuroticism and Depression in College Students
Philip Resnik, Anderson Garron and Rebecca Resnik

Predicting the Resolution of Referring Expressions from User Behavior
Nikos Engonopoulos, Martin Villalba, Ivan Titov and Alexander Koller

Chinese Zero Pronoun Resolution: Some Recent Advances
Chen Chen and Vincent Ng

Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction
Jason Weston, Antoine Bordes, Oksana Yakhnenko and Nicolas Usunier

Simple Customization of Recursive Neural Networks for Semantic Relation Classification
Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka and Takashi Chikayama

Saturday, October 19, 2013 (continued)

Improving Statistical Machine Translation with Word Class Models
Joern Wuebker, Stephan Peitz, Felix Rietig and Hermann Ney

Shift-Reduce Word Reordering for Machine Translation
Katsuhiko Hayashi, Katsuhito Sudoh, Hajime Tsukada, Jun Suzuki and Masaaki Nagata

Decoding with Large-Scale Neural Language Models Improves Translation
Ashish Vaswani, Yinggong Zhao, Victoria Fossum and David Chiang

Bilingual Word Embeddings for Phrase-Based Machine Translation
Will Y. Zou, Richard Socher, Daniel Cer and Christopher D. Manning

Application of Localized Similarity for Web Documents
Peter Reberšek and Mateja Verlic

Dependency Language Models for Sentence Completion
Joseph Gubbins and Andreas Vlachos

A Walk-Based Semantically Enriched Tree Kernel Over Distributed Word Representations
Shashank Srivastava, Dirk Hovy and Eduard Hovy

Automatic Idiom Identification in Wiktionary
Grace Muzny and Luke Zettlemoyer

Elephant: Sequence Labeling for Word and Sentence Segmentation
Kilian Evang, Valerio Basile, Grzegorz Chrupała and Johan Bos

Detecting Compositionality of Multi-Word Expressions using Nearest Neighbours in Vector Space Models
Douwe Kiela and Stephen Clark

Naive Bayes Word Sense Induction
Do Kook Choe and Eugene Charniak

The VerbCorner Project: Toward an Empirically-Based Semantic Decomposition of Verbs
Joshua K. Hartshorne, Claire Bonial and Martha Palmer

Saturday, October 19, 2013 (continued)

Where Not to Eat? Improving Public Policy by Predicting Hygiene Inspections Using Online Reviews
Jun Seok Kang, Polina Kuznetsova, Michael Luca and Yejin Choi

Automatically Identifying Pseudepigraphic Texts
Moshe Koppel and Shachar Seidman

Sunday, October 20, 2013

(8:00-9:00) Breakfast

(9:15-10:30) Second invited talk: Fernando Pereira

(10:30-11:00) Break

(11:00-12:35) Machine Learning for NLP

11:25–11:50 *Dynamic Feature Selection for Dependency Parsing*
He He, Hal Daumé III and Jason Eisner

11:50–12:15 *Semi-Supervised Representation Learning for Cross-Lingual Text Classification*
Min Xiao and Yuhong Guo

12:15–12:35 *Using Crowdsourcing to get Representations based on Regular Expressions*
Anders Søgaard, Hector Martinez, Jakob Elming and Anders Johannsen

(11:00-12:35) Summarization and Generation

11:00–11:25 *Overcoming the Lack of Parallel Data in Sentence Compression*
Katja Filippova and Yasemin Altun

11:25–11:50 *Fast Joint Compression and Summarization via Graph Cuts*
Xian Qian and Yang Liu

11:50–12:15 *Inducing Document Plans for Concept-to-Text Generation*
Ioannis Konstas and Mirella Lapata

Sunday, October 20, 2013 (continued)

12:15–12:35 *Single-Document Summarization as a Tree Knapsack Problem*
Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda and Masaaki Nagata

(11:00-12:35) Information Extraction and Social Media Analysis

11:00–11:25 *A Hierarchical Entity-Based Approach to Structuralize User Generated Content in Social Media: A Case of Yahoo! Answers*
Baichuan Li, Jing Liu, Chin-Yew Lin, Irwin King and Michael R. Lyu

11:25–11:50 *Semantic Parsing on Freebase from Question-Answer Pairs*
Jonathan Berant, Andrew Chou, Roy Frostig and Percy Liang

11:50–12:15 *Scaling Semantic Parsers with On-the-Fly Ontology Matching*
Tom Kwiatkowski, Eunsol Choi, Yoav Artzi and Luke Zettlemoyer

12:15–12:35 *Classifying Message Board Posts with an Extracted Lexicon of Patient Attributes*
Ruihong Huang and Ellen Riloff

(12:35-2:00) Lunch

(2:00-3:35) Machine Translation II

2:00–2:25 *Lexical Chain Based Cohesion Models for Document-Level Statistical Machine Translation*
Deyi Xiong, Yang Ding, Min Zhang and Chew Lim Tan

2:50–3:15 *A Convex Alternative to IBM Model 2*
Andrei Simion, Michael Collins and Cliff Stein

3:15–3:35 *Pair Language Models for Deriving Alternative Pronunciations and Spellings from Pronunciation Dictionaries*
Russell Beckley and Brian Roark

Sunday, October 20, 2013 (continued)

(2:00-3:35) Semantics II

2:00–2:25 *Prior Disambiguation of Word Tensors for Constructing Sentence Vectors*
Dimitri Kartsaklis and Mehrnoosh Sadrzadeh

2:25–2:50 *Multi-Relational Latent Semantic Analysis*
Kai-Wei Chang, Wen-tau Yih and Christopher Meek

2:50–3:15 *A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else)*
Ivan Vulić and Marie-Francine Moens

3:15–3:35 *Deriving Adjectival Scales from Continuous Space Word Representations*
Joo-Kyung Kim and Marie-Catherine de Marneffe

(2:00-3:35) Opinion Mining and Sentiment Analysis I

2:00–2:25 *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*
Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng and Christopher Potts

2:25–2:50 *Open Domain Targeted Sentiment*
Margaret Mitchell, Jacqui Aguilar, Theresa Wilson and Benjamin Van Durme

2:50–3:15 *Exploiting Domain Knowledge in Aspect Extraction*
Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos and Riddhiman Ghosh

(3:35-4:05) Break

Sunday, October 20, 2013 (continued)

(4:05-5:55) Machine Translation III

4:05–4:30 *Dependency-Based Decipherment for Resource-Limited Machine Translation*
Qing Dou and Kevin Knight

4:30–4:55 *Translating into Morphologically Rich Languages with Synthetic Phrases*
Victor Chahuneau, Eva Schlinger, Noah A. Smith and Chris Dyer

4:55–5:20 *Boosting Cross-Language Retrieval by Learning Bilingual Phrase Associations from Relevance Rankings*
Artem Sokokov, Laura Jehl, Felix Hieber and Stefan Riezler

5:20–5:55 *Recurrent Continuous Translation Models*
Nal Kalchbrenner and Phil Blunsom

(4:05-5:55) Information Extraction II

4:05–4:30 *Learning Biological Processes with Global Constraints*
Aju Thalappillil Scaria, Jonathan Berant, Mengqiu Wang, Peter Clark, Justin Lewis, Brittany Harding and Christopher D. Manning

4:30–4:55 *Generating Coherent Event Schemas at Scale*
Niranjan Balasubramanian, Stephen Soderland, Mausam and Oren Etzioni

5:20–5:55 *Orthonormal Explicit Topic Analysis for Cross-Lingual Document Matching*
John Philip McCrae, Philipp Cimiano and Roman Klinger

(4:05-5:55) NLP Applications II

4:05–4:30 *Automated Essay Scoring by Maximizing Human-Machine Agreement*
Hongbo Chen and Ben He

4:55–5:20 *Success with Style: Using Writing Style to Predict the Success of Novels*
Vikas Ganjigunte Ashok, Song Feng and Yejin Choi

5:20–5:55 *A Generative Joint, Additive, Sequential Model of Topics and Speech Acts in Patient-Doctor Communication*
Byron C. Wallace, Thomas A Trikalinos, M. Barton Laws, Ira B. Wilson and Eugene Charniak

Monday, October 21, 2013

(8:00-9:00) Breakfast

(9:00-10:35) Information Extraction III

9:00–9:25 *Harvesting Parallel News Streams to Generate Paraphrases of Event Relations*
Congle Zhang and Daniel S. Weld

9:25–9:50 *Relational Inference for Wikification*
Xiao Cheng and Dan Roth

9:50–10:15 *Event Schema Induction with a Probabilistic Entity-Driven Model*
Nathanael Chambers

10:15–10:35 *Using Soft Constraints in Joint Inference for Clinical Concept Recognition*
Prateek Jindal and Dan Roth

(9:00-10:35) Opinion Mining and Sentiment Analysis II

9:00–9:25 *Exploring Demographic Language Variations to Improve Multilingual Sentiment Analysis in Social Media*
Svitlana Volkova, Theresa Wilson and David Yarowsky

9:25–9:50 *Opinion Mining in Newspaper Articles by Entropy-Based Word Connections*
Thomas Scholz and Stefan Conrad

9:50–10:15 *Collective Opinion Target Extraction in Chinese Microblogs*
Xinjie Zhou, Xiaojun Wan and Jianguo Xiao

10:15–10:35 *Detecting Promotional Content in Wikipedia*
Shruti Bhosale, Heath Vinicombe and Raymond Mooney

Monday, October 21, 2013 (continued)

(9:00-10:35) NLP for Social Media II

9:00–9:25 *Learning Topics and Positions from Debatepedia*
Swapna Gottipati, Minghui Qiu, Yanchuan Sim, Jing Jiang and Noah A. Smith

9:25–9:50 *A Unified Model for Topics, Events and Users on Twitter*
Qiming Diao and Jing Jiang

9:50–10:15 *Authorship Attribution of Micro-Messages*
Roy Schwartz, Oren Tsur, Ari Rappoport and Moshe Koppel

10:15–10:35 *Detection of Product Comparisons - How Far Does an Out-of-the-Box Semantic Role Labeling System Take You?*
Wiltrud Kessler and Jonas Kuhn

(10:35–11:00) Break

(11:00-11:45) Parsing

11:00–11:25 *A Multi-Teraflop Constituency Parser using GPUs*
John Canny, David Hall and Dan Klein

11:25–11:45 *Fish Transporters and Miracle Homes: How Compositional Distributional Semantics can Help NP Parsing*
Angeliki Lazaridou, Eva Maria Vecchi and Marco Baroni

(11:00-11:45) Semantics III

11:00–11:25 *Learning Distributions over Logical Forms for Referring Expression Generation*
Nicholas FitzGerald, Yoav Artzi and Luke Zettlemoyer

11:25–11:45 *Learning to Rank Lexical Substitutions*
György Szarvas, Róbert Busa-Fekete and Eyke Hüllermeier

Monday, October 21, 2013 (continued)

(11:00-11:45) NLP Applications III

11:00–11:25 *Identifying Manipulated Offerings on Review Portals*
Jiwei Li, Myle Ott and Claire Cardie

11:25–11:45 *Well-Argued Recommendation: Adaptive Models Based on Words in Recommender Systems*
Julien Gaillard, Marc El-Beze, Eitan Altman and Emmanuel Ethis

(11:45–12:15) SIGDAT business meeting

(12:15–1:45) Lunch

(1:45–3:15) Plenary session I

1:45–2:15 *Regularized Minimum Error Rate Training*
Michel Galley, Chris Quirk, Colin Cherry and Kristina Toutanova

2:15–2:45 *Of Words, Eyes and Brains: Correlating Image-Based Distributional Semantic Models with Neural Representations of Concepts*
Andrew J. Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio and Marco Baroni

2:45–3:15 *Easy Victories and Uphill Battles in Coreference Resolution*
Greg Durrett and Dan Klein

(3:15–3:45) Break

(3:45–4:45) Plenary Session II

3:45–4:15 *Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction*
Valentin I. Spitkovsky, Hiyan Alshawi and Daniel Jurafsky

4:15–4:45 *Cross-Lingual Discriminative Learning of Sequence Models with Posterior Regularization*
Kuzman Ganchev and Dipanjan Das

(4:45–5:15) Closing session

Event-based Time Label Propagation for Automatic Dating of News Articles

Tao Ge **Baobao Chang*** **Sujian Li** **Zhifang Sui**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
No.5 Yiheyuan Road, Haidian District, Beijing, P.R.China, 100871
{getao,chbb,lisujian,szf}@pku.edu.cn

Abstract

Since many applications such as timeline summaries and temporal IR involving temporal analysis rely on document timestamps, the task of automatic dating of documents has been increasingly important. Instead of using feature-based methods as conventional models, our method attempts to date documents in a year level by exploiting relative temporal relations between documents and events, which are very effective for dating documents. Based on this intuition, we proposed an event-based time label propagation model called confidence boosting in which time label information can be propagated between documents and events on a bipartite graph. The experiments show that our event-based propagation model can predict document timestamps in high accuracy and the model combined with a MaxEnt classifier outperforms the state-of-the-art method for this task especially when the size of the training set is small.

1 Introduction

Time is an important dimension of any information space and can be useful in information retrieval, question-answering systems and timeline summaries. In the applications involving temporal analysis, document timestamps are very useful. For instance, temporal information retrieval models take into consideration the document's creation time for document retrieval and ranking (Kalczynski and Chou, 2005; Berberich et al., 2007) for better dealing with time-sensitive queries; some information retrieval applications such as Google Scholar can list articles published during the time a user specifies for better satisfying users' needs. In addition, timeline summarization techniques (Hu et al., 2011; Binh Tran et al., 2013) and some event-event ordering models (Chambers and Jurafsky, 2008; Yoshikawa et al., 2009) also rely on the timestamps. Unfortunately, many documents on the web do not have a credible timestamp, as Chambers (2012) reported. Therefore, it is significant to date documents, that is to predict document creation time.

One typical method for dating document is based on temporal language models, which were first used for dating by de Jong et al. (2005). They learned language models (unigram) for specific time periods and scored articles with normalized log-likelihood ratio scores. The other typical approach for the task was proposed by Nathanael Chambers (2012). In Chambers's work, discriminative classifiers – maximum entropy (MaxEnt) classifiers were used by incorporating linguistic features and temporal constraints for training, which outperforms the previous temporal language models on a subset of Gigaword Corpus (Graff et al., 2003).

However, the conventional methods have some limitations because they predict creation time of documents mainly based on feature-based models without understanding content of documents, which may lead to wrong predictions in some cases. For instance, assume that $D1$ and $D2$ are documents whose content is given as follows:

> ($D1$) Sudan *last year* accused Eritrea of backing an offensive by rebels in the eastern border region.

*Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1–11,
Seattle, Washington, USA, 18-21 October 2013.

($D2$) *Two years ago*, Sudan accused Eritrea of backing an offensive by rebels in the eastern border region.

Since $D1$ and $D2$ share many important features, the previous dating methods are very likely to predict the same timestamp for the two documents. However, it will be easy to infer that the creation time of $D1$ should be one year earlier than that of $D2$ if we analyze the content of the two documents.

Unlike the previous methods, this paper exploits relative temporal relations between events and documents for dating documents on the basis of an understanding of document content.

It is known that each event in a news article has a relative temporal relation with the document. By analyzing the relative temporal relation, time of the event can be known if we know the document timestamp; on the other hand, if the time of an event is known, it can also be used to predict the creation time of documents mentioning the event, which can be best demonstrated with the above-mentioned example of $D1$ and $D2$. In the example, "*last year*" is an important cue to infer that the event mentioned by the documents occurred in 2002 if we know the timestamp of $D1$ is 2003. With the information that the event occurred in 2002, it can also be inferred from the temporal expression "*Two years ago*" that $D2$ was written in 2004. In this way, the timestamp of the labeled document ($D1$) is propagated to the unlabeled document ($D2$) through the event both of them mention, which is the main intuition of this paper.

In fact, this intuition seems practical to date documents on the web because web data is very redundant. Many documents on the web can be connected via events because an event is usually mentioned by different documents. According to our analysis of a collection of news articles spanning 5 years, it is found that an event is mentioned by 3.44 news articles on average; on the other hand, a document usually refers to multiple events. Therefore, if one knows a document timestamp, time of events the document mentions can be obtained by analyzing the relative temporal relations between the document and the events. Likewise, if the time of an event is known, then it can be used to predict creation time of the documents which mention it.

Based on the intuition, we proposed an event-based time label propagation model called confidence boosting in which timestamps are propagated according to relative temporal relations between documents and events. In this way, documents can be dated with an understanding of content so that this model can date document more credibly. To our knowledge, it is the first time that the relative temporal relations between documents and events are exploited for dating documents, which is proved to be effective by the experimental results.

2 Event-based Time Label Propogation

As mentioned above, the relative temporal relations between documents and events are useful for dating documents. By analyzing the temporal relations, even if there are only a small number of documents labeled with timestamps, this information can be propagated to documents connected with them on a bipartite graph using breadth first traversal (BFS).

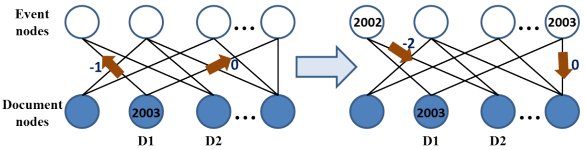


Figure 1: An example of BFS-based propagation

As shown in figure 1, there are two kinds of nodes in the bipartite graph. A document node is a single document while an event node represents an event. The edge between a document node and an event node means that the document mentions the event. Also, the edge carries the information of the relative temporal relation between the document and the event. The label propagation from node i to node j will occur if BFS condition which is defined as follows is satisfied:

$$\begin{cases} e_{ij} \in E \\ i \in L \text{ and } j \notin L \end{cases} \quad \text{(BFS condition)}$$

When the timestamp of i is propagated to j:

$$Y(j) = Y(i) + \delta(i,j)$$
$$L = L \cup \{j\}$$

where E is the set of edges of the bipartite graph, e_{ij} denotes the edge between node i and j, L is the set of nodes which have been already labeled with timestamps, $Y(i)$ is the year of node i and $\delta(i,j)$ is the relative temporal relation between node i and j.

In figure 1, the timestamp of document $D1$ is 2003, which is known. This information can be propagated to its adjacent nodes i.e. the event nodes it mentions according to the relative temporal relations. Then, these event nodes propagate their timestamps to other documents which mention them. By repeating this process, the timestamp of the document can be propagated to documents which are reachable from the initially labeled document on the bipartite graph.

Although the BFS-based propagation process can propagate timestamps from few labeled documents to a large number of unlabeled ones, it has two shortcomings for this task. First, once one timestamp is propagated incorrectly, this error will lead to more mistakes in the following propagations. If such an error occurred at the beginning of the propagation process, it would lead to propagation of errors. Second, BFS-based method cannot address conflict of predictions during propagation, which is shown in figure 2.

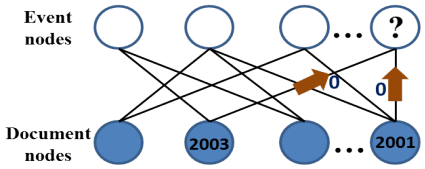


Figure 2: Conflict of predictions during propagation

To address the problems of the BFS-based method, we proposed a novel propagation model called confidence boosting model which improves the BFS-based model by optimizing the global confidence of the bipartite graph. In the confidence boosting model, every node in the bipartite graph has a confidence which measures the credibility of the predicted timestamp of the node. When the timestamp of a node is propagated to other nodes, its confidence will be also propagated to the target nodes with some loss. The loss of confidence is called confidence decay. Formally, the confidence decay process is described as follows:

$$c(j) = c(i) \times \sigma(i, j)$$

where $c(i)$ denotes confidence of node i and $\sigma(i, j)$ is the decay factor from node i to node j. For guaranteeing that timestamps can be propagated on the bipartite graph credibly, we define the following condition which is called CB (Confidence Boosting) condition:

$$\begin{cases} e_{ij} \in E \\ c(i) \times \sigma(i, j) > c(j) \end{cases} \quad \text{(CB condition)}$$

In the confidence boosting model, propagation from node i to node j will occur only if CB condition is satisfied. When timestamps are propagated on the bipartite graph, timestamps and confidence of nodes will be updated dynamically. A node with high confidence is more active than nodes with low confidence to propagate its timestamp because a node with high confidence is more likely to satisfy the CB condition for propagating its timestamp. Moreover, a prediction with low confidence can be corrected by the prediction with high confidence. Therefore, the confidence boosting model can address both propagation of errors and conflict of predictions which cannot be tackled by the BFS-based model.

However, there are challenges for running such propagation models in practice. First, the relative temporal relations between documents and events are usually unavailable. Second, events extracted from different documents do not have any connection even if they refer to the same event. Therefore, each event is connected with only one document in the bipartite graph and thus cannot propagate its timestamp to other documents unless we perform event coreference resolution. Third, propagations from generic events are very likely to lead to propagation errors because generic events can happen in any year. Also, how to set the confidence and decay factors reasonably in practice for a confidence boosting model is worthy of investigation. All these challenges for the propagation models and their corresponding solutions will be discussed in Section 3.

3 Details of Event-based Propagation Models

In this section, details of the event-based time label propagation models including challenges and their corresponding solutions are presented. We first discuss the event extraction and processing involving relative temporal relation mining, event coreference resolution and distinguishing specific extractions from generic ones in Section 3.1. Then, we show the confidence boosting algorithm in detail in Section 3.2.

3.1 Event extraction and processing

As mentioned in previous sections, events play a key role in the propagation models. We define an event as a Subject-Predicate-Object (SPO) triple. To extract events from raw text, an open information extraction software - ReVerb (Fader et al., 2011) is used. ReVerb is a program that automatically identifies and extracts relationships from English sentences. It takes raw text as input and outputs SPO triples which are called extractions.

However, extractions extracted by ReVerb cannot be used directly for our propagation models for three main reasons. First, the relative temporal relations between documents and the extractions are unavailable. Second, the extractions extracted from different documents do not have any connection even if they refer to the same event. Third, propagations from generic events are very likely to lead to propagation errors.

For addressing the three challenges for the propagation models, we first presented a rule-based method for mining the relative temporal relations between extractions and documents in Section 3.1.1. Then, an efficient event coreference resolution method is introduced in Section 3.1.2. Finally, the method for distinguishing specific extractions from generic ones is shown in Section 3.1.3.

3.1.1 Relative temporal relation mining

We used a rule-based method to extract temporal expressions and used Stanford parser (De Marneffe et al., 2006) to analyze association between the temporal expressions and the extractions. Specifically, we define that an extraction is associated with a temporal expression if there is an arc from the predicate of the extraction to the temporal expression in the dependency tree. For a certain extraction, there are the following four cases whose instances are shown in table 1 for handling.

Case 1: The extraction is associated with an absolute temporal expressions with year mentions in the sentence.

In this case, the time of the extraction is equal to the year mention:

$$Y(ex) = YearMention$$

For the example in table 1, $Y(ex) = 1999$.

Case 2: The extraction is associated with a relative temporal expression (not involving year) in the sentence.

Case	Instance
1	*In 1999*, South Korea exported 89,000 tons of pork to Japan.
2	*In April*, however, the BOI investments showed marked improvement.
	Last month, Kazini vowed to resign his top army job.
3	Julius Erving moved with his family to Florida *three years ago*.
4	The meeting focused on ways to revive the stalled Mideast peace process.

Table 1: Instances of various temporal expressions

In this case, the time of the extraction is equal to the creation time of the document:

$$Y(ex) = Y(d)$$

Case 3: The extraction is associated with a relative temporal expression (involving specific year gap) in the sentence.

In this case, the time of the extraction is computed as follows:

$$Y(ex) = Y(d) \pm YearGap$$

For the example in table 1, $Y(ex) = Y(d) - 3$.

Case 4: The extraction is not associated with any temporal expression in the sentence or the other cases.

In this case, it is difficult to recognize the relative temporal relations. However, timeliness can be leveraged to determine the relations as a heuristic method. It is known that timeliness is an important feature of news so that events reported by a news article usually took place a couple of days or weeks before the article was written. Therefore, we heuristically consider the year of the extraction is the same with that of its source document in this case:

$$Y(ex) = Y(d)$$

In the cases except case 1, the relative temporal relation between an extraction and the document it comes from can be determined. To evaluate the performance of the rule-based method, we sampled 3,000 extractions from documents written in the year of 1995-1999 of Gigaword corpus and manually labeled these extractions with a timestamp based on their context and their corresponding document timestamps as golden standard. Table 2 shows

the accuracy of each case which will be used as a part of the decay factor in the confidence boosting model.

Case	Accuracy
1	0.774(168/217)
2	0.994(844/849)
3	0.836(281/336)
4	0.861(1376/1598)
Total	0.890(2669/3000)

Table 2: Accuracy of the four cases

We define the set of these determined relative temporal relations R as follows:

$$R = \{r_{d,ex} | d = doc(ex), ex \in C_2 \cup C_3 \cup C_4\}$$
$$r_{d,ex} =< d, ex, \delta(d, ex) >$$
$$\delta(d, ex) = -\delta(ex, d) = \{0, \pm 1, \pm 2, \pm 3, ...\}$$

where C_k is the set of extractions in case k and $doc(ex)$ is the document which extraction ex comes from. $r_{d,ex}$ is a triple describing the relative temporal relation between d and ex. For example, triple $r_{d,ex} =< d, ex, -1 >$ means that the time of extraction ex is one year before the time of document d.

3.1.2 Event coreference resolution

Extractions from different documents have no connections. However, there are a great number of extractions referring to the same event. For finding such coreferential event extractions efficiently, hierarchical agglomerative clustering (HAC) is used to cluster highly similar extractions into one cluster. We use cosine to measure the similarity between extractions and select bag of words as features. Note that it is less meaningful to cluster the extractions from the same document because coreferential extractions from the same document are not helpful for timestamp propagations. For this reason, similarity between extractions from the same documents is set to 0.

For HAC, selection of threshold is important. If the threshold is set too high, only a few extractions can be clustered despite high purity; on the contrary, if the threshold is set too low, purity of clusters will descend. In fact, selection of threshold is a trade-off between the precision and recall of event coreference resolution. For selecting a suitable threshold, extractions from documents written in 1995-1999 are used as a development set.

In practice, it is difficult for us to directly evaluate the performance of the coreference resolution of event extractions without golden standard which requires much labors for manual annotations. Alternatively, entropy which measures the purity of clusters is used for evaluation because it can indirectly reflect the precision of coreference resolution to some extent:

$$\text{Entropy} = -\sum_j \frac{n_j}{n} \sum_i P(i, j) \times \log_2 P(i, j)$$

where $P(i, j)$ is the probability of finding an extraction whose timestamp is i in the cluster j, n_j is the number of items in cluster j and n is the total number of extractions. Note that timestamp of an extraction is assigned based on its document timestamp using the method proposed in Section 3.1.1.

Figure 3 shows the effect of selection of the threshold on cluster performance. It can be found that when the threshold reaches 0.8, the entropy starts descending gently and is low enough. Since we want to find as many coreferential extractions as possible on the premise that the precision is good, the threshold is set to 0.8. Note that extractions which are single in one cluster will be filtered out because they do not have any connections with any other documents.

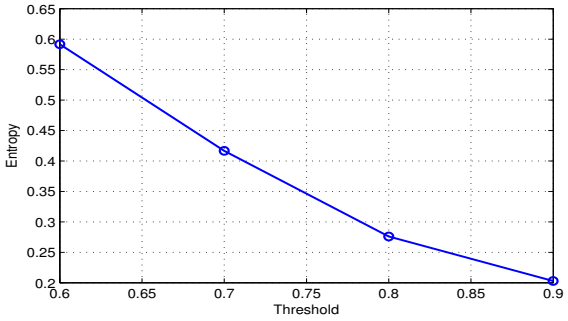


Figure 3: Entropy of clusters under different thresholds

3.1.3 Distinguishing specific events from generic ones

Not all extractions extracted by ReVerb refer to a specific event. For instance, the extraction "Germany's DAX index was down 0.2 percent" is undesirable for our task because it refers to a generic

event and this event may occur in any year. In other words, it is not able to indicate a certain timestamp and thus propagations from a generic event node are very likely to result in propagation errors. In contrast, the extraction "some of the provinces in China were hit by SARS" refers to a specific event which took place in 2003. For our task, such specific event extractions which are associated with one certain timestamp are desirable. For the sake of distinguishing such extractions from the generic ones, a MaxEnt classifier is used to classify extractions as either specific ones or generic ones.

Training Set Generation A training set is indispensable for training a MaxEnt classifier. In order to generate training examples, we performed HAC discussed in Section 3.1.2 for event coreference resolution on extractions from all documents written in May and June of 1995-1999 and then analyzed each cluster. If extractions in a cluster have different timestamps, then the extractions in this cluster will be labeled as generic extractions (negative); otherwise, extractions in the cluster are labeled as specific ones (positive). In this way, the training set can be generated without manually labeling. To avoid bias of positive and negative examples, we sampled 3,500 positive examples and 3,500 negative examples to train the model.

Feature Selection The following features were selected for training:

Named Entities: People and places are often discussed during specific time periods, particularly in news genre. Intuitively, if an extraction contains specific named entities then this extraction is less likely to be a generic event. If an extraction contains named entities, types and uninterrupted tokens of the named entities will be included as features.

Numeral: According to our analysis of the training set generated by the above-mentioned method, generic extractions usually contain numerals. For example, the extraction "15 people died in this accident" and the extraction "225 people died in this accident" have the same tokens except numerals and they are labeled as a generic event because they are clustered into one group due to high similarity but they in fact refer to different events happening in different years. Therefore, if an extraction contains numerals, the feature "NUM" will be included.

Bag of words: Bag of words can also be an indicator of specific extractions and generic ones. For example, an extraction containing 'stock', 'index', 'fell' and 'exchange' is probably a generic one.

The model obtained after training can be used to predict whether an extraction is a specific one. We define $P(S = 1|ex)$ as the probability that an extraction is a specific one, which can be provided by the classifier. Extractions whose probability to be a specific one is less than 0.05 are filtered out. For the other extractions, this probability is used as a part of the decay factor in the confidence boosting model, which will be discussed in detail in Section 3.2.

3.2 Confidence boosting

After extracting and processing the event extractions, relative temporal relations between documents and events can be constructed. This can be formally represented by a bipartite graph G=$\langle V, E\rangle$. There are two kinds of nodes on the bipartite graph: document nodes and event nodes. Slightly different with the event node mentioned in Section 2, an event node in practice is a cluster of coreferential extractions and it can be connected with multiple document nodes. Note that the bipartite graph does not contain any isolate node. For briefness, we define $DNode$ as the set of document nodes and $ENode$ as the set of event nodes. The set of edges E is formally defined as follows:
$E = \{e_{ij}, e_{ji}|i \in DNode, j \in ENode, r_{i,j} \in R\}$
where R is the set of relative temporal relations defined as Section 3.1.1.

3.2.1 Confidence and decay factor

As mentioned in Section 2, the confidence of a node measures the credibility of the predicted timestamp. According to the definition, we set the confidence of initially labeled nodes to 1 and set confidence of nodes without any timestamp to 0 in practice. When the timestamp of a node is propagated to another node, its confidence will be propagated to the target node with some loss, as discussed in Section 2. The confidence loss is caused by two factors in practice. The first one is the credibility of the relative temporal relation between two nodes and the other one depends on whether an extraction refers to a specific event.

Relative temporal relations between documents

and extractions we mined using the rule-based method in Section 3.1.1 are not absolutely correct. The credibility of the relations has an effect on the confidence decay. Formally, we used $\pi(i, j)$ to denote the credibility of the relative temporal relation between node i and node j. The credibility of a relative temporal relation in each case can be estimated through table 2. If the credibility of the relative temporal relation between i and j is low, propagation from node i to j probably leads to error. Therefore, the confidence loss should be much in this case. On contrary, if the relation is highly credible, it will be less likely that propagation errors occur. Therefore, the confidence loss should be little.

In addition, whether an extraction refers to a generic event or a specific one exerts an impact on the confidence loss. If an extraction refers to a generic event, then the extractions in the same cluster with it probably have different timestamps. Since our propagation model assumes that extractions in a cluster are coreferent and thus they should have the same timestamp, propagations from a generic event node are very likely to result in propagation errors. Therefore, the timestamp of a generic event node in fact is less credible for propagations and confidence of such event nodes should be low for limiting propagations from the nodes. For this reason, propagation from a document node to a generic event node leads to much loss of confidence. We define the probability that an event node refers to a specific event as follows:

$$P(S = 1|enode) = \frac{1}{|\mathbb{C}|} \sum_{ex \in \mathbb{C}} P(S = 1|ex)$$

where $\mathbb{C}$ is the set of extractions in the event node and $P(S = 1|ex)$ is the probability that an extraction refers to a specific event, which can be provided by the MaxEnt classifier discussed in Section 3.1.3.

Considering the two factors for confidence loss, we formally define the decay factor by (1).

$$\sigma(s, t) = \begin{cases} \pi(s, t) & \textbf{if } t \in DNode \\ \pi(s, t) \times P(S = 1|t) & \textbf{otherwise} \end{cases} \quad (1)$$

3.2.2 Confidence boosting algorithm

In confidence boosting model, the propagation from i to j will occur only if the CB condition is satisfied. The confidence boosting propagation process can be described as figure 4.

Algorithm: Confidence Boosting
Input: Array Y, Array c, Array δ, Array σ
Output: Array Y
Initialize Array c and Array Y
while $\exists i, j$ s.t. CB condition
  $Y(j) = Y(i) + \delta(i, j)$
  $c(j) = c(i) \times \sigma(i, j)$
end **while**

Figure 4: Algorithm of confidence boosting

Whenever timestamps are propagated to other nodes, the global confidence of the bipartite graph will increase. For this reason, this propagation process is called confidence boosting. In this model, a node with high confidence is more active than nodes with low confidence to propagate its timestamp. Moreover, a prediction with low confidence can be corrected by the prediction with high confidence. Therefore, the confidence boosting model can alleviate the problem of propagation of errors to some extent and handle conflict of predictions. Thus, it can propagate timestamps more credibly than the BFS-based model. It can also be proved that each node on the bipartite graph must reach the highest confidence it can reach so that the global confidence of the bipartite graph must be optimal when the confidence boosting propagation process ends regardless of propagation orders, which will be discussed in Section 3.2.3.

3.2.3 Proof of the optimality of confidence boosting

Proof by contradiction can be used to prove that propagation orders do not affect the optimality of the confidence boosting model.

Proof Assume by contradiction that there is some node that does not reach its highest confidence it can reach when a confidence boosting process in propagation order A ends:

$$\exists v_t \text{ s.t. } c_A(v_t) < c^*(v_t)$$

where $c_A(v_t)$ is the confidence of v_t when the propagation process in order A ends and $c^*(v_t)$ is the highest confidence that v_t can reach. Assume that $(v_1, v_2, \cdots, v_{t-1}, v_t)$ is the optimal propagation

path from the propagation source node v_1 to the node v_t that leads to the highest confidence of v_t, which means that $c^*(v_t) = c^*(v_{t-1}) \times \sigma(v_{t-1}, v_t)$, $c^*(v_{t-1}) = c^*(v_{t-2}) \times \sigma(v_{t-2}, v_{t-1})$, ..., $c^*(v_2) = c^*(v_1) \times \sigma(v_1, v_2)$. Then according to CB condition, since $c_A(v_{t-1}) \times \sigma(v_{t-1}, v_t) \leq c_A(v_t) < c^*(v_t) = c^*(v_{t-1}) \times \sigma(v_{t-1}, v_t)$, the inequality $c_A(v_{t-1}) < c^*(v_{t-1})$ must hold. Similarly, it can be easily inferred that $c_A(v_{t-2}) < c^*(v_{t-2})$ and finally $c_A(v_1) < c^*(v_1)$. Since v_1 is the source node whose timestamp is initially labeled and its confidence is 1, the inequality $c_A(v_1) < c^*(v_1)$ cannot hold. Thus, the assumption that $c_A(v_t) < c^*(v_t)$ cannot be satisfied. Therefore, it can be proved that each node on the bipartite graph must reach the highest confidence it can reach so that the global confidence of the bipartite graph must be optimal when confidence boosting propagation process ends no matter what order time labels are propagated in.

4 Experiments

In this section, we evaluate the performance of our time label propagation models and different automatic document dating models on the Gigaword dataset. We first present the experimental setting. Then we show experimental results and perform an analysis.

4.1 Experimental Setting

Dataset To simulate the environment of the web where data is very redundant, we use all documents written in April, June, July and September of 2000-2004 of Gigaword Corpus as dataset instead of sampling a subset of documents from each period. The dataset contains 900,199 news articles.

Pre-processing Many extractions extracted by ReVerb are short and uninformative and do not carry any valuable information for propagating temporal information. Also, some extractions do not refer to events which already happened. These extractions may affect the performance of event coreference resolution and the rule-based method proposed in Section 3.1.1 for mining relative temporal relations. Therefore, we filter out these undesirable extractions in advance with a rule-based method. The rules are shown in table 3. This preprocessing removes large numbers of "bad" extractions which are undesirable for our task. As a result, not only computation efficiency but also precision of event coreference resolution will be improved.

Rule1	If the number of tokens of the extraction is less than 5 then this extraction will be filtered out.
Rule2	If the maximum idf of terms of the extraction is less than 3.0 then this extraction will be filtered out.
Rule3	If the tense of the extraction is not past tense then this extraction will be filtered out.
Rule4	If the extraction is the content of direct quotation then this extraction will be filtered out.

Table 3: Pre-processing Rules

$\|DNode\|$	550,124
$\|ENode\|$	968,064
$\|E\|$	3,104,666

Table 4: Basic information of the bi-partite graph

Basic information of the document-event bipartite graph constructed is shown in table 4.

Evaluation To evaluate the performance of the propagation models for the task of dating on different sizes of the training set, we used different sizes of the labeled documents for training and considered the remaining documents as the test set. Note that the training set is randomly sampled from the dataset. To be more persuasive, we repeated above experiments for five times.

However, in the time label propagation process, not all documents can be labeled. For those documents which cannot be labeled in the process of propagation, a MaxEnt classifier serves as a complementary approach to predict their timestamps. For the MaxEnt classifier, unigrams and named entities are simply selected as features and the initially labeled documents as well as documents labeled during propagation process are used for training.

Baseline methods are temporal language models proposed by de Jong et al. (2005) and the state-of-the-art discriminative classifier with linguistic features and temporal constraints which was proposed

Initially Labeled	1k	5k	10k	50k	100k	200k	500k
Reached Min	443980	448653	453022	484562	518603	599724	732701
Reached Max	444266	448998	454028	484996	519333	579878	732799
Reached Avg	444107	448742	453786	484622	519110	579835	732758
Prop Ratio	444.1	89.7	45.4	9.7	5.2	2.9	1.5
Prop acc(BFS)	0.438	0.515	0.551	0.646	0.691	0.725	0.775
Prop acc(CB)	0.494	0.569	0.603	0.701	0.746	0.776	0.807

Table 5: Performance of Propagation

Initially Labeled	1k	5k	10k	50k	100k	200k	500k
Temporal LMs	0.277	0.323	0.353	0.412	0.422	0.425	0.420
Maxent(Unigrams)	0.326	0.378	0.407	0.486	0.517	0.553	0.590
Maxent(Unigrams+NER)	0.331	0.383	0.418	0.506	0.549	0.590	0.665
Chambers's	0.331	0.386	0.423	0.524	0.571	0.615	0.690
BFS+Maxent	0.459	0.508	0.533	0.595	0.626	0.658	0.707
CB+Maxent	**0.486**	**0.535**	**0.559**	**0.624**	**0.655**	**0.685**	**0.726**

Table 6: Overall accuracy of dating models

by Nathanael Chambers (2012). In Chambers's joint model, the interpolation parameter λ is set to 0.35 which is considered optimal in his work.

4.2 Experimental Results

Table 5 shows the performance of propagation models where *Reached* denotes the number of documents labeled when the propagation process ends, *prop ratio* and *prop accuracy* are defined as follows:

$$\text{Prop Ratio} = \frac{\#ReachedDocNodes}{\#LabeledDocNodes}$$

$$\text{Prop Accuracy} = \frac{\#CorrectDocNodes - \#LabeledDocNodes}{\#ReachedDocNodes - \#LabeledDocNodes}$$

where $\#LabeledDocNodes$ is the number of initially labeled document nodes which are documents in the training set and $\#ReachedDocNodes$ is the number of document nodes labeled when the propagation process ends.

Note that prop ratio and accuracy in table 5 are the mean of the prop ratio and accuracy of the five groups of experiments. It is clear that confidence boosting model improves the prop accuracy over BFS-based model. When only 1,000 documents are initially labeled with timestamps, the confidence boosting model can propagate their timestamps to more than 400,000 documents with an accuracy of 0.494, approximately 12.8% relative improvement over the BFS counterpart, which proves effectiveness of the confidence boosting model.

However, as shown in table 5, hardly can the propagation process propagate timestamps to all documents. One reason is that the number of document nodes on the bipartite graph is only 550,124, approximately 61.1% of all documents. The other documents may not mention events which are also mentioned by other documents, which means they are isolate and thus are excluded from the bipartite graph. Also, the event coreference resolution phase does not guarantee finding all coreferential extractions; in other words, recall of event coreference resolution is not 100%. The other reason is that some documents are unreachable from the initially labeled nodes even if they are in the bipartite graph.

The overall accuracy of different dating models is shown in table 6. As with table 5, overall accuracy in table 6 is the average performance of models in the five groups of experiments. As reported by Nathanael Chambers (2012), the discriminative classifier performs much better than the temporal language models on the Gigaword dataset. In the case of 500,000 training examples, the Maxent classifier using unigram features outperforms the temporal language models by 40.5% relative accuracy. If the size of the training set is large enough, named

entities and linguistic features as well as temporal constraints will improve the overall accuracy significantly. However, if the size of the training set is small, these features will not result in much improvement.

Compared with the previous models, the propagation models predict the document timestamps much more accurately especially in the case where the size of the training set is small. When the size of the training set is 1,000, our BFS-based model and confidence boosting model combined with the MaxEnt classifier outperform Chambers's joint model which is considered the state-of-the-art model for the task of automatic dating of documents by 38.7% and 46.8% relative accuracy respectively. This is because the feature-based methods are not very reliable especially when the size of the training set is small. In contrast, our propagation models can predict timestamps of documents with an understanding of document content, which allows our method to date documents more credibly than the baseline methods. Also, by comparing table 5 with table 6, it can be found that prop accuracy is almost always higher than overall accuracy, which also verifies that the propagation models are more credible for dating document than the feature-based models. Moreover, data is so redundant that a great number of documents can be connected with events they share. Therefore, even if a small number of documents are labeled, the labeled information can be propagated to large numbers of articles through the connections between documents and events according to relative time relations. Even if the size of the training set is large, e.g. 500,000, our propagation models still outperform the state-of-the-art dating method. Additionally, some event nodes on the bipartite graph may be labeled with a timestamp during the process of propagation as a byproduct. The temporal information of the events would be useful for other temporal analysis tasks.

5 Related Work

In addition to work of de Jong et al. (2005) and Chambers (2012) introduced in previous sections, there is also other research focusing on the task of document dating. Kanhabua and Norvag (2009) improved temporal language models by incorporating temporal entropy and search statistics and applying two filtering techniques to the unigrams in the model. Kumar et al. (2011) is also based on the temporal language models, but more historically-oriented, which models the timeline from the present day back to the 18th century. In addition, they used KL-divergence instead of normalized log likelihood ratio to measure differences between a document and a time period's language model.

However, these methods are based on temporal language models so they also suffer from the problem of the method of de Jong et al. (2005). Therefore, they inevitably make wrong predictions in some cases, just as mentioned in Section 1. Compared with these methods, our event-based propagation models exploit relative temporal relations between documents and events for dating document on a basis of an understanding of document content, which is more reasonable and also proved to be more effective by the experimental results.

6 Conclusion

The main contribution of this paper is exploiting relative temporal relations between events and documents for the document dating task. Different with the conventional work which dates documents with feature-based methods, we proposed an event-based time label propagation model called confidence boosting in which timestamps are propagated on a document-event bipartite graph according to relative temporal relations between documents and events for dating documents on a basis of an understanding of document content. We discussed challenges for the propagation models and gave the corresponding solutions in detail. The experimental results show that our event-based propagation model can predict document timestamps in high accuracy and the model combined with a MaxEnt classifier outperforms the state-of-the-art method on a data-redundant dataset.

Acknowledgements

We thank the anonymous reviewers for their valuable suggestions. This paper is supported by NSFC Project 61075067, NSFC Project 61273318 and National Key Technology R&D Program (No: 2011BAH10B04-03).

References

Klaus Berberich, Srikanta Bedathur, Thomas Neumann, and Gerhard Weikum. 2007. A time machine for text search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 519–526. ACM.

Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen. 2013. Predicting relevant news events for timeline summaries. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 91–92. International World Wide Web Conferences Steering Committee.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 698–706. Association for Computational Linguistics.

Nathanael Chambers. 2012. Labeling documents with timestamps: Learning from their time expressions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 98–106. Association for Computational Linguistics.

FMG de Jong, Henning Rode, and Djoerd Hiemstra. 2005. Temporal language models for the disclosure of historical text. Royal Netherlands Academy of Arts and Sciences.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Po Hu, Minlie Huang, Peng Xu, Weichang Li, Adam K Usadi, and Xiaoyan Zhu. 2011. Generating breakpoint-based timeline overview for news topic retrospection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 260–269. IEEE.

Pawel Jan Kalczynski and Amy Chou. 2005. Temporal document retrieval model for business news archives. *Information processing management*, 41(3):635–650.

Nattiya Kanhabua and Kjetil Nørvåg. 2009. Using temporal language models for document dating. In *Machine Learning and Knowledge Discovery in Databases*, pages 738–741. Springer.

Abhimanu Kumar, Matthew Lease, and Jason Baldridge. 2011. Supervised language modeling for temporal resolution of texts. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2069–2072. ACM.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language: Volume 1-Volume 1*, pages 405–413. Association for Computational Linguistics.

Exploiting Discourse Analysis for Article-Wide Temporal Classification

Jun-Ping Ng[1], **Min-Yen Kan**[1,2], **Ziheng Lin**[3], **Wei Feng**[4], **Bin Chen**[5], **Jian Su**[5], **Chew-Lim Tan**[1]
[1]School of Computing, National University of Singapore, Singapore
[2]Interactive and Digital Media Institute, National University of Singapore, Singapore
[3]Research & Innovation, SAP Asia Pte Ltd, Singapore
[4]Department of Computer Science, University of Toronto, Canada
[5]Institute for Infocomm Research, Singapore
junping@comp.nus.edu.sg

Abstract

In this paper we classify the temporal relations between pairs of events on an article-wide basis. This is in contrast to much of the existing literature which focuses on just event pairs which are found within the same or adjacent sentences. To achieve this, we leverage on discourse analysis as we believe that it provides more useful semantic information than typical lexico-syntactic features. We propose the use of several discourse analysis frameworks, including 1) Rhetorical Structure Theory (RST), 2) PDTB-styled discourse relations, and 3) topical text segmentation. We explain how features derived from these frameworks can be effectively used with support vector machines (SVM) paired with convolution kernels. Experiments show that our proposal is effective in improving on the state-of-the-art significantly by as much as 16% in terms of F_1, even if we only adopt less-than-perfect automatic discourse analyzers and parsers. Making use of more accurate discourse analysis can further boost gains to 35%.

1 Introduction

A good amount of research had been invested in understanding temporal relationships within text. Particular areas of interest include determining the relationship between an event mention and a time expression (timex), as well as determining the relationship between two event mentions. The latter, which we refer to as event-event (*E-E*) temporal classification is the focus of this work.

For a given event pair which consists of two events $e1$ and $e2$ found *anywhere* within an article, we want to be able to determine if $e1$ happens before $e2$ (BEFORE), after $e2$ (AFTER), or within the same time span as $e2$ (OVERLAP).

Consider this sentence[1]:

> At least 19 people were **killed** and 114 people were **wounded** in Tuesday's southern Philippines airport blast, officials **said**, but reports said the death toll could climb to 30. (1)

Three event mentions found within the sentence are bolded. We say that there is an OVERLAP relationship between the "**killed** – **wounded**" event pair as these two events happened together after the airport blast. Similarly there is a BEFORE relationship between both the "**killed** – **said**", and "**wounded** – **said**" event pairs, as the death and injuries happened before reports from the officials.

Being able to infer these temporal relationships allows us to build up a better understanding of the text in question, and can aid several natural language understanding tasks such as information extraction and text summarization. For example, we can build up a temporal characterization of an article by constructing a temporal graph denoting the relationships between all events within an article (Verhagen et al., 2009). This can then be used to help construct an event timeline which layouts sequentially event mentions in the order they take place (Do et al., 2012). The temporal graph can also be used in text summarization, where temporal order can be used to improve sentence ordering and thereby the eventual generated summary (Barzilay et al., 2002).

Given the importance and value of temporal relations, the community has organized shared tasks

[1]From article AFP_ENG_20030304.0250 of the ACE 2005 corpus (ACE, 2005).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 12–23,
Seattle, Washington, USA, 18-21 October 2013.

to spur research efforts in this area, including the TempEval-1, -2 and -3 evaluation workshops (Verhagen et al., 2009; Verhagen et al., 2010; Uzzaman et al., 2012). Most related work in this area have focused primarily on the task defintitions of these evaluation workshops. In the task definitions, *E-E* temporal classification involves determining the relationship between events found within the same sentence, or in adjacent sentences. For brevity we will refer to this loosely as intra-sentence *E-E* temporal classification in the rest of this paper.

This definition however is limiting and insufficient. It was adopted as a trade-off between completeness, and the need to simplify the evaluation process (Verhagen et al., 2009). In particular, one deficiency is that it does not allow us to construct the complete temporal graph we seek. As illustrated in Figure 1, being able to perform only intra-sentence *E-E* temporal classification may result in a forest of disconnected temporal graphs. A sentence $s3$ separates events C and D, as such an intra-sentence *E-E* classification system will not be able to determine the temporal relationship between them. While we can determine the relationship between A and C in the figure with the use of temporal transitivity rules (Setzer et al., 2003; Verhagen, 2005), we cannot reliably determine the relationship between say A and D.

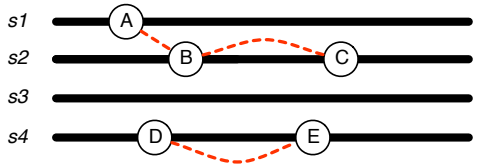


Figure 1: A disconnected temporal graph of events within an article. Horizontal lines depict sentences $s1$ to $s4$, and the circles identify events of interest.

In this work, we seek to overcome this limitation, and study what can enable effective article-wide *E-E* temporal classification. That is, we want to be able to determine the temporal relationship between two events located anywhere within an article.

The main contribution of our work is going beyond the surface lexical and syntactic features commonly adopted by existing state-of-the-art approaches. We suggest making use of semantically motivated features derived from discourse analysis instead, and show that these discourse features are superior.

While we are just focusing on *E-E* temporal classification, our work can complement other approaches such as the joint inference approach proposed by Do et al. (2012) and Yoshikawa et al. (2009) which builds on top of event-timex (*E-T*) and *E-E* temporal classification systems. We believe that improvements to the underlying *E-T* and *E-E* classification systems will help with global inference.

2 Related Work

Many researchers have worked on the *E-E* temporal classification problem, especially as part of the TempEval series of evaluation workshops. Bethard and Martin (2007) presented one of the earliest supervised machine learning systems, making use of support vector machines (SVM) with a variety of lexical and syntactic features. Kolya et al. (2010) described a conditional random field (CRF) based learner making use of similar features. Other researchers including Uzzaman and Allen (2010) and Ha et al. (2010) made use of Markov Logic Networks (MLN). By leveraging on the transitivity properties of temporal relationships (Setzer et al., 2003), they found that MLNs are useful in inferring new temporal relationships from known ones.

Recognizing that the temporal relationships between event pairs and time expressions are related, Yoshikawa et al. (2009) proposed the use of a joint inference model and showed that improvements in performance are obtained. However this gain is attributed to the joint inference model they had developed, making use of similar surface features.

To the best of our knowledge, the only piece of work to have gone beyond sentence boundaries and tackle the problem of article-wide *E-E* temporal classification is by Do et al. (2012). Making use of integer linear programming (ILP), they built a joint inference model which is capable of classifying temporal relationships between any event pair within a given document. They also showed that event co-reference information can be useful in determining these temporal relationships. However they did not make use of features directed specifically at determining the temporal relationships of event pairs

across different sentences. Other than event coreference information, they adopted the same mix of lexico-syntactic features.

Underlying these disparate data-driven methods for similar temporal processing tasks, the reviewed works all adopted a similar set of surface features including vocabulary features, part-of-speech tags, constituent grammar parses, governing grammar nodes and verb tenses, among others. We argue that these features are not sufficiently discriminative of temporal relationships because they do not explain how sentences are combined together, and thus are unable to properly differentiate between the different temporal classifications. Supporting our argument is the work of Smith (2010), where she argued that syntax cannot fully account for the underlying semantics beneath surface text. D'Souza and Ng (2013) found out as much, and showed that adopting richer linguistic features such as lexical relations from curated dictionaries (*e.g.* Webster and WordNet) as well as discourse relations help temporal classification. They had shown that the Penn Discourse TreeBank (PDTB) style (Prasad et al., 2008) discourse relations are useful. We expand on their study to assess the utility of adopting additional discourse frameworks as alternative and complementary views.

3 Making Use of Discourse

To highlight the deficiencies of surface features, we quote here an example from Lascarides and Asher (1993):

> [A] Max opened the door. The room was pitch dark.
> [B] Max switched off the light. The room was pitch dark. (2)

The two lines of text A and B in Example 2 have similar syntactic structure. Given only syntactic features, we may be drawn to conclude that they share similar temporal relationships. However in the first line of text, the events temporally OVERLAP, while in the second line they do not. Clearly, syntax alone is not going to be useful to help us arrive at the correct temporal relations.

If existing surface features are insufficient, what is sufficient? Given a *E-E* pair which crosses sentence boundaries, how can we determine the temporal relationship between them? We take our cue from the work of Lascarides and Asher (1993). They suggested instead that discourse relations hold the key to interpreting such temporal relationships.

Building on their observations, we believe that discourse analysis is integral to any solution for the problem of article-wide *E-E* temporal classification. We thus seek to exploit a series of different discourse analysis studies, including 1) the Rhetorical Structure Theory (RST) discourse framework, 2) Penn Discourse Treebank (PDTB)-styled discourse relations based on the lexicalized Tree Adjoining Grammar for Discourse (D-LTAG), and 3) topical text segmentation, and validate their effectiveness for temporal classification.

RST Discourse Framework. RST (Mann and Thompson, 1988) is a well-studied discourse analysis framework. In RST, a piece of text is split into a sequence of non-overlapping text fragments known as elementary discourse units (EDUs). Neighboring EDUs are related to each other by a typed relation. Most RST relations are *hypotactic*, where one of the two EDUs participating in the relationship is demarcated as a *nucleus*, and the other a *satellite*. The nucleus holds more importance, from the point of view of the writer, while the satellite's purpose is to provide more information to help with the understanding of the nucleus. Some RST relations are however *paratactic*, where the two participating EDUs are both marked as nuclei. A discourse tree can be composed by viewing each EDU as a leaf node. Nodes in the discourse tree are linked to one another via the discourse relations that hold between the EDUs.

RST discourse relations capture the semantic relation between two EDUs, and these often offer a clue to the temporal relationship between events in the two EDUs too. As an example, let us refer once again to Example 2. Recall that in the second line of text "**switched off**" happens BEFORE "**dark**". The RST discourse structure for the second line of text is shown on the left of Figure 2. We see that the two sentences are related via a "*Result*" discourse relation. This fits our intuition that when there is causation, there should be a BEFORE/AFTER relationship. The RST discourse relation in this case is very useful in helping us determine the relationship between the two events.

PDTB-styled Discourse Relations. Another widely adopted discourse relation annotation is the PDTB framework (Prasad et al., 2008). Unlike the RST

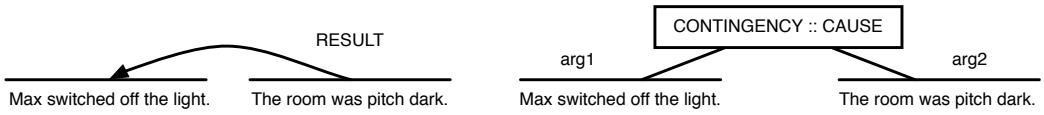


Figure 2: RST and PDTB discourse structures for the second line of text in Example 2. The structure on the left is the RST discourse structure, while the structure on the right is for PDTB.

framework, the discourse relations in PDTB build on the work on D-LTAG by Webber (2004), a lexicon-grounded approach to discourse analysis. Practically, this means that instead of starting from a pre-identified set of discourse relations, PDTB-styled annotations are more focused on detecting possible connectives (can be either explicit or implicit) within the text, before identifying the text fragments which they connect and how they are related to one another.

Applied again to the second line of text we have in Example 2, we get a structure as shown on the right side of Figure 2. From the figure we can see that the two sentences are related via a "*Cause*" relationship. Similar to what we have explained earlier for the case of RST, the presence of a causal effect here strongly hints to us that events in the two sentences share a BEFORE/AFTER relationship.

At this point we want to note the differences between the use of the RST framework and PDTB-styled discourse relations in the context of our work. The theoretical underpinnings behind these two discourse analysis are very different, and we believe that they can be complementary to each other. First, the RST framework breaks up text within an article linearly into non-overlapping EDUs. Relations can only be defined between neighboring EDUs. However this constraint is not found in PDTB-styled relations, where a text fragment can participate in one discourse relation, and a subsequence of it participate in another. PDTB relations are also not restricted only to adjacent text fragments. In this aspect, the flexibility of the PDTB relations can complement the seemingly more rigid RST framework.

Second, with PDTB-styled relations not every sentence needs to be in a relation with another as the PDTB framework does not aim to build a global discourse tree that covers all sentence pairs. This is a problem when we need to do an article-wide analysis. The RST framework does not suffer from this limitation however as we can build up a discourse tree connecting all the text within a given article.

Topical Text Segmentation. A third complementary type of inter-sentential analysis is topical text segmentation. This form of segmentation separates a piece of text into non-overlapping segments, each of which can span several sentences. Each segment represents passages or topics, and provides a coarse-grained study of the linear structure of the text (Skorochod'Ko, 1972; Hearst, 1994). The transition between segments can represent possible topic shifts which can provide useful information about temporal relationships.

Referring to Example 3[2], we have delimited the different lines of text into segments with parentheses along with a subscript. Segment (1) talks about the casualty numbers seen at a medical centre, while Segment (2) provides background information that informs us a bomb explosion had taken place. The segment boundary signals to us a possible temporal shift and can help us to infer that the bombing event took place BEFORE the deaths and injuries had occurred.

(The Davao Medical Center, a regional government hospital, recorded 19 deaths with 50 wounded. Medical evacuation workers however said the injured list was around 114, spread out at various hospitals.)$_1$
(A powerful bomb tore through a waiting shed at the Davao City international airport at about 5.15 pm (0915 GMT) while another explosion hit a bus terminal at the city.)$_2$ (3)

4 Methodology

Having motivated the use of discourse analysis for our problem, we now proceed to explain how we can make use of them for temporal classification. The different facets of discourse analysis that we are exploring in this work are structural in nature. RST

[2]From article AFP_ENG_20030304.0250 of the ACE 2005 corpus.

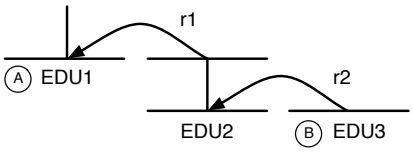


Figure 3: A possible RST discourse tree. The two circles denote two events A and B which we are interested in.

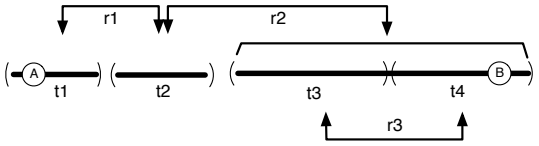


Figure 4: A possible PDTB-styled discourse annotation where the circles represent events we are interested in.

and PDTB discourse relations are commonly represented as graphs, and we can also view the output of text segmentation as a graph with individual text segments forming vertices, and the transitions between them forming edges.

Considering this, we propose the use of support vector machines (SVM), adopting a convolution kernel (Collins and Duffy, 2001) for its kernel function (Vapnik, 1999; Moschitti, 2006). The use of convolution kernels allows us to do away with the extensive feature engineering typically required to generate flat vectorized representations of features. This process is time consuming and demands specialized knowledge to achieve representations that are discriminative, yet are sufficiently generalized. Convolution kernels had also previously been shown to work well for the related problem of *E-T* temporal classification (Ng and Kan, 2012), where the features adopted are similarly structural in nature.

We now describe our use of the discourse analysis frameworks to generate appropriate representations for input to the convolution kernel.

RST Discourse Framework. Recall that the RST framework provides us with a discourse tree for an entire input article. In recent years several automatic RST discourse parsers have been made available. In our work, we first make use of the parser by Feng and Hirst (2012) to obtain a discourse tree representation of our input. To represent the meaningful portion of the resultant tree, we encode path information between the two sentences of interest.

We illustrate this procedure using the example discourse tree illustrated in Figure 3. EDUs including $EDU1$ to $EDU3$ form the vertices while discourse relations $r1$ and $r2$ between the EDUs form the edges. For a *E-E* pair, $\{A, B\}$, we can obtain a feature structure by first locating the EDUs within which A and B are found. A is found inside $EDU1$ and B is found within $EDU3$. We trace the shortest path between $EDU1$ and $EDU3$, and use this path as the feature structure for the *E-E* pair, *i.e.* $\{r1 \rightarrow r2\}$.

PDTB-styled Discourse Relations. We make use of the automatic PDTB discourse parser from Lin et al. (2013) to obtain the discourse relations over an input article. Similar to how we work with the RST discourse framework, for a given *E-E* pair, we retrieve the relevant text fragments and use the shortest path linking the two events as a feature structure for our convolution kernel classifier.

An example of a possible PDTB-styled discourse annotation is shown in Figure 4. The horizontal lines represent different sentences in an article. The parentheses delimit text fragments, $t1$ to $t4$, which have been identified as arguments participating in discourse relations, $r1$ to $r3$. For a given *E-E* pair $\{A, B\}$, we use the trace of the shortest path between them *i.e.* $\{r1 \rightarrow r2\}$ as a feature structure.

We take special care to regularize the input (as, unlike EDUs in RST, arguments to different PDTB relations may overlap, as in $r2$ and $r3$). We model each PDTB discourse annotation as a graph and employ Dijkstra's shortest path algorithm. The graph resulting from the annotation in Figure 4 is given in Figure 5. Each text fragment t_i maps to a vertex n_i in the graph. PDTB relations between text fragments form edges between corresponding vertices. As $r2$ relates $t2$ to both $t3$ and $t4$, two edges link up $n2$ to the corresponding vertices $n3$ and $n4$ respectively. By doing this, Dijkstra's algorithm will always allow us to find the desired shortest path.

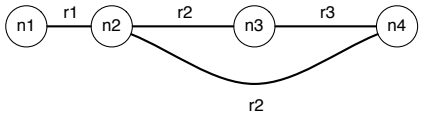


Figure 5: Graph derived from discourse annotation in Figure 4.

Topical Text Segmentation. Taking as input a complete text article, we make use of the state-of-the-art text segmentation system from Kazantseva and Szpakowicz (2011). The output of the system is a series of non-overlapping, linear text segments, which we can number sequentially.

In Figure 6 the horizontal lines represent sentences. Parentheses with subscripts mark out the segment boundaries. We can see two segments $s1$ and $s2$ here. Given a target *E-E* pair $\{A, B\}$ (represented as circles inside the figure), we identify the segment number of the corresponding segment in which each of A and B is found. We build a feature structure with the identified segment numbers, *i.e.* $\{s1 \rightarrow s2\}$ to capture the segmentation.

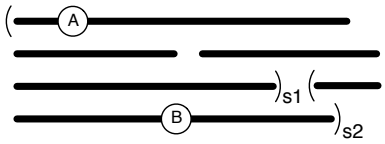


Figure 6: A possible segmentation of three sentences into two segments.

5 Results

We conduct a series of experiments to validate the utility of our proposed features.

Data Set. We make use of the same data set built by Do et al. (2012). The data set consists of 20 newswire articles which originate from the ACE 2005 corpus (ACE, 2005). Initially, the data set consist of 324 event mentions, and a total of 375 annotated *E-E* pairs. We perform the same temporal saturation step as described in Do et al. (2012), and obtained a total of 7,994 *E-E* pairs[3].

A breakdown of the number of instances by each temporal classes is shown in Table 1. Unlike earlier data sets such as that for TempEval-2 where more than half (about 55%) of test instances belong to the OVERLAP class, OVERLAP instances make up just 10% of the data set.

This difference is due mainly to the fact that our data set consists not only of intra-sentence *E-E* pairs, but also of article-wide *E-E* pairs. Figure 7 shows the number of instances for each temporal class broken down by the number of sentences (*i.e.* sentence gap) that separate the events within each *E-E* pair. We see that as the sentence gap increases, the proportion of OVERLAP instances decreases. The intuitive explanation for this is that when event mentions are very far apart in an article, it becomes more unlikely that they happen within the same time span.

Class	AFTER	BEFORE	OVERLAP
# *E-E* pairs	3,588 (45%)	3,589 (45%)	815 (10%)

Table 1: Number of *E-E* pairs in data set attributable to each temporal class. Percentages shown in parentheses.

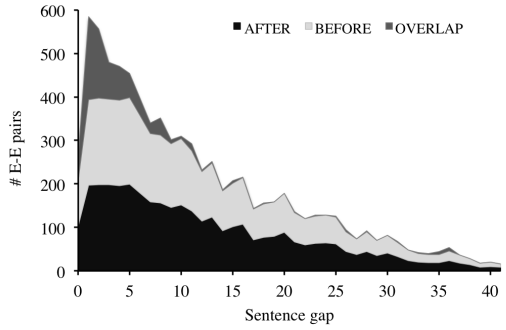


Figure 7: Breakdown of number of *E-E* pairs for each temporal class based on sentence gap.

Experiments. The work done in Do et al. (2012) is highly related to our experiments, and so we have reported the relevant results for local *E-E* classification in Row 1 of Table 2 as a reference. While largely comparable, note that a direct comparison is not possible because 1) the number of *E-E* instances we have is slightly different from what was reported, and 2) we do not have access to the exact partitions they have created for 5-fold cross-validation.

As such, we have implemented a baseline adopting similar surface lexico-syntactic features used in previous work (Mani et al., 2006; Bethard and Martin, 2007; Ng and Kan, 2012; Do et al., 2012), including 1) part-of-speech tags, 2) tenses, 3) dependency parses, 4) relative position of events in article,

[3]Though we have obtained the data set from the original authors, there was a discrepancy in the number of *E-E* pairs. The original paper reported a total of 376 annotated *E-E* pairs. Besides this, we also repeated the saturation steps iteratively until no new relationship pairs are generated. We believe this to be an enhancement as it ensures that all inferred temporal relationships are generated.

	System	Precision	Recall	F_1
(1)	DO2012	43.86	52.65	47.46
(2)	BASE	59.55	38.14	46.50
(3)	BASE + RST + PDTB + TOPICSEG	71.89	41.99	53.01
(4)	BASE + RST + PDTB + TOPICSEG + COREF	75.23	43.58	55.19
(5)	BASE + O-RST + PDTB + O-TOPICSEG + O-COREF	78.35	54.24	64.10

Table 2: Macro-averaged results obtained from our experiments. The difference in F_1 scores between each successive row is statistically significant, but a comparison is not possible between rows (1) and (2).

5) the number of sentences between the target events and 6) VerbOcean (Chklovski and Pantel, 2004) relations between events. This baseline system, and the subsequent systems we will describe, comprises of three separate one-vs-all classifiers for each of the temporal classes. The result obtained by our baseline is shown in Row 2 (*i.e.* BASE) in Table 2. We note that our baseline is competitive and performs similarly to the results obtained by Do et al. (2012). However as we do not have the raw judgements from Do's system, we cannot test for statistical significance.

We also implemented our proposed features and show the results obtained in the remaining rows of Table 2. In Row 3, RST denotes the RST discourse feature, PDTB denotes the PDTB-styled discourse features, and TOPICSEG denotes the text segmentation feature. Compared to our own baseline, there is a relative increase of 14% in F_1, which is statistically significant when verified with the one-tailed Student's paired *t*-test ($p < 0.01$).

In addition, Do et al. (2012) have shown the value of event co-reference. Therefore we have also included this feature by making use of an automatic event co-reference system by Chen et al. (2011). The result obtained after adding this feature (denoted by COREF) is shown in Row 4. The relative increase in F_1 of about 4% from Row 3 is statistically significant ($p < 0.01$) and affirms that event co-reference is a useful feature to have, together with our proposed features. We note that our complete system in Row 4 gives a 16% improvement in F_1, relative to the reference system DO2012 in Row 1.

To get a better idea of the performance we can obtain if oracular versions of our features are available, we also show the results obtained if hand-annotated RST discourse structures, text segments, as well as event co-reference information were used. Annotations for the RST discourse structures and text segments were performed by the first author (RST annotations were made following the annotation guidelines given by Carlson and Marcu (2001)). Oracular event co-reference information was included in the dataset that we have used.

In Row 5 the prefix O denotes oracular versions of the features we had proposed. From the results we see that there is a marked increase of over 15% in F_1 relative to Row 4. Compared to Do's state-of-the-art system, there is also a relative gain of at least 35%. These oracular results further confirm the importance of non-local discourse analysis for temporal processing.

6 Discussion

Ablation tests. We performed ablation tests to assess the efficacy of the discourse features used in our earlier experiments. Starting from the full system, we dropped each discourse feature in turn to see the effect this has on overall system performance. Our test is performed over the same data set, again with 5-fold cross-validation. The results in Table 3 show a statistically significant (based on the one-tailed Student's paired *t*-test) drop in F_1 in each case, which proves that each of our proposed features is useful and required.

From the ablation tests, we also observe that the RST discourse feature contributes the most to overall system performance while the PDTB discourse feature contributes the least. However we should not conclude prematurely that the former is more useful than the latter; as the results are obtained using parses from automatic systems, and are not reflective of the full utility of ground truth discourse annotations.

Useful Relations. The ablation test results showed us that discourse relations (in particular RST dis-

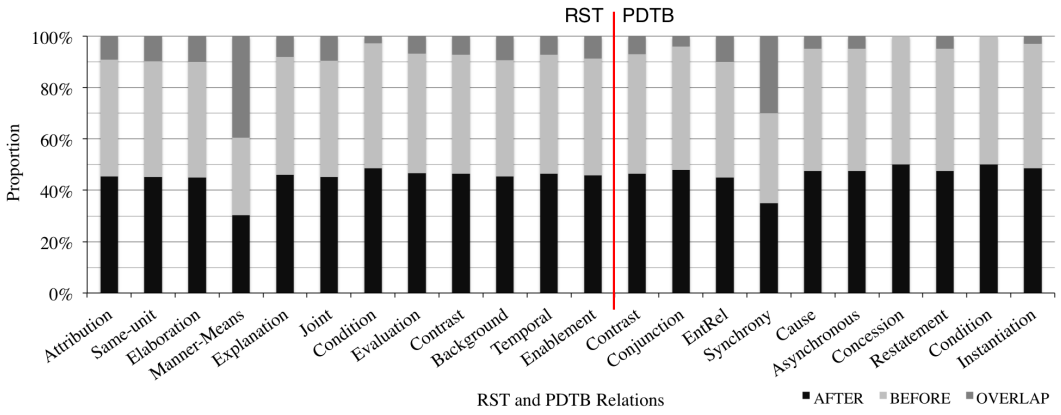


Figure 8: Proportion of occurence in temporal classes for every RST and PDTB relation.

Ablated Feature	Change in F_1	Sig
−RST	-9.03	**
−TOPICSEG	-2.98	**
−COREF	-2.18	**
−PDTB	-1.42	*

Table 3: Ablation test results. '**' and '*' denote statistically significance against the full system with $p < 0.01$ and $p < 0.05$, respectively.

course relations) are the most important in our system. We have also motivated our work earlier with the intuition that certain relations such as the RST "*Result*" and the PDTB "*Cause*" relations provide very useful temporal cues. We now offer an introspection into the use of these discourse relations.

Figure 8 illustrates the relative proportion of temporal classes in which each RST and PDTB relation appear. If the relations are randomly distributed, we should expect their distribution to follow that of the temporal classes as shown in Table 1. However we see that many of the relations do not follow this distribution. For example, we observe that several relations such as the RST "*Condition*" and PDTB "*Cause*" relations are almost exclusively found within AFTER and BEFORE event pairs only, while the RST "*Manner-means*" and PDTB "*Synchrony*" relations occur in a disproportionately large number of OVERLAP event pairs. These relations are likely useful in disambiguating between the different temporal classes.

To verify this, we examine the convolution tree fragments that lie on the support vector of our SVM classifier. The work of Pighin and Moschitti (2010) in linearizing kernel functions allows us to take a look at these tree fragments. Applying the linearization process leads to a different classifier from the one we have used. The identified tree fragments are therefore just an approximation to those actually employed by our classifier. However, this analysis still offers an introspection as to what relations are most influential for classification.

BEFORE		OVERLAP	
B1	(Temporal ...	O1	(Manner-means ...
B2	(Temporal (Elaboration ...		
B3	(Condition (Explanation ...		
B4	(Condition (Attribution ...		
B5	(Elaboration (Bckgrnd ...		

Table 4: Subset of top RST discourse fragments on support vectors identified by linearizing kernel function.

Table 4 shows a subset of the top RST discourse fragments identified for the BEFORE and OVERLAP one-vs-all classifiers. The list is in line with what we expect from Figure 8. The former consists of fragments containing relations such as "*Temporal*" and "*Condition*", while the latter has a sole fragment containing "*Manner-Means*".

To illustrate what these fragments may mean, we show several example sentences from our data set in Example 4. Sentence A consists of the tree fragment B1, *i.e.* "*(Temporal...*". Its corresponding discourse structure is illustrated in the top half of Figure 9. This fragment indicates to us (correctly) that the event "**wielded**" happened BEFORE Milosevic was "**swept out**" of power. Sentence B is made up of tree fragment O1, *i.e.* "*(Manner-means...*",

and its discourse structure is shown in the bottom half of Figure 9. As with the previous example, the fragment suggests (correctly) that there should be a OVERLAP relationship for the "**requested** – **said**" event pair.

> [A] Milosevic and his wife **wielded** enormous power in Yugoslavia for more than a decade before he was **swept out** of power after a popular revolt in October 2000.
>
> [B] The court order was **requested** by Jack Welch's attorney, Daniel K. Webb, who **said** Welch would likely be asked about his business dealings, his health and entries in his personal diary. (4)

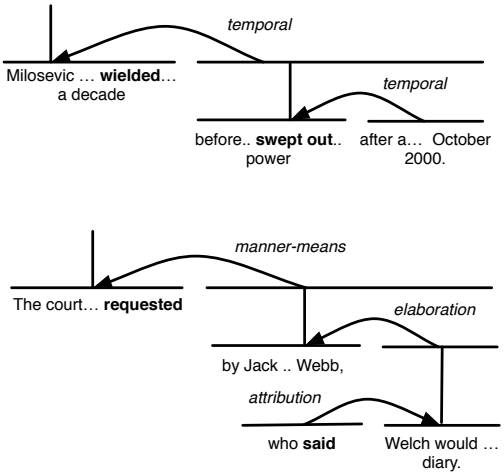


Figure 9: RST discourse structures for sentences A (top half) and B (bottom half) in Example 4.

Segment Numbers. From the ablation test results, text segmentation is the next most important feature after the RST discourse feature. This is interesting given that the defined feature structure for topical text segmentation is not the most intuitive. By using actual segment numbers, the structure may not generalize well for articles of different lengths for example, as each article may have vastly different number of segments. The transition across segments may also not carry the same semantic significance for different articles.

Our experiments have however shown that this feature design is useful in improving performance. This is likely because:

1. The default settings of the text segmentation system we had used are such that precision is favoured over recall (Kazantseva and Szpakowicz, 2011, p. 292). As such there is just an average of between two to three identified segments per article. This makes the feature more generalizable despite making use of actual segment numbers.

2. The style of writing in newswire articles which we are experimenting on generally follows common journalistic guidelines. The semantics behind the transitions across the coarse-grained segments that were identified are thus likely to be of a similar nature across many different articles.

We leave for future work an investigation into whether more fine-grained topic segments can lead to further performance gains. In particular, it will be interesting to study if work on argumentative zoning (Teufel and Kan, 2011) can be applied to newswire articles, and whether the subsequent learnt document structures can be used to delineate topic segments more accurately.

Error Analysis. Besides examining the features we had used, we also want to get a better idea of the errors made by our classifier. Recall that we are using separate one-vs-all classifiers for each of the temporal classes, so each of the three classifiers generates a column in the aggregate confusion matrix shown in Table 5. In cases where none of the SVM classifiers return a positive confidence value, we do not assign a temporal class (captured as column **N**). The high number of event pairs which are not assigned to any temporal class explains the lower recall scores obtained by our system, as observed in Table 2.

	Predicted			
	O	**B**	**A**	**N**
O	119 (14.7%)	114 (14.1%)	104 (12.8%)	474 (58.5%)
B	19 (0.5%)	2067 (57.9%)	554 (15.5%)	928 (26.0%)
A	16 (0.5%)	559 (15.7%)	2046 (57.3%)	947 (26.5%)

Table 5: Confusion matrix obtained for the full system, classifying into (**O**)VERLAP, (**B**)EFORE, (**A**)FTER, and (**N**)o result.

Additionally, an interesting observation is the low percentage of OVERLAP instances that our classifier managed to predict correctly. About 57% of BEFORE and AFTER instances are classified cor-

rectly, however only about 15% of OVERLAP instances are correct.

Figure 10 offers more evidence to suggest that our classifier works better for the BEFORE and AFTER classes than the OVERLAP class. We see that as sentence gap increases, we achieve a fairly consistent performance for both BEFORE and AFTER instances. OVERLAP instances are concentrated where the sentence gap is less than 7, with the best accuracy figure coming in below 30%.

Although not definitive, this may be because our data set consists of much fewer OVERLAP instances than the other two classes. This bias may have led to insufficient training data for accurate OVERLAP classification. It will be useful to investigate if using a more balanced data set for training can help overcome this problem.

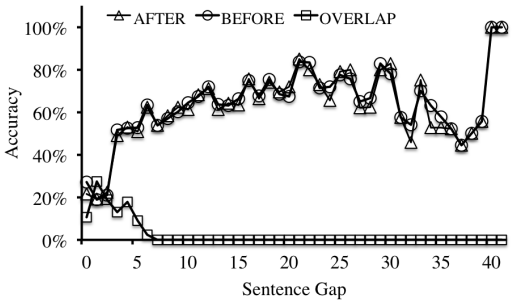


Figure 10: Accuracy of the classifer for each temporal class, plotted against the sentence gap of each *E-E* pair.

7 Conclusion

We believe that discourse features play an important role in the temporal ordering of events in text. We have proposed the use of different discourse analysis frameworks and shown that they are effective for classifying the temporal relationships of article-wide *E-E* pairs. Our proposed discourse-based features are robust and work well even though automatic discourse analysis is noisy. Experiments further show that improvements to these underlying discourse analysis systems will benefit system performance.

In future work, we will like to explore how to better exploit the various discourse analysis frameworks for temporal classification. For instance, RST relations are either *hypotactic* or *paratactic*. Marcu (1997) made use of this to generate automatic summaries by considering EDUs which are nuclei to be more salient. We believe it is interesting to examine how such information can help. We are also interested to apply discourse features in the context of a global inferencing system (Yoshikawa et al., 2009; Do et al., 2012), as we think such analyses will also benefit these systems as well.

Acknowledgments

We like to express our gratitude to Quang Xuan Do, Wei Lu, and Dan Roth for generously making available the data set they have used for their work in EMNLP 2012. We would also like to thank the anonymous reviewers who reviewed this paper for their valuable feedback.

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

ACE. 2005. The ACE 2005 (ACE05) Evaluation Plan. October.

Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2002. Inferring Strategies for Sentence Ordering in Multidocument News Summarization. *Journal of Artificial Intelligence Research (JAIR)*, 17:35–55.

Steven Bethard and James H. Martin. 2007. CU-TMP: Temporal Relation Classification Using Syntactic and Semantic Features. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval)*, pages 129–132, June.

Lynn Carlson and Daniel Marcu. 2001. Discourse tagging manual. Technical Report ISI-TR-545, Information Sciences Institute, University of Southern California, July.

Bin Chen, Jian Su, Sinno Jialin Pan, and Chew Lim Tan. 2011. A Unified Event Coreference Resolution by Integrating Multiple Resolvers. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 102–110, November.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 33–40, July.

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of NIPS*.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint Inference for Event Timeline Construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 677–689, July.

Jennifer D'Souza and Vincent Ng. 2013. Classifying Temporal Relations with Rich Linguistic Knowledge. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 918–927, June.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistics Features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 60–68, July.

Eun Young Ha, Alok Baikadi, Carlyle Licata, and James C. Lester. 2010. NCSU: Modeling Temporal Relations with Markov Logic and Lexical Ontology. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 341–344, July.

Marti A. Hearst. 1994. Multi-Paragraph Segmentation of Expository Text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9–16, June.

Anna Kazantseva and Stan Szpakowicz. 2011. Linear Text Segmentation Using Affinity Propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 284–293, July.

Anup Kumar Kolya, Asif Ekbal, and Sivaji Bandyopadhyay. 2010. JU_CSE_TEMP: A First Step Towards Evaluating Events, Time Expressions and Temporal Relations. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 345–350, July.

Alex Lascarides and Nicholas Asher. 1993. Temporal Interpretation, Discourse Relations and Commonsense Entailment. *Linguistics and Philosophy*, 16(5):437–493.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2013. A PDTB-styled End-to-End Discourse Parser. *Natural Language Engineering*, FirstView:1–34, February.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine Learning of Temporal Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 753–760, July.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 1997. From Discourse Structures to Text Summaries. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, volume 97, pages 82–88, July.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, September.

Jun-Ping Ng and Min-Yen Kan. 2012. Improved Temporal Relation Classification using Dependency Parses and Selective Crowdsourced Annotations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 2109–2124, December.

Daniele Pighin and Alessandro Moschitti. 2010. On Reverse Feature Engineering of Syntactic Tree Kernels. In *Proceedings of the 14th Conference on Natural Language Learning (CoNLL)*, August.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, May.

Andrea Setzer, Robert Gaizauskas, and Mark Hepple. 2003. Using Semantic Inferences for Temporal Annotation Comparison. In *Proceedings of the 4th International Workshop on Inference in Computational Semantics (ICoS)*, September.

Eduard F. Skorochod'Ko. 1972. Adaptive Method of Automatic Abstracting and Indexing. In *Proceedings of the IFIP Congress*, pages 1179–1182.

Carlota S. Smith. 2010. Temporal Structures in Discourse. *Text, Time, and Context*, 87:285–302.

Simone Teufel and Min-Yen Kan. 2011. Robust Argumentative Zoning for Sensemaking in Scholarly Documents. In *Advanced Language Technologies for Digital Libraries*, pages 154–170. Springer.

Naushad Uzzaman and James F. Allen. 2010. TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 276–283, July.

Naushad Uzzaman, Hector Llorens, James F. Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. TempEval-3: Evaluating Events, Time Expressions, and Temporal Relations. *Computing Research Repository (CoRR)*, abs/1206.5333.

Vladimir N. Vapnik, 1999. *The Nature of Statistical Learning Theory*, chapter 5. Springer.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The TempEval Challenge: Identifying

Temporal Relations in Text. *Language Resources and Evaluation*, 43(2):161–179.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 57–62, July.

Marc Verhagen. 2005. Temporal Closure in an Annotation Environment. *Language Resources and Evaluation*, 39(2-3):211–241.

Bonnie Webber. 2004. D-LTAG: Extending Lexicalized TAG to Discourse. *Cognitive Science*, 28(5):751–779.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly Identifying Temporal Relations with Markov Logic. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (AFNLP)*, pages 405–413, August.

Combining Generative and Discriminative Model Scores for Distant Supervision

Benjamin Roth, Dietrich Klakow
Saarland University
Spoken Language Systems
Saarbrücken, Germany
{benjamin.roth|dietrich.klakow}@lsv.uni-saarland.de

Abstract

Distant supervision is a scheme to generate noisy training data for relation extraction by aligning entities of a knowledge base with text. In this work we combine the output of a discriminative at-least-one learner with that of a generative hierarchical topic model to reduce the noise in distant supervision data. The combination significantly increases the ranking quality of extracted facts and achieves state-of-the-art extraction performance in an end-to-end setting. A simple linear interpolation of the model scores performs better than a parameter-free scheme based on non-dominated sorting.

1 Introduction

Relation extraction is the task of finding relational facts in unstructured text and putting them into a structured (tabularized) knowledge base. Training machine learning algorithms for relation extraction requires training data. If the set of relations is pre-specified, the training data needs to be labeled with those relations.

Manual annotation of training data is laborious and costly, however, the knowledge base may already partially be filled with instances from the relations. This is utilized by a scheme known as distant supervision (DS) (Mintz et al., 2009): text is automatically labeled by aligning (matching) pairs of entities that are contained in a knowledge base with their textual occurrences. Whenever such a match is encountered, the surrounding context (sentence) is assumed to express the relation.

This assumption, however, can fail. Consider the example given in (Takamatsu et al., 2012): If the tuple `place_of_birth(Michael Jackson, Gary)` is contained in the knowledge base, one matching context could be:

Michael Jackson *was born in* ***Gary*** *...*

And another possible context:

Michael Jackson *moved from* ***Gary*** *...*

Clearly, only the first context indeed expresses the relation and should be labeled accordingly.

Three basic approaches have been proposed to deal with noisy distant supervision instances: The *discriminative at-least-one* approach (Riedel et al., 2010), that requires that at least one of the matches for a relation-entity tuple indeed expresses the relation; The *generative* approach (Alfonseca et al., 2012) that separates relation-specific distributions from noise distributions by using hierarchical topic models; And the *pattern correlation* approach (Takamatsu et al., 2012) that assumes that contexts which match argument pairs have a high overlap in argument pairs with other patterns expressing the relation.

In this work we combine 1) a *discriminative at-least-one* learner, that requires high scores for both a dedicated noise label and the matched relation, and 2) a *generative topic model* that uses a feature-based representation to separate relation-specific patterns from background or pair-specific noise. We score surface patterns and show that combining the two approaches results in a better ranking quality of relational facts. In an end-to-end evaluation we set a threshold on the pattern scores and apply the pat-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 24–29,
Seattle, Washington, USA, 18-21 October 2013.

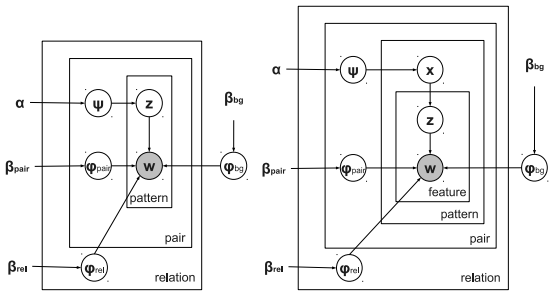


Figure 1: Hierarchical topic models. Intertext model (left) and feature model (right).

terns in a TAC KBP-style evaluation. Although the surface patterns are very simple (only strings of tokens), they achieve state-of-the-art extraction results.

2 Related Work

2.1 At-Least-One Models

The original form of distant supervision (Mintz et al., 2009) assumes all sentences containing an entity pair to be potential patterns for the relation holding between the entities. A variety of models relax this assumption and only presume that *at least one* of the entity pair occurrences is a textual manifestation of the relation. The first proposed model with an at-least-one learner is that of Riedel et al. (2010) and Yao et al. (2010). It consists of a factor graph that includes binary variables for contexts, and groups contexts together for each entity pair. MultiR (Hoffmann et al., 2011) can be viewed as a multi-label extension of (Riedel et al., 2010). A further extension is MIMLRE (Surdeanu et al., 2012), a jointly trained two-stage classification model.

2.2 Hierarchical Topic Model

The hierarchical topic model (*HierTopics*) by Alfonseca et al. (2012) models the distant supervision data by a generative model. For each corpus match of an entity pair in the knowledge base, the corresponding surface pattern is assumed to be typical for either the entity pair, the relation, or neither. This principle is then used to infer distributions over patterns of one of the following types:

1. For every entity pair, a pair-specific distribution.
2. For every relation, a relation-specific distribution.
3. A general background distribution.

The generative process assumes that for each argument pair in the knowledge base, all patterns are generated by first choosing a hidden variable z which can take on three values, B for background, R for relation and P for pair. Corresponding vocabulary distributions ($\phi_{bg}, \phi_{rel}, \phi_{pair}$) for generating the context patterns are chosen according to the value of z. The Dirichlet-smoothed vocabulary distributions are shared on the respective levels. Figure 1 shows the plate diagram of the HierTopics model.

3 Model Extensions and Combination

3.1 Generative Model

We use a feature-based extension (Roth and Klakow, 2013) of Alfonseca et al. (2012) to include bigrams for a more fine-grained representation of the patterns. For including features in the model, the model is extended with a second layer of hidden variables. A variable x represents a choice of B, R or P for every pattern, i.e. there is one variable x for every pattern. Each feature is generated conditioned on a second variable $z \in \{B, R, P\}$, i.e. there are as many variables z for a pattern as there are features for it. First, the hidden variable x is generated, then all z variables are generated for the corresponding features (see Figure 1). The values B, R or P of z depend on the corresponding x by a transition distribution:

$$P(Z_i = z | X_{j(i)} = x) = \begin{cases} p_{same}, & \text{if } z = x \\ \frac{1-p_{same}}{2}, & \text{otherwise} \end{cases}$$

where features at indices i are mapped to the corresponding pattern indices by a function $j(i)$; p_{same} is set to .99 to enforce the correspondence between pattern and feature topics. [1]

3.2 Discriminative Model

As a second feature-based model, we employ a perceptron model that enforces constraints on the labels for patterns (Roth and Klakow, 2013). The model consists of log-linear factors for the set of relations

[1] The hyper-parameters used for the feature-based topic model are $\alpha = (1, 1, 1)$ and $\beta = (.1, .001, .001)$.

Algorithm 1 At-Least-One Perceptron Training

$\theta \leftarrow 0$
for $r \in \mathcal{R}$ **do**
  for $pair \in$ kb_pairs(r) **do**
    for $s \in$ sentences$(pair)$ **do**
      for $r' \in \mathcal{R} \setminus r$ **do**
        if $P(r|s,\theta) \leq P(r'|s,\theta)$ **then**
          $\theta \leftarrow \theta + \phi(s,r) - \phi(s,r')$
        if $P(NIL|s,\theta) \leq P(r'|s,\theta)$ **then**
          $\theta \leftarrow \theta + \phi(s,NIL) - \phi(s,r')$
    if $\forall_{s \in sentences(pair)} : P(r|s,\theta) \leq P(NIL|s,\theta)$ **then**
      $s^* = \arg\max_s \frac{P(r|s,\theta)}{P(NIL|s,\theta)}$
      $\theta \leftarrow \theta + \phi(s^*,r) - \phi(s^*,NIL)$

$\mathcal{R}$ as well as a factor for the *NIL* label (no relation). Probabilities for a relation r given a sentence pattern s are calculated by normalizing over log-linear factors defined as $f_r(s) = exp\left(\sum_i \phi_i(s,r)\theta_i\right)$, with $\phi(s,r)$ the feature vector for sentence s and label assignment r, and θ_r the feature weight vector.

The learner is directed by the following semantics: First, for a sentence s that has a distant supervision match for relation r, relation r should have a higher probability than any other relation $r' \in \mathcal{R} \setminus r$. As extractions are expected to be noisy, high probabilities for NIL are enforced by a second constraint: NIL must have a higher probability than any relation $r' \in \mathcal{R} \setminus r$. Third, at least one DS sentence for an argument pair is expected to express the corresponding relation r. For sentences s for an entity pair belonging to relation r, this can be written as the following constraints:

$$\forall_{s,r'} : P(r|s) > P(r'|s) \wedge P(NIL|s) > P(r'|s)$$
$$\exists_s : P(r|s) > P(NIL|s)$$

The violation of any of the above constraints triggers a perceptron update. The basic algorithm is sketched in Algorithm 1.[2]

3.3 Model Combination

The per-pattern probabilities $P(r|pat)$ are calculated as in Alfonseca et al. (2012) and aggregated over all pattern occurrences: For the topic model, the number of times the relation-specific topic has been sampled for a pattern is divided by $n(pat)$, the number of times the same pattern has been observed. Analogously for the perceptron, the number of times a pattern co-occurs with entity pairs for r is multiplied by the perceptron score and divided by $n(pat)$.

[2]The weight vectors are averaged over 20 iterations.

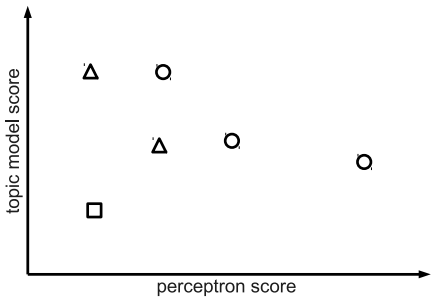


Figure 2: Score combination by non-dominated sorting: Circles indicate patterns on the Pareto-frontier, which are ranked highest. They are followed by the triangles, the square indicates the lowest ranked pattern in this example.

For the patterns of the form ***[ARG1]*** *context* ***[ARG2]***, we compute the following scores:

- **Maximum Likelihood (MLE)**: $\frac{n(pat,r)}{n(pat)}$
- **Topic Model**: $\frac{n(pat,topic(r))}{n(pat)}$
- **Perceptron**: $\frac{n(pat,r)}{n(pat)} \cdot \frac{P(r|s,\theta)}{P(r|s,\theta)+P(NIL|s,\theta)}$
- **Interpolation**: $\frac{0.5 \cdot n(pat,topic(r))}{n(pat)} + \frac{0.5 \cdot n(pat,r) \cdot P(r|s,\theta)}{n(pat) \cdot (P(r|s,\theta)+P(NIL|s,\theta))}$

The topic model and perceptron approaches are based on plausible yet fundamentally different principles of modeling noise without direct supervision. It is therefore an interesting question how complementary the models are and how much can be gained from a combination. As the two models do not use direct supervision, we also avoid tuning parameters for their combination.

We use two schemes to obtain a combined ranking from the two model scores: The first is a ranking based on non-dominated sorting by successively computing the Pareto-frontier of the 2-dimensional score vectors (Borzsony et al., 2001; Godfrey et al., 2007). The underlying principle is that all data points (patterns in our case) that are not dominated by another point[3] build the frontier and are ranked highest (see Figure 2), with ties broken by linear

[3]A data point h_1 dominates a data point h_2 if $h_1 \geq h_2$ in all metrics and $h_1 > h_2$ in at least one metric.

combination. Sorting by computing the Pareto-frontier has been applied to training machine translation systems (Duh et al., 2012) to combine the translation quality metrics BLEU, RIBES and NTER, each of which is based on different principles. In the context of machine translation it has been found to outperform a linear interpolation of the metrics and to be more stable to non-smooth metrics and non-comparable scalings. We compare non-dominated sorting with a simple linear interpolation with uniform weights.

4 Evaluation

4.1 Ranking-Based Evaluation

Evaluation is done on the ranking quality according to TAC KBP gold annotations (Ji et al., 2010) of extracted facts from all TAC KBP queries from 2009-2011 and the TAC KBP 2009-2011 corpora. First, candidate sentences are retrieved in which the query entity and a second entity with the appropriate type are contained. Candidate sentences are then used to provide answer candidates if one of the extracted patterns matches. The answer candidates are ranked according to the score of the matching pattern.

The basis for pattern extraction is the noisy DS training data of a top-3 ranked system in TAC KBP 2012 (Roth et al., 2012). The retrieval component of this system is used to obtain sentence and answer candidates (ranked according to their respective pattern scores). Evaluation results are reported as averages over per-relation results of the standard ranking metrics mean average precision (*map*), geometric map (*gmap*), precision at 5 and at 10 (*p@5*, *p@10*).

The maximum-likelihood estimator (*MLE*) baseline scores patterns by the relative frequency they occur with a certain relation. The hierarchical topic (*hier orig*) as described in Alfonseca et al. (2012) increases the scores under most metrics, however the increase is only significant for p@5 and p@10. The feature-based extension of the topic model (*hier feat*) has significantly better ranking quality. Slightly better scores are obtained by the at-least-one perceptron learner. It is interesting to see that the model combinations both by non-dominated sorting *perc+hier (pareto)* as well as uniform interpolation *perc+hier (itpl)* give a further increase in ranking quality. The simpler interpolation scheme generally works best. Figure 3 shows the Precision/Recall curves of the basic models and the linear interpolation. On the P/R curve, the linear interpolation is equal or better than the single methods on all recall levels.

method	map	gmap	p@5	p@10
MLE	.253	.142	.263	.232
hier orig	.270	.158	.353*	.297*
hier feature	.318$^{\dagger *}$	.205$^{\dagger *}$	.363*	.321*
perceptron	.330$^{\dagger *}$	.210$^{\dagger *}$	.379*	.337*
perc+hier (pareto)	.340$^{\dagger *}$	**.220**$^{\dagger *}$	.400*	.340*
perc+hier (itpl)	**.344**$^{\dagger *}$	**.220**$^{\dagger *}$	**.426**$^{\dagger *}$	**.353**$^{\dagger *}$

Table 1: Ranking quality of extracted facts. Significance (paired t-test, $p < 0.05$) w.r.t. *MLE*(*) and *hier orig*(†).

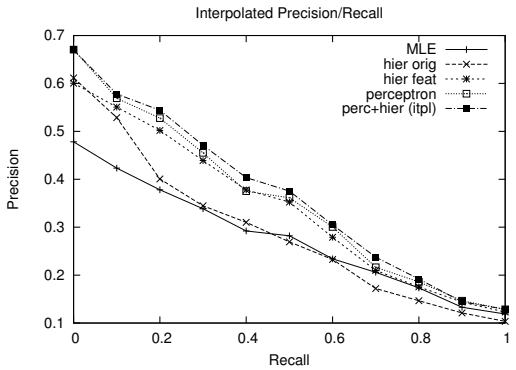


Figure 3: Precision at recall levels.

4.2 End-To-End Evaluation

We evaluate the extraction quality of the induced *perc+hier (itpl)* patterns in an end-to-end setting. We use the evaluation setting of (Surdeanu et al., 2012) and the results obtained with their pipeline for MIMLRE and their re-implementation of MultiR as a point of reference.

In Surdeanu et al. (2012) evaluation is done using a subset of queries from the TAC KBP 2010 and 2011 evaluation. The source corpus is the TAC KBP source corpus and a 2010 Wikipedia dump. Only those answers are considered in scoring that are contained in a list of possible answers from their candidates (reducing the number of gold answers from 1601 to 576 and thereby considerably increasing the value of reported recall).

For evaluating our patterns, we take the same

queries for testing as Surdeanu et al. (2012). As the document collection, we use the TAC KBP source collection and a Wikipedia dump from 07/2009 that was available to us. From this document collection, we use our retrieval pipeline of Roth et al. (2012) and take those sentences that contain query entities and slot filler candidates according to NE-tags. We filter out all candidates that are not contained in the list of candidates considered in (Surdeanu et al., 2012), and use the same reduced set of 576 gold answers as the key. We tune a single threshold parameter $t = .3$ on held-out development data and take all patterns with higher scores. Table 2 shows that results obtained with the induced patterns compare well with state-of-the-art relation extraction systems.

method	Recall	Precision	F1
MultiR	.200	.306	.242
MIMLRE	.314	.247	.277
perc+hier (itpl)	.248	.401	.307

Table 2: TAC Scores on (Surdeanu et al., 2012) queries.

4.3 Illustration: Top-Ranked Patterns

Figure 4 shows top-ranked patterns for **per:title** and **org:top_members_employees**, the two relations with most answers in the gold annotations. For maximum likelihood estimation the score is 1.0 if the patterns occurs only with the relation in question – this includes all cases where the pattern is only found once in the corpus. While this could be circumvented by frequency thresholding, we leave the long tail of the data as it is and let the algorithm deal with both frequent and infrequent patterns.

One can see that while the maximum likelihood patterns contain some reasonable relational contexts, they are less prototypical and more prone to distant supervision errors. The patterns scored high by the proposed combination generalize better, variation at the top is achieved by re-combining elements that carry relational meaning (*"is an"*, *"vice president"*, *"president director"*) or are closely correlated to the particular relation.

per:title, MLE
[ARG1] *, a singing* ***[ARG2]***
****[ARG1]*** *Best film : Capote (as* ***[ARG2]***
[ARG1] *Nunn (born October 7 , 1957 in Little Rock , Arkansas) is an American jazz* ***[ARG2]***
****[ARG2]*** *Kevin Weekes , subbing for a rarely rested* ***[ARG1]***
[ARG1] *Butterfill FRICS (born February 14 , 1941 , Surrey) is a British* ***[ARG2]***

per:title, perc+hier (itpl)
[ARG1] *, is a Canadian* ***[ARG2]***
[ARG1] *Hilligoss is an American* ***[ARG2]***
[ARG1] *, is an American film* ***[ARG2]***
[ARG1] *, is an American film and television* ***[ARG2]***
****[ARG1]*** *for Best* ***[ARG2]***

org:top_members_employees, MLE
[ARG2] *remained chairman of* ***[ARG1]***
****[ARG2]*** *asks the ball whether he and* ***[ARG1]***
[ARG2] *was chairman of the* ***[ARG1]***
****[ARG1]*** *, Joe Lieberman and* ***[ARG2]***
****[ARG1]*** *'s responsibility to pin down just how the government decided to front $ 30 billion in taxpayer dollars for the Bear Stearns deal , " Chairman* ***[ARG2]***

org:top_members_employees, perc+hier (itpl)
[ARG2] *, Vice President of the* ***[ARG1]***
[ARG1] *Vice president* ***[ARG2]***
[ARG1] *president director* ***[ARG2]***
[ARG1] *vice president director* ***[ARG2]***
[ARG1] *Board member* ***[ARG2]***

Figure 4: Top-scored patterns for maximum likelihood (MLE) and the interpolation (perc+hier itpl) method. Inexact patterns are marked by *.

5 Conclusion

We have combined two models based on distinct principles for noise reduction in distant supervision: a feature-based extension of a hierarchical topic model, and an at-least-one perceptron. Interpolation increases the quality of extractions and achieves state-of-the-art extraction performance. A combination scheme based on non-dominated sorting, that was inspired by work on combining machine translation metrics, was not as good as a simple linear combination of scores. We think that the good results motivate research into more integrated combinations of noise reduction approaches.

Acknowledgment

Benjamin Roth is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellowship.

References

Enrique Alfonseca, Katja Filippova, Jean-Yves Delort, and Guillermo Garrido. 2012. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 54–59. Association for Computational Linguistics.

S Borzsony, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430. IEEE.

Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada, and Masaaki Nagata. 2012. Learning to translate with multiple objectives. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1–10. Association for Computational Linguistics.

Parke Godfrey, Ryan Shipley, and Jarek Gryz. 2007. Algorithms and analyses for maximal vector computation. *The VLDB JournalThe International Journal on Very Large Data Bases*, 16(1):5–28.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 541–550.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Benjamin Roth and Dietrich Klakow. 2013. Feature-based models for improving the quality of noisy training data for relation extraction. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM)*. ACM.

Benjamin Roth, Grzegorz Chrupala, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2012. Generalizing from freebase and patterns using distant supervision for slot filling. In *Proceedings of the Text Analysis Conference (TAC)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 721–729, Stroudsburg, PA, USA. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023. Association for Computational Linguistics.

Exploring the utility of joint morphological and syntactic learning from child-directed speech

Stella Frank
sfrank@inf.ed.ac.uk

Frank Keller
keller@inf.ed.ac.uk

Sharon Goldwater
sgwater@inf.ed.ac.uk

ILCC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

Abstract

Children learn various levels of linguistic structure concurrently, yet most existing models of language acquisition deal with only a single level of structure, implicitly assuming a sequential learning process. Developing models that learn multiple levels simultaneously can provide important insights into how these levels might interact synergistically during learning. Here, we present a model that jointly induces syntactic categories and morphological segmentations by combining two well-known models for the individual tasks. We test on child-directed utterances in English and Spanish and compare to single-task baselines. In the morphologically poorer language (English), the model improves morphological segmentation, while in the morphologically richer language (Spanish), it leads to better syntactic categorization. These results provide further evidence that joint learning is useful, but also suggest that the benefits may be different for typologically different languages.

1 Introduction

Models of language acquisition seek to infer linguistic structure from data with minimal amounts of prior knowledge, in order to discover which characteristics of the input data are useful for learning, and thus potentially utilised by human learners. Most previous work has focused on learning individual aspects of linguistic structure. However, children clearly learn multiple aspects in parallel, rather than sequentially, implying that models of language acquisition should also incorporate joint learning. Joint models investigate the interaction between different levels of linguistic structure during learning. These interactions are often (but not necessarily) synergistic, enabling better, more robust, learning by making use of cues from multiple sources. Recent models using joint learning to model language acquisition have spanned various domains including phonology, word segmentation, syntax and semantics (Feldman et al., 2009; Elsner et al., 2012; Doyle and Levy, 2013; Johnson, 2008; Kwiatkowski et al., 2012).

In this paper we examine the joint learning of syntactic categories and morphology, which are acquired by children at roughly the same age (Clark, 2003b), implying possible interactions in the learning process. Both morphology and word order depend on categorising words based on their morpho-syntactic function. However, previous models of syntactic category learning have relied principally on surrounding context, i.e., word order constraints, whereas models of morphology use word-internal cues. Our joint model integrates both sources of information, allowing the model to flexibly weigh them according to their utility.

Languages differ in the richness of their morphology and strictness of word order. These characteristics appear to be (anti)correlated, with rich morphology co-occurring with free word order and vice versa (Blake, 2001; McFadden, 2003). The timecourse of acquisition is also influenced by language typology: learners of morphologically rich languages become productive in morphology earlier (Xanthos et al., 2011), suggesting that richer morphology may be more salient for learners than impoverished morphology. Sentence comprehension in children also shows cross-linguistic differences in the cues used to make sense of non-canonical sentence structure: learners of a morphologically rich language (Turkish) disregard word order in

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 30–41,
Seattle, Washington, USA, 18-21 October 2013.

favour of morphology, whereas learners of English favour word order (Slobin, 1982; MacWhinney et al., 1984). These interactions between morphology and word order suggest that a joint model will be better able to support the differences in cue strength (rich morphology versus strict word order), and thus be more language-general, than single-task models.

Both syntactic category and morphology induction have been the focus of much recent work. (See Hammarström and Borin (2011) for an overview of unsupervised morphology learning, likewise Christodoulopoulos et al. (2010) for a comparison of part of speech/syntactic category induction systems.) However, given the tightly coupled nature of these two tasks, there has been surprisingly little work in *joint* learning of morphology and syntactic categories. Systems for inducing syntactic categories often make use of morpheme-like features, such as word-final characters (Smith and Eisner, 2005; Haghighi and Klein, 2006; Berg-Kirkpatrick et al., 2010; Lee et al., 2010), or model words at the character-level (Clark, 2003a; Blunsom and Cohn, 2011), but do not include morphemes explicitly. Other systems (Dasgupta and Ng, 2007; Christodoulopoulos et al., 2011) use morphological segmentations learned by a separate morphology model as features in a pipeline approach.

Models of morphology induction generally operate over a lexicon, i.e. a list of word types, rather than token corpora (Goldsmith, 2006; Creutz and Lagus, 2007; Kurimo et al., 2010). These models find morphological categories on the basis of word-internal features, without taking syntactic context into account (which is of course not available in a lexicon).

Lee et al. (2011) and Sirts and Alumäe (2012) present models that infer morphological segmentations and syntactic categories jointly, although Lee et al. (2011) do not evaluate the inferred syntactic categories. Both make use of a word-type constraint which limits each word form to a single analysis (i.e., all instances of *ducks* are assigned to a single category and will have the same morpheme analysis, ignoring the gold standard distinction between a plural noun and third person singular verb). This can make inference more tractable, and often increases performance, but does not respect the ambiguity inherent in natural language, both over syntactic categories and morphological analyses. The degree of ambiguity is language dependent, so that even if a type-constraint is perhaps relatively unproblematic in English, it will pose problems in morphologically richer languages. Furthermore, these two models make use of an array of heuristics that may not allow them to be easily generalisable across languages and datasets (e.g., likelihood scaling (Sirts and Alumäe, 2012), sequential suffix matching (Lee et al., 2011)).

In this paper, we present a joint model composed of two well-known individual models. This allows us to cleanly investigate the effects of joint learning and its potential benefits over the single task models. The simplicity of our models also allows us to avoid modelling and inference heuristics.

Previous models have used adult-directed written texts, which differs significantly from the type of language available to child learners. We test our joint model on child-directed utterances in English (a morphologically poor language) and Spanish (with richer morphology)[1]. Our results indicate that our joint model is able to flexibly accommodate languages with differing levels of morphological richness. The joint model matches the performance of single task models on both tasks, demonstrating that the additional complexity is not a problem (i.e., it does not add noise). Moreover, the joint model improves performance significantly on the task corresponding to the language's weaker cue, indicating a transfer of information from the stronger cue. The fact that the nature of this improvement varies by language provides evidence that joint learning can effectively accommodate typological diversity.

2 Model

The task is to assign word tokens to part of speech categories and simultaneously segment the tokens into morphemes. We assume a relatively simple yet commonly used concatenative morphology which models a word as a stem plus (possibly null) suffix[2].

[1]There are languages with much richer morphology than Spanish, but none with a child-directed corpus suitably annotated for evaluation.

[2]Fullwood and O'Donnell (2013) recently presented a model of non-concatenative morphology that could be integrated into this model; however, it does not perform well on English (and presumably other mostly concatenative languages).

Since this is an unsupervised model, the inferred categories and morphemes lack meaningful labels, but ideally will correspond to gold standard categories and morphemes.

2.1 Word Order

We model a sequence of words as a Hidden Markov Model (HMM) with a non-parametric emission distribution. As usual, the latent states of the HMM represent syntactic categories. The tag sequence is generated by a trigram Dirichlet-multinomial distribution, where transition parameters τ are drawn from a symmetric Dirichlet distribution with the hyperparameter α_t. Each tag t_i in the sequence is then drawn from the transition distribution conditioned on the previous two tags:

$$\begin{aligned} \tau_{(t,t')} &\sim \text{Dir}(\alpha_t) \\ t_i = t | t_{i-1} = t', t_{i-2} = t'', \tau &\sim \text{Mult}(\tau_{(t',t'')}) \end{aligned}$$

This model is token-based, permitting different tokens of the same word type to have different syntactic categories. Most recent models have included a constraint forcing all tokens of a given type into the same category, which improves performance but often complicates inference. The Bayesian HMM's performance is therefore not state-of-the-art, but is comparable to other token-based models (Christodoulopoulos et al., 2010) and the model is easy to extend within the Bayesian framework, allowing us to compare multiple versions.

This part of the model is parametric, operating over a fixed number of tags T, and is identical to the formulation of tag transitions in the Bayesian HMM (Goldwater and Griffiths, 2007). However, we replace the BHMM's emission distribution with the morphologically-informed distributions described below. As in the BHMM, the emission distributions are conditioned on the tag, i.e., each tag has its own morphology.

2.2 Morphology

The morphology model introduced by Goldwater et al. (2006) generates morphological analyses for a set of tokens. These analyses consist of a tag plus a stem and suffix pair, which are concatenated to form the observed words. Both stem s and suffix f are generated from Dirichlet-multinomials conditioned on the tag t:

$$\begin{aligned} \kappa &\sim \text{Dir}(\alpha_\kappa) \\ t | \kappa &\sim \text{Mult}(\kappa) \\ \sigma &\sim \text{Dir}(\alpha_s) \\ s | t, \sigma &\sim \text{Mult}(\sigma_t) \\ \phi &\sim \text{Dir}(\alpha_f) \\ f | t, \phi &\sim \text{Mult}(\phi_t) \end{aligned}$$

The αs are hyperparameters governing the Dirichlet distributions from which the multinomials κ, σ, ϕ are drawn. In turn, t, s, and f are drawn from these multinomials.

The probability of a word under this model is the sum of the probabilities of all possible analyses $l = (t, s, f)$:

$$P_0(w) = \sum_l P_0(l) = \sum_{\substack{t,s,f \text{ s.t.} \\ s \oplus f = w}} P(s|t) P(f|t) P(t) \quad (1)$$

where $s \oplus f = w$ denotes that the concatenation of stem and suffix results in the word w.

On its own, this distribution over morphological analyses makes independence assumptions that are too strong: most word tokens of a word type have the same analysis, but P_0 will re-generate that analysis for every token. To resolve this problem, a Pitman-Yor process (PYP) is placed over the generating distribution above. The Pitman-Yor process has been found to be useful for representing the power-law distributions common in natural language (Teh, 2006; Goldwater and Griffiths, 2007; Blunsom and Cohn, 2011).

The distribution of draws from a Pitman-Yor process (which, in our case, determines the distribution of word tokens with each morphological analysis) is commonly described using the metaphor of a Chinese restaurant. A series of customers (tokens $\boldsymbol{z} = z_1 \ldots z_N$) enter a restaurant with an infinite number of initially empty tables. Upon entering, each customer is seated at a table k with probability

$$p(z_i = k | z_1 \ldots z_{i-1}, a, b) = \quad (2)$$
$$\begin{cases} \frac{n_k - a}{i - 1 + b} & \text{if } 1 \le k \le K \\ \frac{Ka + b}{i - 1 + b} & \text{if } k = K + 1 \end{cases}$$

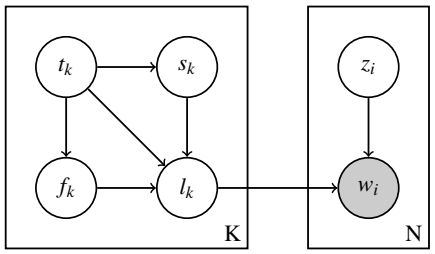


Figure 1: Plate diagram depicting the morphology model (adapted from Goldwater et al. (2006)). Hyperparameters have been omitted for clarity. The left-hand plate depicts the base distribution P_0; note that the morphological analyses l_k are generated deterministically as (t_k, s_k, f_k). The observed words w_i are also deterministic given $z_i = k$ and l_k, since $w_i = s_k \oplus f_k$.

where n_k is the number of customers already sitting at table k, K is the total number of tables occupied by the $i-1$ previous customers, and $0 \leq a < 1$ and $b \geq 0$ are hyperparameters of the process. The probability of being seated at a table increases with the number of customers already seated at that table, creating a 'rich-get-richer' power-law distribution of tokens to tables; a and b control the amount of reuse of existing tables, with smaller values leading to more reuse.

Crucially, each table serves a dish generated by the base distribution P_0—i.e., the dish is a morphological analysis $l_k = (t, s, f)$—and all the customers seated at the same table share the same dish, which is generated only once (at the point when that table is first occupied). The model can thus reuse the analysis for a particular word and avoid regenerating the same analysis multiple times. Note that multiple tables may have identical analyses, $l_k = l_{k'}$. Figure 1 illustrates how the full PYP morphology model generates the observed sequence of word tokens.

2.3 Combined Model

The full model (Figure 2) combines the latent tag sequence with the morphology model. Tag tokens are generated conditioned on local context, not the base distribution, as in the morphology model. Instead of a single PYP generating morphological analyses for all tokens, as in the Goldwater et al. (2006) model, we have a separate PYP for each tag type, i.e., each tag has its own restaurant with its own customers (the tokens labeled with that tag) and its own morphological analyses. The distribution of customers

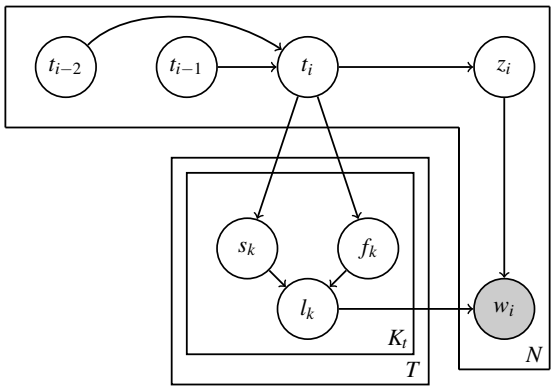


Figure 2: Plate diagram depicting the joint model. Hyperparameters have been omitted for clarity. The L-shaped plate contains the tokens, while the square plates contain the morphological analyses. The t are latent tags, z_i is an assignment to a morphological analysis $l_k = (s_k, f_k)$, and w_i is the observed word. T is the number of distinct tags, and K_t the number of tables used by tag type t.

in each of the tag-specific restaurants is still determined by Equation 2, except that all of the counts and indices are with respect to only the tokens and tables assigned to that tag.

Each tag-specific PYP (restaurant) also has a separate base distribution, $P_0^{(t)}$, resulting in distinct distributions over stems and suffixes for each tag. The analyses generated by the base distributions consist of (stem, suffix) pairs; the tag is given by the identity of the generating PYP.

$$P_0^{(t)}(w) = \sum_l P_0^{(t)}(l = (s, f)) = \sum_{\substack{s,f \text{ s.t.} \\ s \oplus f = w}} P(s|t)P(f|t) \tag{3}$$

The full joint posterior distribution of a sequence of words, tags, and morpheme analyses is shown in Figure 3. Note that all tag-specific morphology models share the same Pitman-Yor parameters a and b.

3 Inference

We use Gibbs sampling for inference over the three sets of discrete variables: tags $\boldsymbol{t}$, their assignments to morphological analyses (tables) $\boldsymbol{z}$, and the analyses themselves $\boldsymbol{l}$.

Each iteration of the sampler has two stages: First the morphological analyses $\boldsymbol{l}$ are sampled, and then each token samples a new tag and a new assignment to an analysis/table. Because the table assignments

$$P(\boldsymbol{t},\boldsymbol{l},\boldsymbol{z}|\alpha_t,a,b,\alpha_s,\alpha_f) = P(\boldsymbol{t}|\alpha_t)P(\boldsymbol{l}|\boldsymbol{t},\alpha_s,\alpha_f)P(\boldsymbol{z}|a,b) \tag{4}$$

$$P(\boldsymbol{t}|\alpha_t) = \prod_{i=2}^{N} P(t_i|t_{i-1},t_{i-2},\boldsymbol{t}_{1\ldots i-1},\alpha_t) = \prod_{t,t'=1}^{T} \frac{\Gamma(T\alpha_t)}{\Gamma(n_{tt'}+T\alpha_t)} \prod_{t''=1}^{T} \frac{\Gamma(n_{tt't''}+\alpha_t)}{\Gamma(\alpha_t)} \tag{5}$$

$$P(\boldsymbol{l}|\boldsymbol{t},\alpha_s,\alpha_f) = \prod_{t=1}^{T}\prod_{k=1}^{K_t} P_t(l_k=(s,f)|\boldsymbol{l}_{1\ldots k-1},\alpha_s,\alpha_f) \tag{6}$$

$$= \prod_{t=1}^{T} \frac{\Gamma(S\alpha_s)}{\Gamma(m_t+S\alpha_s)} \frac{\Gamma(F\alpha_f)}{\Gamma(m_t+F\alpha_f)} \prod_{s=1}^{S} \frac{\Gamma(m_{ts}+\alpha_s)}{\Gamma(\alpha_s)} \prod_{f=1}^{F} \frac{\Gamma(m_{tf}+\alpha_f)}{\Gamma(\alpha_f)} \tag{7}$$

$$P(\boldsymbol{z}|a,b) = \prod_{t=1}^{T}\prod_{i=1}^{N_t} P(z_i|t,\boldsymbol{z}_{1\ldots i-1},a,b) \tag{8}$$

$$= \prod_{t=1}^{T} \frac{\Gamma(1+b)}{\Gamma(n_t+b)} \prod_{k=1}^{K_t} (ka+b) \frac{\Gamma(n_k-a)}{\Gamma(1-a)} \tag{9}$$

Figure 3: The posterior distribution of our joint model. Because the sequence of words $\boldsymbol{w}$ is deterministic given analyses $\boldsymbol{l}$ and assignments to analyses (tables) $\boldsymbol{z}$, the joint posterior over all variables $P(\boldsymbol{w},\boldsymbol{t},\boldsymbol{l},\boldsymbol{z}|\alpha_t,a,b,\alpha_s,\alpha_f)$ is equal to $P(\boldsymbol{t},\boldsymbol{l},\boldsymbol{z}|\alpha_t,a,b,\alpha_s,\alpha_f)$ when $l_{z_i}=w_i$ for all i, and 0 otherwise. We give equations for the non-zero case. ns refer to token counts, ms to table counts. We add two dummy tokens at the start, end, and between sentences to pad the context history.

are conditioned on tags (i.e., a token must be assigned to a table in the correct PYP restaurant) resampling the tag requires immediate resampling of the table assignment as well.

3.1 Initialization

The tags are initialized uniformly at random. For each token, a segmentation point is chosen uniformly at random (we disallow segmentations with a null stem). If this segmentation is new within the PYP associated with that token's tag, a new table is created for the token in that PYP. If it matches an existing analysis, z_i is sampled from the existing tables $\boldsymbol{k}$ plus a possible new table k'.

3.2 Morphological Analyses

Each l_k represents the morphological analysis for the set of tokens assigned to table k. Resampling the segmentation point (stem and suffix identity) of the analysis changes the segmentation of all of the word tokens assigned to that analysis. Note that the tag is not included in l_k in the combined model, because the tag identity is dependent on the local contexts of all the tokens seated at the table.

Analyses are sampled from a product of Dirichlet-multinomial posteriors as follows:

$$p(l_k=(s,f)|t,\boldsymbol{l}^{\setminus k}) = \frac{m_s^{\setminus k}+\alpha_s}{m^{\setminus k}+S\alpha_s} \frac{m_f^{\setminus k}+\alpha_f}{m^{\setminus k}+F\alpha_f} \tag{10}$$

where m_s and m_f are the number of analyses for this tag that share a stem or suffix with l_k, and m is the total number of analyses for this tag. S and F are the total number of stems and suffixes in the model. $\boldsymbol{l}^{\setminus k}$ indicates that the current analysis l_k has been removed from the distribution and the appropriate counts, to create the correct conditioning distribution for the Gibbs sampler.

3.3 Tags

Tags are sampled from the product of posteriors of the transition and emission distributions. The transition distribution is a standard Dirichlet-multinomial posterior. Calculating the emission distribution probability, i.e. the marginal probability of the word given the tag, involves summing over the probability of all the existing tables in the given PYP that emit the correct word, plus the probability of a new table being created, which also includes the probability of a new analysis from $P_0^{(t)}$.

More precisely, tags are sampled from the following distribution:

$$
\begin{aligned}
& p(t_i = t | w_i = w, \boldsymbol{t}^{\setminus i}, \boldsymbol{z}^{\setminus i}, \boldsymbol{l}, \alpha_t, a, b) \qquad (11) \\
& \propto p(t_i = t | t_{i-1}, t_{i-2}, \boldsymbol{t}^{\setminus i}, \alpha_t) \times p(w | t, \boldsymbol{z}^{\setminus i}, \boldsymbol{l}) \\
& = p(t_i = t | t_{i-1}, t_{i-2}, \boldsymbol{t}^{\setminus i}, \alpha_t) \\
& \times \Big(\sum_{k \text{ s.t. } l_k = w} p(z_i = k | t, w, \boldsymbol{z}^{\setminus i}) + p(z_i = k_{\text{new}} | t, w, \boldsymbol{z}^{\setminus i}) \Big) \\
& = \frac{n_{t_{i-2} t_{i-1} t} + \alpha_t}{n_{t_{i-2} t_{i-1}} + T \alpha_t} \\
& \times \Big(\sum_{k \text{ s.t. } l_k = w} \frac{n_k - a}{n_t + b} + \frac{K_t a + b}{n_t + b} P_0^{(t)}(w) \Big)
\end{aligned}
$$

where $l_k = w$ matches tables compatible with w, i.e., the concatenation of stem and suffix form the word, $s_{l_k} \oplus f_{l_k} = w$. n_k is the number of words assigned to the table k and K_t is the total number of tables in the PYP for tag t. Note that all counts are obtained after the removal of the current t_i and z_i, i.e., from $\boldsymbol{t}^{\setminus i}$ and $\boldsymbol{z}^{\setminus i}$.

3.4 Table Assignments

Once a new tag has been sampled for a token, the table assignment must be resampled conditioned on the new tag. The assignment z_i is drawn over all compatible tables in the tag's PYP (that is, where $l_k = w$), plus a possible new table:

$$
p(z_i = k | t_i = t, \boldsymbol{w}, \boldsymbol{z}^{\setminus i}, a, b) \propto \qquad (12)
$$
$$
\begin{cases}
\frac{n_k - a}{n_t + b} & \text{if } 1 \leq k \leq K_t \\
\frac{K_t a + b}{n_t + b} \, P_0^{(t)}(w) & \text{if } k = K_t + 1
\end{cases}
$$

$P_0^{(t)}$ is calculated by summing over the probability of all possible segmentations for a new analysis for word w_i, using Equation 3. If a new table is drawn ($k > K_t$) then we also sample a new analysis for that table from $P_0^{(t)}$.

4 Preliminary Experiments

An important argument for joint learning is that it affords increased flexibility and robustness across a wider range of input data. A model that relies on word order cannot learn syntactic categories from a morphologically complex language with free word order; likewise a model attempting to categorise words using morphology alone will fail on a language without morphology. An effective joint model will be able to make use of the different cues in both language types in a flexible way.

Language A
abdc fefh pomo rtut usst
cdcc bcba gghh npop npoo
cdca aaaa fefh hfeg pnon
Language B
noom.no usrs.st bbdb.ac cbab.cc cdaa.cc
rttt.uu cbab.aa mnom.oo ccda.bc onmm.om
rruu.ts npop.mm gehg.fh trrt.uu tssu.uu

Table 1: Example sentences in the synthetic languages. Words in Category 1 are made of characters `a-d`, Category 2 `e-h`, Category 3 `m-p`, Category 4 `r-u`. Suffixes in Language B are separated with periods (.) for illustrative purposes only.

In order to test the proposed model, we run two experiments on synthetic languages, which simulate languages in which either word order or morphology is the sole cue. Most natural languages fall between these extremes, but these experiments show that our model can capture the full spectrum.

Language A is a strict word order language lacking morphology. It has a vocabulary of 200 word types, split into four different categories. The 50 word types in each category are created by combining four letters, with replacement, into four-letter words, with a different set of letters used in each category[3]. Words within a category may thus share beginning or ending characters, which could be posited as stems or suffixes by the model, but since only 50 of 256 possible strings are used, there will be no strong evidence for consistent stem and suffixes (i.e. stems appearing with multiple suffixes and vice versa). Each sentence in Language A consists of five words in one of twenty possible category sequences. In these sequences, each category is either followed by itself or the next category (i.e. [2,2,2,3,4] is valid but [2,4,3,1,4] is not). Word order is thus strongly constrained by category membership.

Language B has free word order, with category membership signalled by suffixes. Words are cre-

[3]We achieved the same results with a language using the same four characters in all categories, but using different characters makes the categories human-readable. The model does not have a orthographic/phonological component and so will not recognise the within-category similarity, other than possibly positing spurious stems or suffixes.

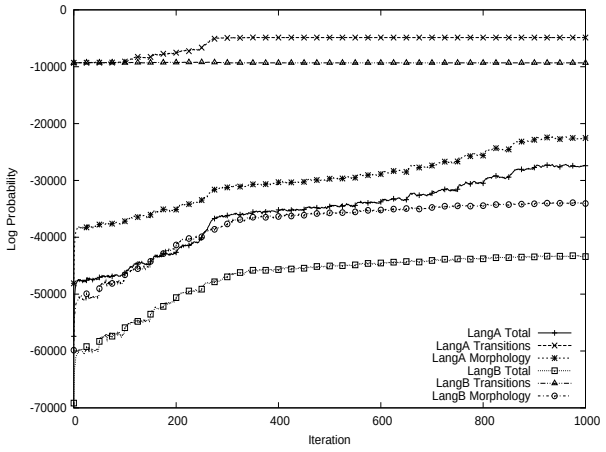


Figure 4: Log probability of the sampler state over 1000 iterations on Languages A and B.

ated by the concatenation of a stem and a suffix, where the stems are the same as the words in language A (50 stems in each of four categories). One of six category-specific suffixes is appended to each stem, resulting in 300 word types per category. Each suffix is two letters long, created by combining three possible letters (the same letters used to create the stems), thus making mis-segmentation possible (for instance, up to three of the suffixes could have the same final letter). Sentences are again five words long, but the sequence of categories is drawn at random, resulting in uniformly random word order. See Table 1 for example sentences in both languages.

We create a 5000 word corpus for each language, and run our model on these corpora. Hyperparameters are set to the same values in both languages[4].

We run the sampler on each dataset for 1000 iterations with simulated annealing. In both cases, the correct solution is found by iteration 500. Figure 4 shows that the morphology component continues to increase the log probability by increasing the number of tokens seated at a table. Note that the correct solution in Language A involves learning a very peaked transition distribution as well as an even more extreme distribution over suffixes (where only the null suffix has high probability), whereas the same distributions in Language B are much flatter. The fact that the same hyperparameter setting is able to correctly identify the two language extremes indicates that the model is robust to hyperparameter values.

These experiments demonstrate that our joint model is able to learn correctly even when only either morphology or word order is informative in a language. We now turn to acquisition data from natural languages in which both morphology and word order are useful cues but to varying degrees.

5 CDS Experiments

5.1 Data

We use two corpora, Eve (Brown, 1973) and Ornat (Ornat, 1994), from the CHILDES database (MacWhinney, 2000). These corpora consist of the child-directed utterances heard by two children, the former learning English and the latter Spanish. These have been annotated for part of speech categories and morphemes.

The CHILDES corpora are tagged with a very rich set of part of speech tags (74 tags), which we collapse to a smaller set of tags[5]. The Eve corpus has 61224 tokens and is thus larger than the Spanish corpus, which has 40497 tokens. However, the English corpus has only 17 gold suffix types, while Spanish has 83. The increased richness of Spanish morphology also has an effect on the number of word types in the corpus: the Spanish dataset has 3046 word types, whereas the larger English dataset has only 1957.

Morphology is annotated using a stem-affix encoding which does not directly correspond to our segmentation-based model. The word *running* is annotated as *run*-ING, *jumping* as *jump*-ING; the annotation is thus agnostic about ortho-morphemic segmentation (i.e., whether to segment as *run.ning* or *runn.ing*), whereas the model is forced to choose a segmentation point. Syncretic suffixes (sharing an identical surface form) are disambiguated: *sings* is annotated as *sing*-3S, *plums* as *plum*-PL. Conversely, the annotation scheme merges allomorphs into a single suffix: infinitive verbs in Spanish, for instance, are encoded as ending with -INF, corresponding to *-ar*, *-er*, and *-ir* surface forms.

[4]The PYP parameters are set to $a = 0.1, b = 1.0$ and the HMM transition parameter $\alpha_t = 1.0$; the parameters in the base distribution are $\alpha_s, \alpha_f = 0.001, \alpha_k = 0.5$.

[5]These are 13 for English (ADJ, ADV, AUX, CONJ, DET, INF, NOUN, NEG, OTH, PART, PREP, PRO, VERB) and 10 for Spanish, since the gold standard does not distinguish AUX, PART or INF.

We ignore irregular/non-affixing forms annotated with *&* (e.g. *was*, annotated as *be&PAST*) and use only hyphen-separated suffixes to evaluate. Where multiple suffixes are concatenated together (e.g., *dog*-DIM-PL) we treat this as a single suffix (-DIM-PL) for evaluation purposes.

In Spanish, many words are annotated as having a suffix of effectively zero length, e.g. the imperative *gusta* is annotated as *gusta*-2S&IMP. We replace these suffixes (where the stem is equal to the word) with a null suffix, excluding them from evaluation, as they are impossible for a segmentation-based model to find.

5.2 Evaluation

Tags are evaluated using VM (Rosenberg and Hirschberg, 2007), as has become standard for this task (Christodoulopoulos et al., 2010). VM is a measure of the normalised cross-entropy between gold and proposed clusters; it ranges between 0 and 100, with higher scores being better.

We also use VM to evaluate the morphological segmentation: all tokens with a common suffix are clustered together, and these clusters are compared against the gold suffix clusters[6]. Using a clustering metric avoids the need to evaluate against a gold segmentation point (which the annotation lacks). Tag membership is added to the non-null model suffixes, so that a final *-s* suffix found in tag 2 is distinguished from the same suffix found in tag 8 (creating suffixes -s-T8 and -s-T2), analogous to the gold annotation distinction between syncretic morphemes -PL and -3S.

Note that ceiling performance of our model on Suffix VM will be below 100, since our model cannot cluster allomorphs, which are represented by a single abstract morpheme in the gold standard.

5.3 Baselines

We test the full model, MORTAG, against a number of variations to investigate the advantages of jointly modelling the two tasks.

Two variants remove the transition distributions, and thus local syntactic context, from the model. MORTAGNOTRANS is the full model without transitions between tag tokens; morphology PYP draws remain conditioned on token tags. We add a Dirichlet prior over tags ($\alpha_t = 0.1$) to encourage tag sparsity (analogous to the transition distribution in the full model). MORCLUSTERS is the original model of Goldwater et al. (2006), in which tags (called clusters in the original) are drawn by P_0.

MORTAGNOSEG is a variant in which the only available suffix is the null suffix; thus segmentations are trivial and only tags are inferred. This model is approximately equivalent to a simple Bayesian HMM but with the addition of PYPs within the emission distribution. We also evaluate against tags found by the BHMM, with a Dirichlet-multinomial emission distribution and no morphology.

MORTAGTRUETAGS is the full model but with all tags fixed to their gold values. This model gives us oracle-type results for morphology. (Due to the annotation scheme used in CHILDES, oracle morphological segmentations are unavailable, so we were unable to test a model with gold morphology and inferred tags.)

5.4 Experimental Procedure

Hyperparameter values for the Pitman-Yor process were found using grid search on a development set (Section 10 of Eve and Section 8 of Ornat; these sections are removed from the dataset we report results on). We use the values which give the best Suffix VM performance on the development data; however we stress that the development results did not vary greatly over a wide range of hyperparameter values, and only deteriorated significantly at extreme values of a.

There are a number of other hyperparameters in the model which we set to fixed values. The transition hyperparameter α_t is set to 0.1 in all models. We set the hyperparameters for the stem and suffix distributions in the morphology base distribution P_0 to 0.001 for both α_s and α_f; α_k over tags in the MORCLUSTERS model is set to 0.5. The number of possible stems and suffixes is given by the dataset: in the Eve dataset there are 5339 candidate stems and 6617 candidate suffixes; in the Ornat dataset these numbers are 8649 and 6598, respectively. The number of tags available to the model is set to the number of gold tags in the data.

[6] We also evaluated stem morpheme clusters and found near-ceiling performance due to the high number of null-suffix words in both corpora.

	Tag VM	Suffix VM
MORTAG	59.1(1.9)	41.9(10.0)
MORCLUSTERS	22.4(1.0)*	28.0(11.9)*
MORTAGNOTRANS	19.3(1.2)*	24.4(5.2)*
MORTAGNOSEG	59.4(1.7)	–
BHMM	56.2(2.3)*	–
MORTAGTRUETAGS	–	42.5(5.2)

Table 2: English Eve corpus results. Standard deviations are in parentheses; $*$ denotes a significant difference from the MORTAG model.

	Tag VM	Suffix VM
MORTAG	43.4(2.6)	41.4(2.5)
MORCLUSTERS	20.3(2.5)*	46.5(3.2)
MORTAGNOTRANS	14.4(1.7)*	36.4(2.0)*
MORTAGNOSEG	39.6(3.7)*	–
BHMM	36.4(0.7)*	–
MORTAGTRUETAGS	–	59.8(0.4)*

Table 3: Spanish Ornat corpus results. Standard deviations are in parentheses; $*$ denotes a significant difference from the MORTAG model.

Sampling is run for 5000 iterations with annealing. Inspection of the posterior log-likelihood indicates that the models converge after about 1000 iterations. We run inference over all models ten times and report the average performance. Significance is reported using the non-parametric Wilcoxon rank-sum test with a significance level of $\rho < 0.05$.

5.5 Results: English

Results on the English Eve corpus are shown in Table 2. We use PYP parameters $a = 0.3$ and $b = 10$, though we found similar performance over a wide range of values of a and b. Our results show a clear improvement in the morphological segmentations found by the joint model and stable tagging performance across all models with context information.

The syntactic clusters found by models using only morphological patterns, MORTAGNOTRANS and MORCLUSTERS, are clearly inferior and lead to low Tag VM results. The models with local syntactic context all perform approximately equally well in terms of finding tags. We find no improvement on tagging performance in English when adding morphology, compared to the MORTAGNOSEG baseline in which words are not segmented. However, we do see a small but significant improvement over the BHMM for both of these models, due to the replacement of the multinomial emission distribution in the BHMM with the PYP.

Morphological segmentations, as measured by Suffix VM, clearly improve with the addition of local contexts (and the ensuing better tags): the full model outperforms the baselines without syntactic contexts. On this dataset, the joint MORTAG model even matches the performance of the model using oracle tags. The standard deviation over Suffix VM scores is quite large for MORTAG and MORCLUSTERS; this is due to frequent words having two high probability segmentations (most notably *is*, which in some runs was segmented as *i.s*).

5.6 Results: Spanish

For the Spanish Ornat corpus, we found slightly different optimal PYP hyperparameters and set $a = 0.1$ and $b = 0.1$. Results are shown in Table 3.

The Spanish results pattern in the opposite way as English. Here we see a statistically significant improvement in tagging performance of the full joint model over both models without morphology (MORTAGNOSEG and BHMM). Models without context information again find much worse tags, mainly because (as in English) function words are not identifiable by suffixes.

However, the full model does not find better morphological segmentations than the MORCLUSTERS model, despite better tags (the two models' Suffix VM scores are not statistically significantly different). We also see that the difference between the segmentations found by the model using gold tags and estimated tags is quite large. This is due to the oracle model finding the rarer suffixes which were not distinguished by the models with noisier tags. This demonstrates the importance of syntactic categorisation for the morpheme induction task, and suggests that a more sophisticated tagging model (with better performance) may yet improve morpheme segmentation performance in Spanish.

6 Conclusion

We have presented a model of joint syntactic category and morphology induction. Operating within a generative Bayesian framework means that combining single-task components is straightforward and well-founded. Our model is token-based, allowing for syntactic and morphemic ambiguity.

To our knowledge, this is the first joint model to be tested on child-directed speech data, which is less complex than the newswire corpora used by previous joint models. Child-directed speech may be simple enough for joint learning not to be necessary: our results indicate the contrary, namely that joint learning is indeed helpful when learning from realistic acquisition data.

We tested this model on two languages with different morphological characteristics. On English, a language with relatively little morphology, especially in child directed speech, we found that better categorisation of words yielded much better morphology in terms of suffixes learned. Conversely, in Spanish we saw less difference on the morphology task between models with categories inferred solely from morphemic patterns and models that also used local syntactic context for categorisation. However, in Spanish we saw an improvement in the tagging task when morphology information was included.

This suggests that English and Spanish make different word-order and morphology trade-offs. In English, local context provides at least as much information as morphology in terms of determining the correct syntactic category, but knowing a good estimate of the correct syntactic category is useful for determining a word's morphology. In Spanish, a word's morphology can more easily be determined simply by looking at frequent suffixes within a purely morphological system. On the other hand, word order is freer, making local syntactic context unreliable, so taking morphological information into account can improve tagging. These differences between languages demonstrate the benefits of joint learning, which enables the learner to more flexibly utilise the information available in the input data.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Cote, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, 2010.

Barry J. Blake. *Case*. Cambridge University Press, 2001.

Phil Blunsom and Trevor Cohn. A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

Roger Brown. *A first language: The early stages*. Harvard University Press, Cambridge, MA, 1973.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. A Bayesian mixture model for part-of-speech induction using multiple features. In *Proceedings of the 16th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

Alexander Clark. Combining distributional and morphological information for part of speech induction. In *Proceedings of the 10th annual Meeting of the European Association for Computational Linguistics (EACL)*, 2003a.

Eve V. Clark. *First Language Acquisition*. Cambridge University Press, 2003b.

Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):1–34, 2007.

Sajib Dasgupta and Vincent Ng. Unsupervised part-of-speech acquisition for resource-scarce languages. In *Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2007.

Gabriel Doyle and Roger Levy. Combining multiple information types in Bayesian word segmentation. In *Proceedings of NAACL-HLT 2013*, pages 117–126, 2013.

Micha Elsner, Sharon Goldwater, and Jacob Eisenstein. Bootstrapping a unified model of lexical and phonetic acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

Naomi Feldman, Thomas Griffiths, and James Morgan. Learning phonetic categories by learning a lexicon. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society (CogSci)*, 2009.

Michelle A. Fullwood and Timothy J. O'Donnell. Learning non-concatenative morphology. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, 2013.

John Goldsmith. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(4):353–371, December 2006.

Sharon Goldwater and Thomas L. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems 18*, 2006.

Aria Haghighi and Dan Klein. Prototype-driven grammar induction. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

Harald Hammarström and Lars Borin. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350, 2011.

Mark Johnson. Using Adaptor Grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

Mikko Kurimo, Sami Virpioja, and Ville T. Turunen. Proceedings of the MorphoChallenge 2010 workshop. Technical Report TKK-ICS-R37, Aalto University School of Science and Technology, Espoo, Finland, 2010.

Tom Kwiatkowski, Sharon Goldwater, Luke Zettelmoyer, and Mark Steedman. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2012.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Simple type-level unsupervised POS tagging. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Modeling syntactic context improves morphological segmentation. In *Proceedings of Fifteenth Conference on Computational Natural Language Learning*, 2011.

Brian MacWhinney. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Mahwah, NJ, 2000.

Brian MacWhinney, Elizabeth Bates, and Reinhold Kliegl. Cue validity and sentence interpretation in English, German, and Italian. *Journal of Verbal Learning and Verbal Behavior*, 23:127–150, 1984.

Thomas McFadden. On morphological case and word-order freedom. In *Proceedings of the Annual Meeting of the Berkeley Linguistics Society*, volume 29, pages 295–306, 2003.

S. Lopez Ornat. *La adquisicion de la lengua espagnola*. Siglo XXI, Madrid, 1994.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2007.

Kairit Sirts and Tanel Alumäe. A hierarchical Dirichlet process model for joint part-of-speech and morphology induction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2012.

Dan Slobin. Universal and particular in the acquisition of language. In Eric Wanner and Lila R. Gleitman, editors, *Language acquisition: the state*

of the art, pages 128–170. Cambridge University Press, 1982.

Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

Yee Whye Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

Aris Xanthos, Sabine Laaha, Steven Gillis, Ursula Stephany, Ayhan Aksu-Koç, Anastasia Christofidou, Natalia Gagarina, Gordana Hrzica, F. Nihan Ketrez, Marianne Kilani-Schoch, Katharina Korecky-Kröll, Melita Kovačević, Klaus Laalo, Marijan Palmović, Barbara Pfeiler, Maria D. Voeikova, and Wolfgang U. Dressler. On the role of morphological richness in the early development of noun and verb inflection. *First Language*, 31(4):461–479, 2011.

A Joint Learning Model of Word Segmentation, Lexical Acquisition, and Phonetic Variability

Micha Elsner
melsner0@gmail.com
Dept. of Linguistics
The Ohio State University

Sharon Goldwater
sgwater@inf.ed.ac.uk
ILCC, School of Informatics
University of Edinburgh

Naomi H. Feldman
nhf@umd.edu
Dept. of Linguistics
University of Maryland

Frank Wood
fwood@robots.ox.ac.uk
Dept. of Engineering
University of Oxford

Abstract

We present a cognitive model of early lexical acquisition which jointly performs word segmentation and learns an explicit model of phonetic variation. We define the model as a Bayesian noisy channel; we sample segmentations and word forms simultaneously from the posterior, using beam sampling to control the size of the search space. Compared to a pipelined approach in which segmentation is performed first, our model is qualitatively more similar to human learners. On data with variable pronunciations, the pipelined approach learns to treat syllables or morphemes as words. In contrast, our joint model, like infant learners, tends to learn multiword collocations. We also conduct analyses of the phonetic variations that the model learns to accept and its patterns of word recognition errors, and relate these to developmental evidence.

1 Introduction

By the end of their first year, infants have acquired many of the basic elements of their native language. Their sensitivity to phonetic contrasts has become language-specific (Werker and Tees, 1984), and they have begun detecting words in fluent speech (Jusczyk and Aslin, 1995; Jusczyk et al., 1999) and learning word meanings (Bergelson and Swingley, 2012). These developmental cooccurrences lead some researchers to propose that phonetic and word learning occur jointly, each one informing the other (Swingley, 2009; Feldman et al., 2013). Previous computational models capture some aspects of this joint learning problem, but typically simplify the problem considerably, either by assuming an unrealistic degree of phonetic regularity for word segmentation (Goldwater et al., 2009) or assuming pre-segmented input for phonetic and lexical acquisition (Feldman et al., 2009; Feldman et al., in press; Elsner et al., 2012). This paper presents, to our knowledge, the first broad-coverage model that learns to segment phonetically variable input into words, while simultaneously learning an explicit model of phonetic variation that allows it to cluster together segmented tokens with different phonetic realizations (e.g., *[ju]* and *[jɪ]*) into lexical items (*/ju/*).

We base our model on the Bayesian word segmentation model of Goldwater et al. (2009) (henceforth GGJ), using a noisy-channel setup where phonetic variation is introduced by a finite-state transducer (Neubig et al., 2010; Elsner et al., 2012). This integrated model allows us to examine how solving the word segmentation problem should affect infants' strategies for learning about phonetic variability and how phonetic learning can allow word segmentation to proceed in ways that mimic the idealized input used in previous models.

In particular, although the GGJ model achieves high segmentation accuracy on phonemic (non-variable) input and makes errors that are qualitatively similar to human learners (tending to undersegment the input), its accuracy drops considerably on phonetically noisy data and it tends to oversegment rather than undersegment. Here, we demonstrate that when the model is augmented to account for phonetic variability, it is able to learn common phonetic changes

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 42–54,
Seattle, Washington, USA, 18-21 October 2013.

and by doing so, its accuracy improves and its errors return to the more human-like undersegmentation pattern. In addition, we find small improvements in lexicon accuracy over a pipeline model that segments first and then performs lexical-phonetic learning (Elsner et al., 2012). We analyze the model's phonetic and lexical representations in detail, drawing comparisons to experimental results on adult and infant speech processing. Taken together, our results support the idea that a Bayesian model that jointly performs word segmentation and phonetic learning provides a plausible explanation for many aspects of early phonetic and word learning in infants.

2 Related Work

Nearly all computational models used to explore the problems addressed here have treated the learning tasks in isolation. Examples include models of word segmentation from phonemic input (Christiansen et al., 1998; Brent, 1999; Venkataraman, 2001; Swingley, 2005) or phonetic input (Fleck, 2008; Rytting, 2007; Daland and Pierrehumbert, 2011; Boruta et al., 2011), models of phonetic clustering (Vallabha et al., 2007; Varadarajan et al., 2008; Dupoux et al., 2011) and phonological rule learning (Peperkamp et al., 2006; Martin et al., 2013).

Elsner et al. (2012) present a model that is similar to ours, using a noisy channel model implemented with a finite-state transducer to learn about phonetic variability while clustering distinct tokens into lexical items. However (like the earlier lexical-phonetic learning model of Feldman et al. (2009; in press)) their model assumes known word boundaries, so to perform both segmentation and lexical-phonetic learning, they use a pipeline that first segments using GGJ and then applies their model to the results.

Neubig et al. (2010) also present a transducer-based noisy channel model that performs joint inference on two out of the three tasks we consider here; their model assumes fixed probabilities for phonetic changes (the noise model) and jointly infers the word segmentation and lexical items, as in our 'oracle' model below (though unlike our system their model learns from phone lattices rather than a single transcription). They evaluate only on phone recognition, not scoring the inferred lexical items.

Recently, Börschinger et al. (2013) did present a joint learner for segmentation, phonetic learning, and lexical clustering, but the model and inference are tailored to investigate word-final /*t*/-deletion, rather than aiming for a broad coverage system as we do.

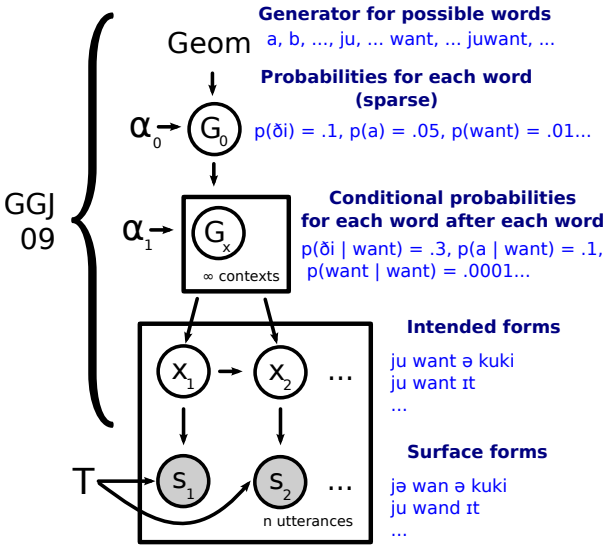


Figure 1: The graphical model for our system (Eq. 1-4). Note that the s_i are not distinct observations; they are concatenated together into a continuous sequence of characters which constitute the observations.

3 Model

We follow several previous models of lexical acquisition in adopting a Bayesian noisy channel framework (Eq. 1-4; Fig. 1). The model has two components: a *source* distribution $P(X)$ over utterances without phonetic variability X, i.e., *intended forms* (Elsner et al., 2012) and a *channel* or *noise* distribution $T(S|X)$ that translates them into the observed surface forms S. The boundaries between surface forms are then deterministically removed so that the actual observations are just the unsegmented string of characters in the surface forms.

$$G_0|\alpha_0, p_{stop} \sim DP(\alpha_0, Geom(p_{stop})) \quad (1)$$

$$G_x|G_0, \alpha_1 \sim DP(\alpha_1, G_0) \quad (2)$$

$$X_i|X_{i-1} \sim G_{X_{i-1}} \quad (3)$$

$$S|X;\theta \sim T(S|X;\theta) \quad (4)$$

The source model is an exact copy of GGJ[1]: to generate the intended-form word sequences X, we

[1]We use their best reported parameter values: $\alpha_0 = 3000, \alpha_1 = 100, p_{stop} = .2$ and for unigrams, $\alpha_0 = 20$.

sample a random language model from a hierarchical Dirichlet process (Teh et al., 2006) with character strings as atoms. To do so, we first draw a unigram distribution G_0 from a Dirichlet process prior whose base distribution generates intended form word strings by drawing each phone in turn until the stop character is drawn (with probability p_{stop}). Then, for each possible context word x, we draw a conditional distribution on words following that context $G_x = P(X_i = \bullet | X_{i-1} = x)$ using G_0 as a prior. Finally, we sample word sequences $x_1 \dots x_n$ from the bigram model.

The channel model is a finite transducer with parameters θ which independently rewrites single characters from the intended string into characters of the surface string. We use MAP point estimates of these parameters; single characters (without n-gram context) are used for computational efficiency. Also for efficiency, the transducer can insert characters into the surface string, but cannot delete characters from the intended string. As in several previous phonological models (Dreyer et al., 2008; Hayes and Wilson, 2008), the probabilities are learned using a feature-based log-linear model. For features, we use all the unigram features from Elsner et al. (2012), which check faithfulness to voicing, place and manner of articulation (for example, for $k \rightarrow g$, active features are *faith-manner*, *faith-place*, *output-g* and *voiceless-to-voiced*).

Below, we present two methods for learning the transducer parameters θ. The *oracle* transducer is estimated using the gold-standard word segmentations and intended forms for the dataset; it represents the best possible approximation under our model of the actual phonetics of the dataset. We can also estimate the transducer using the *EM* algorithm. We first initialize a simple transducer by putting small weights on the faithfulness features to encourage phonologically plausible changes. With this initial model, we begin running the sampler used to learn word segmentations. After several hundred sampler iterations, we start re-estimating the transducer by maximum likelihood after each iteration. We regularize our estimates by adding 200 pseudocounts for the rewrite $x \rightarrow x$ during training (rather than regularizing the weights for particular features). We also show *segment only* results for a model without the transducer component (i.e., $S = X$); this recovers the GGJ baseline.

4 Inference

Inference for this model is complicated for two reasons. First, the hypothesis space is extremely large. Since we allow the input string to be probabilistically lengthened, we cannot be sure how long it is, nor which characters it contains. Second, our hypotheses about nearby characters are highly correlated due to lexical effects. When deciding how to interpret *[wənt]*, if we posit that the intended vowel is */ʌ/*, the word is likely to be */wʌn/* "one" and the next word begins with */t/*; if instead we posit that the vowel is */ɔ/*, the word is probably */wɔnt/* "want". Thus, inference methods that change only one character at a time are unlikely to mix well. Since they cannot simultaneously change the vowel and resegment the */t/*, they must pass through a low-probability intermediate state to get from one state to the other, so will tend to get stuck in a bad local minimum. A Gibbs sampler which inserts or deletes a single segment boundary in each step (Goldwater et al., 2009) suffers from this problem.

Mochihashi et al. (2009) describe an inference method with higher mobility: a block sampler for the GGJ model that samples from the posterior over analyses of a whole utterance at once. This method encodes the model as a large HMM, using dynamic programming to select an analysis. We encode our own model in the same way, constructing the HMM and composing it with the transducer (Mohri, 2004) to form a larger finite-state machine which is still amenable to forward-backward sampling.

4.1 Finite-state encoding

Following Mochihashi et al. (2009) and Neubig et al. (2010), we can write the original GGJ model as a Hidden Semi-Markov model. States in the HMM, written `ST:[`w`][`C`]`, are labeled with the previous word w and the sequence of characters C which have so far been incorporated into the current word. To produce a word boundary, we transition from `ST:[`w`][`C`]` to `ST:[`C`][]` with probability $P(x_i = C | x_{i-1} = w)$. We can also add the next character s to the current word, transitioning from `ST:[`w`][`C`]` to `ST:[`w`][`$C:s$`]`, at no cost (since the full cost of the word is paid at its boundary, there

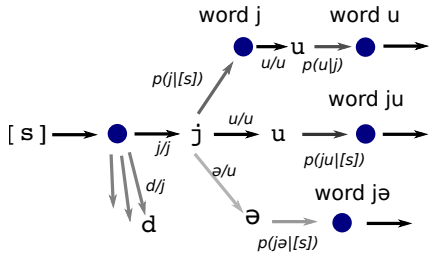


Figure 2: A fragment of the composed finite-state machine for word segmentation and character replacement for the surface string *ju*. The start state [s] is followed by a word boundary (filled circle); the next intended character is probably *j* but can be *d* or others with lower probability. After *j* can be a word boundary (forming the intended word *j*), or another character such as *u*, *ə* or other (not shown) alternatives.

is no cost for the individual characters)[2].

In addition to analyses using known words, we can also encode the uniform-geometric prior over unknown words using a finite-state machine. We can choose to select a word from the prior by transitioning to a state `ST:[`$Geom$`][]` with probability $P(\text{new word}|x_{i-1} = w)$ immediately after a word boundary. While in $Geom$, we can transition to a new $Geom$ state and produce any character with uniform probability $P(c) = (1-P_{stop})\frac{1}{|C|}$; otherwise, we can end the word, transitioning to `ST:[`$unk.word$`][]`, with probability P_{stop}.

This construction is also approximate; it ignores the possibility that the prior will generate a known word w, in which case our final transition ought to be to `ST:[`w`][]` instead of `ST:[`$unk.word$`][]`. This approximation means we do not need to add context to the $Geom$ state to remember the sequence of characters it produced, which allows us to keep only a single $Geom$ state on the chart at each timestep.

When we compose this model with the channel model, the number of states expands. Each state must now keep track of the previous word, what intended characters C have been posited and what surface characters S have been recognized, `ST:[`w`][`C`][`S`]`. To recognize the current word, we transition to `ST:[`C`][][][]` with probability $P(x_i = C|x_{i-1} = w)$. To parse a new surface character s by positing intended character x (note that x might be ϵ), we transition to `ST:[`w`][`$C:x$`][`$S:s$`]` with probability $T(s|x)$. (As above, we pay no cost for our choice of x, which is paid for when we recognize the word; however, we must pay for s.) For efficiency, we do not allow the G_0 states to hypothesize different surface and intended characters, so when we initially propose an unknown word, it must surface as itself.[3]

4.2 Beam sampler

This machine has too many states to fully fill the chart before backward sampling, so we restrict the set of trajectories under consideration using beam sampling (Van Gael et al., 2008) and simulated annealing.

The beam sampler is closely related to the standard beam search technique, which uses a probability cutoff to discard parts of the FST which are unlikely to figure in the eventual solution. Unlike conventional beam search, the sampler explores using stochastic cutoffs, so that all trajectories are explored, but most of the bad ones are explored infrequently, leading to higher efficiency.

We design our beam sampler to restrict the set of potential intended characters at each timestep. In particular, given a stream of input characters $S = s_1 \dots s_n$, we introduce a set of auxiliary cutoff variables $U = u_1 \dots u_n$. The u_i variables represent limits on the probability of the emission of surface character s_i; we exclude any hypothesized x_i whose probability of generating s_i, $T(s_i|x_i)$, is less than u_i. To create a beam sampling scheme, we must devise a distribution for U given a state sequence Q (as discussed above, the sequence of states encodes the intended character sequence and the segmentation of the surface string), $P_u(U|Q)$ and then incorporate the probability of U into the forward messages.

If q_i is the state in Q at which s_i is generated, and x_i the corresponding intended character, we require that $P_u < T(s_i|x_i)$; that is, the cutoffs must not exclude any states in the sequence Q. We define P_u

[2] Though not mentioned by Mochihashi et al. (2009) or Neubig et al. (2010), this construction is not exact, since transitions in a Bayesian HMM are exchangeable but not independent (Beal et al., 2001): if a word occurs twice in an utterance, its probability is slightly higher the second time. For single utterances, this bias is small and easy to correct for using a Metropolis-Hastings acceptance check (Börschinger and Johnson, 2012) using the path probability from the HMM as the proposal.

[3] Again, this approximation is corrected for by the Metropolis-Hastings step.

as a λ-mixture of two distributions:

$$P_u(u|s_i, x_i) = \lambda U[0, min(.05, T(s_i|x_i))] + (1-\lambda)T(s_i|x_i)Beta(5, 1e-5)$$

The former distribution is quite unrestrictive, while the latter prefers to prune away nearly all the states. Thus, for most characters in the string, we do not permit radical changes, while for a fraction, we do.

We follow Huggins and Wood (2013), who extended Van Gael et al. (2008) to the case of a non-uniform P_u, to define our forward message α as:

$$\alpha(q_i, i) \propto P(q_i, S_{0..i}, U_{0..i}) \quad (5)$$
$$= \sum_{q_{i-1}} P_u(u_i|s_i, x_i)T(s_i|x_i)\alpha(q_{i-1}, i-1)$$

This is the standard HMM forward message, augmented with the probability of u. Since $P_u(\cdot|s_i, x_i)$ is required to be less than $T(s_i|x_i)$, it will be 0 whenever $T(s_i|x_i) < u$; this is how the u variables function as cutoffs. In practice, we use the u variables to filter the lexical items that begin at each position i in advance, using a simple 0/1 edit distance Markov model which runs faster than our full model. (For example, we can quickly check if the current U allows *want* as the intended form for *wɔlk* at i; if not, we can avoid constructing the prefix `ST:[`x_{i-1}`][wa][wɔ]` since the continuation will fail.)

The algorithm's speed depends on the size and uncertainty of the inferred LM: large numbers of plausible words mean more states to explore. When inference starts, and the system is highly uncertain about word boundaries, it is therefore reasonable to limit the exploration of the character sequence. We do so by annealing in two ways: as in Goldwater et al. (2009), we raise $P(X)$ (Eq. 3) to a power t which increases linearly from .3. To sample from the posterior, we would want to end with $t = 1$, but as in previous noisy-channel models (Elsner et al., 2012; Bahl et al., 1980) we get better results when we emphasize the LM at the expense of the channel and so end at $t = 2$. Meanwhile, as t rises and we explore fewer implausible lexical sequences, we can explore the character sequence more. We begin by setting the λ interpolation parameter of P_u to 0 to minimize exploration and increase it linearly to .3 (allowing the system to change about a third of the characters on each sweep). This is similar to the scheme for altering P_u in Huggins and Wood (2013).

4.3 Dataset and metrics

We use the corpus released by Elsner et al. (2012), which contains 9790 child-directed English utterances originally from the Bernstein-Ratner corpus (Bernstein-Ratner, 1987) and later transcribed phonemically (Brent, 1999). This standard word segmentation dataset was modified by Elsner et al. (2012) to include phonetic variation by assigning each token a pronunciation independently selected from the empirical distribution of pronunciations of that word type in the closely-transcribed Buckeye Speech Corpus (Pitt et al., 2007). Following previous work, we hold out the last 1790 utterances as unseen test data during development. In the results presented here, we run the model on all 9790 utterances but score only these 1790. We average results over 5 runs of the model with different random seeds.

We use standard metrics for segmentation and lexicon recovery. For segmentation, we report precision, recall and F-score for word boundaries (*bds*), and for the positions of word tokens in the surface string (*srf*; both boundaries must be correct).

For normalization of the pronunciation variation, we follow Elsner et al. (2012) in measuring how well the system clusters together variant pronunciations of the same lexical item, without insisting that the intended form the system proposes for them match the one in our corpus. For example, if the system correctly clusters *[ju]* and *[jɪ]* together but assigns them the incorrect intended form */jɪ/*, we can still give credit to this cluster if it is the one that overlaps best with the gold-standard */ju/* cluster. To compute these scores, we find the optimal one-to-one mapping between our clusters of pronunciations and the true lexical entries, then report scores for mapped tokens (*mtk*; boundaries and mapping to gold standard cluster must be correct) and mapped types[4] (*mlx*).

[4]Elsner et al. (2012) calls the *mlx* metric *lexicon F*, which is possibly confusing. We map the clusters to a gold-standard lexicon (plus potentially some words that don't correspond to anything in the gold standard) and compute a type-level F-score on this lexicon.

	Prec	Rec	F-score
Pipeline (segment, then cluster): (Elsner et al., 2012)			
Bds	70.4	**93.5**	80.3
Srf	56.5	69.7	62.4
Mtk	44.2	**54.5**	48.8
Mlx	48.6	43.1	45.7
Bigram model, segment only			
Bds	73.9 (-0.6:0.7)	91.0 (-0.6:0.4)	**81.6** (-0.5:0.6)
Srf	60.8 (-0.7:1.1)	**70.8** (-0.8:0.9)	65.4 (-0.6:1.0)
Mtk	41.6 (-0.6:1.2)	48.4 (-0.5:1.2)	44.8 (-0.6:1.2)
Mlx	36.6 (-0.7:0.8)	**49.8** (-1.0:0.8)	42.2 (-0.9:0.8)
Unigram model, oracle transducer			
Bds	**81.4** (-0.8:0.4)	72.1 (-0.9:0.8)	76.4 (-0.5:0.7)
Srf	63.6 (-1.0:1.1)	58.5 (-1.2:1.2)	60.9 (-0.9:1.2)
Mtk	46.8 (-1.0:1.1)	43.0 (-1.1:1.2)	44.8 (-1.0:1.2)
Mlx	**56.7** (-1.1:1.0)	47.6 (-1.4:0.8)	**51.7** (-1.2:0.8)
Bigram model, oracle transducer			
Bds	76.1 (-0.6:0.6)	83.8 (-0.9:1.0)	79.8 (-0.8:0.4)
Srf	62.2 (-0.9:1.0)	66.7 (-1.2:1.1)	64.4 (-1.1:0.8)
Mtk	47.2 (-0.7:0.9)	50.6 (-1.0:0.8)	48.8 (-0.8:0.7)
Mlx	40.1 (-1.0:1.2)	43.7 (-0.6:0.7)	41.8 (-0.8:0.6)
Bigram model, EM transducer			
Bds	80.1 (-0.5:0.8)	83.0 (-1.4:1.3)	81.5 (-0.5:0.7)
Srf	**66.1** (-0.8:1.4)	67.8 (-1.4:1.7)	**66.9** (-0.9:1.4)
Mtk	**49.0** (-0.9:0.7)	50.3 (-1.1:1.4)	**49.6** (-1.0:1.0)
Mlx	43.0 (-1.0:1.4)	49.5 (-1.5:1.1)	46.0 (-1.0:1.3)

Table 1: Mean segmentation (*bds*, *srf*) and normalization (*mtk*, *mlx*) scores on the test set over 5 runs. Parentheses show min and max scores as differences from the mean.

5 Results and discussion

In the following sections, we analyze how our model with variability compares to GGJ on noisy data. We give quantitative scores and also show that qualitative patterns of errors are often similar to those of human learners and listeners.

5.1 Clean versus variable input

We begin by evaluating our model as a word segmentation system. (Table 1 gives segmentation and normalization scores for various models and baselines on the 1790 test utterances.) We first confirm that our inference method is reasonable. The bigram model without variability ("segment only") should have the same segmentation performance as the standard `dpseg` implementation of GGJ. This is the case: `dpseg` has boundary F of 80.3 and token F of 62.4; we get 81.6 and 65.4. Thus, our sampler is finding good solutions, at least for the no-variability model.

We compare segmentation scores between the "segment only" system and the two bigram models with transducers ("oracle" and "EM"). While these systems all achieve similar segmentation scores, they do so in different ways. "Segment only" finds a solution with boundary precision 73.9% and boundary recall 91.0% for a total F of 81.6%. The low precision and high recall here indicate a tendency to oversegment; when the analysis of a given subsequence is unclear, the system prefers to chop it into small chunks. The bigram models which incorporate transducers score P: 76.1, R: 83.8 (oracle) and P: 80.1, R: 83.0 (EM), indicating that they prefer to find longer sequences (undersegment) more.

In previous experiments on datasets without variation, GGJ also has a strong tendency to undersegment the data (boundary P: 90.1, R: 80.3), which Goldwater et al. argue is rational behavior for an ideal learner seeking a parsimonious explanation for the data. Undersegmentation occurs especially when ignoring lexical context (a unigram model), but to some extent even in bigram models. Human learners also tend to learn collocations as single words (Peters, 1983; Tomasello, 2000), and the GGJ model has been shown to capture several other effects seen in laboratory segmentation tasks (Frank et al., 2010). Together, these findings support the idea that human learners may behave in important respects like the Bayesian ideal learners that Goldwater et al. presented.

However, experiments on data with variation have called these conclusions into question. In particular, GGJ has previously been shown to oversegment rather than undersegment as the input grows noisier (Fleck, 2008), and our results replicate this finding (oversegmentation for the "segment only" model). In addition, the GGJ bigram model, which achieves much higher segmentation accuracy than the unigram model on clean data, actually performs worse on very noisy data (Jansen et al., 2013). Infants are known to track statistical dependencies across words (Gómez and Maye, 2005), so it is worrisome that these dependencies hurt GGJ's segmentation accuracy when learning from noisy data.

Our results show that modeling phonetic variability reverses the problematic trends described above. Although the models with phonetic variability show similar overall segmentation accuracy on noisy data to the original GGJ model, the pattern of errors changes, with less oversegmentation and more un-

dersegmentation. Thus, their qualitative performance on variable data resembles GGJ's on clean data, and therefore the behavior of human learners.

5.2 Phonetic variability

We next analyze the model's ability to normalize variations in the pronunciation of tokens, by inspecting the *mtk* score. The "segment only" baseline is predictably poor, F: 44.8. The pipeline model scores 48.8, and our oracle transducer model matches this exactly. The EM transducer scores better, F: 49.6. Although the confidence intervals overlap slightly, the EM system also outperforms the pipeline on the other F-measures; altogether, these results suggest at least a weak learning synergy (Johnson, 2008) between segmentation and phonetic learning.

It is interesting that EM can perform better than the oracle. However, EM is more conservative about which sound changes it will allow, and thus tends to avoid mistakes caused by the simplicity of the transducer model. Since the transducer works segment-by-segment, it can apply rare contextual variations out of context. EM benefits from not learning these variations to begin with.

We can also compare the bigram and unigram versions of the model. The unigram model is a reasonable segmenter, though not quite as good as the bigram model, with boundary F of 76.4 and token F of 60.9 (compared to 79.8 and 64.4 using the bigram model). However, it is not good at normalizing variation; its *mtk* score is comparable to the baseline at 44.8%[5]. Although bigram context is only moderately effective for telling where words are, the model seems heavily reliant on lexical context to decide *what* words it is hearing.

5.3 Error analysis

To gain more insight into the differing behavior of our model versus a pipelined system, we inspect the intended word strings X proposed by each one in detail. Below, we categorize the kinds of intended word strings that the model might propose to span a given gold-standard word token:

Correct Correctly segmented, mapped to the correct lexical item (e.g., gold intended */ju/*, surface segmentation *[ju]*, intended */ju/*)

Wrong form Correctly segmented, mapped to the wrong lexical item (*/ju/*, surf. *[ju]*, int. */jɛs/*)

Colloc Missegmented as part of a sequence whose boundaries correspond to real word boundaries (*/ju•want/*, surf. *[juwant]*, int. */juwant/*)

Corr. colloc As above, but proposed lexical item maps to this word (*/ar•ju/*, surf. *[arjə]* int. */ju/*)

Split Missegmented with a word-internal boundary (*/dɔgiz/*, surf. *[dɔ•giz]*, int. */dɔ•giz/*)

Corr. split As above, but one proposed word maps correctly (*/dɔgi/*, surf. *[dɔg•i]*, int. */dɔgi•ə/*)

One boundary One boundary correct, the other wrong (*/ju•wa…/*, surf. *[juw]*, int. */juw/*)

Other Not a collocation, both boundaries are wrong (*/du•ju•wa…/*, surf. *[ujuw]*, int. */ujuw/*)

	EM-learned	Segment only
Correct	49.88	47.61
Wrong form	17.96	23.73
Collocation	14.25	7.59
Split	8.26	15.18
One bound	7.11	15.18
Corr. colloc.	1.35	< 0.01
Other	0.75	0.22
Corr. split	0.43	0.66

Table 2: Distribution (%) of error types (see text) in a single run on the full dataset.

Table 2 shows the distribution over intended word strings proposed by the "segment only" baseline and the EM-learned transducer. Both systems propose a large number of correct forms, and the most common error category is "wrong form" (lexical error without segmentation error), an error which could potentially be repaired in a pipeline system. However, the remaining errors represent segmentation mistakes which a pipeline could not repair. Here the two systems behave quite differently. The EM-learned transducer analyses 14% of real tokens as parts of multiword collocations like "doyou"; in another 1.35%, the underlying content word is even correctly detected. The non-variable system, on the other hand, analyses 15% of real tokens by splitting them into pieces. Since infant learners tend to learn collocations, this supports our analysis that the model with variation better models human behavior.

[5] Elsner et al. (2012) show a similar result for a unigram version of their pipelined system.

EM	*ju*: 805, *duju*: 239, *juwan*: 88, *jɪ*: 58, *eˑju*: 54, *judu*: 47, *jæ*: 39, *julʌk*: 39, *ʃu*: 30, *u*: 23, *ʒu*: 18, *j*: 17, *jeˑ*: 16, *tʃu*: 15, *aj*:15, *ðerjugo*: 12, *dʒu*: 12
GGJ	*ju*: 498, *jɪ*: 280, *jə*: 165, *ji*: 119, *duju*: 106, *dujɪ*: 44, *kɪnju*: 39, *i*: 32, *u*: 29, *kɪnjɪ*: 29, *julʌk*: 24, *juwan*: 23, *j*: 22, *ʃu*: 19, *jʊ*: 18, *eˑju*: 18, *ɹ*:16, *ʒu*: 15, *dʒ•u*: 13, *jɛ*: 12, *ʃɪ*: 11, *θæŋkju*: 11

Table 3: Forms proposed with frequency > 10 for gold-standard tokens of "you" in one sample from EM-transducer and segment-only (GGJ) system.

To illustrate this behavior anecdotally, we present the distribution of intended word strings spanning tokens whose gold intended form is */ju/* "you" (Table 3). The EM-learned solution proposes 805 tokens of */ju/*, which is the correct analysis[6]; the "segment only" system instead finds varying forms like */jɪ/*, */jæ/* etc. This is unsurprising and could be repaired by a suitable pipelined system. However, the EM system also proposes 239 instances of "doyou", 88 instances of "youwant", 54 instances of "areyou" and several other collocations. The "segment only" system finds some of these collocations, split into different versions: for instance 106 instances of */duju/* and 44 of */dujɪ/*. In a pipelined system, we could combine these variants to find 150 instances— but this is still 89 instances short of the 239 found when allowing for variability. The same pattern holds for "youlike" and "youwant". Because the non-variable system must learn each variant separately, it learns only the most common instances of these long collocations, and analyzes infrequent variants differently.

We also perform this analysis specifically for words beginning with vowels. Infants show a delay in their ability to segment these words from continuous speech (Mattys and Jusczyk, 2001; Nazzi et al., 2005; Seidl and Johnson, 2008), and Seidl and Johnson (2008) suggest a perceptual explanation— initial vowels can be hard to hear and often exhibit variation due to coarticulation or resyllabification. Although our dataset does not contain coarticulation as such, it should show this pattern of greater variation, which we hypothesize might lead to difficulty in segmenting and recognizing vowel-initial words.

The model's behavior is consistent with this hypothesis (Table 4). Both the "segment only" and EM transducer models find approximately the same proportion of vowel-initial tokens, and both systems do somewhat better on consonant-initial words than vowel-initial words. The advantage is stronger for the transducer model, which gets only 41.5% of vowel-initial tokens correct as opposed to 52.1% of consonant-initial words. It proposes more collocations for vowel-initial words (19.2%) than for consonants (12.5%). In cases where they do not propose a collocation, both systems are somewhat more likely to find the right boundary of a vowel-initial token than the left boundary (although again this difference is larger for the EM system); this suggests that the problem is indeed caused by the initial segment.

Segment only	Vow. init	Cons. init
Correct	47.5	51.7
Wrong form	18.6	15.7
Collocation	14.6	12.2
Split	6.2	10.8
Right bd. corr.	5.8	3.6
Left bd. corr.	4.6	3.8
EM transducer	**Vow. init**	**Cons. init**
Correct	41.5	52.1
Wrong form	20.4	17.3
Collocation	19.2	12.5
Split	5.2	9.1
Right bd. corr.	6.2	2.7
Left bd. corr.	2.7	3.1

Table 4: Most common error types (%; see text) for intended forms beginning with vowels or consonants. Rare error types are not shown. "One bound" errors are split up by which boundary is correct.

5.4 Phonetic Learning

We next compare phonetic variations learned by the model to characteristics of infant speech perception. Infants show an asymmetry between consonants and vowels, losing sensitivity to non-native vowel contrasts by eight months (Kuhl et al., 1992; Bosch and Sebastián-Gallés, 2003) but to non-native consonant contrasts only by 10-12 months (Werker and Tees, 1984). The observed ordering is somewhat puzzling when one considers the availability for distributional information (Maye et al., 2002), which is much stronger for stop consonants than for vowels (Lisker and Abramson, 1964; Peterson and Barney, 1952). Infants are also conservative in generalizing across phonetic variability, showing a delayed abil-

[6]Not all the variants are merged, however. *jɪ*, *jæ*, *ʃu* etc. are still occasionally analyzed as separate lexical items.

ity to generalize across talkers, affects, and dialects. They have difficulty recognizing word tokens that are spoken by a different talker or in a different tone of voice until 11 months (Houston and Jusczyk, 2000; Singh et al., 2004), and the ability to adapt to unfamiliar dialects appears to develop even later, between 15 and 19 months (Best et al., 2009; Heugten and Johnson, in press; White and Aslin, 2011).

Similar to infants, our model shows both a vowel-consonant asymmetry and a reluctance to accept the full range of adult phonetic variability. Table 5 shows some segment-to-segment alternations learned in various transducers. The oracle learns a large amount of variation (*u* surfaces as itself only 68% of the time) involving many different segments, whereas EM is similar to infant learners in learning a more conservative solution with fewer alternations overall. Moreover, EM appears to identify patterns of variability in vowels before consonants. It learns a similar range of alternations for *u* as in the oracle, although it treats the sound as less variable than it actually is. It learns much less variability for consonants; it picks up the alternation of *ð* with *s* and *z*, but predicts that *ð* will surface as itself 91% of the time when the true figure is only 69%. And it fails to learn any meaningful alternations involving *k*. These results suggest that patterns of variability in vowels are more evident than patterns of variability in consonants when infants are beginning to solve the word segmentation problem.

To investigate the effect of data size on this conservativism, we ran the system on 1000 utterances instead of 9790. This leads to an even more conservative solution, with variations for *u* but none of the others (although *i* and *ð* still vary more than *k*).

5.5 Segmentation and recognition errors

A particularly interesting set of errors are those that involve both a missegmentation and a simultaneous misrecognition, since the joint model is prone to such errors while the pipelined model is not. Relatively little is known about infants' misrecognitions of words in fluent speech, although it is clear that they find words in medial position harder (Plunkett, 2005; Seidl and Johnson, 2006). However, adults make missegmentation/misrecognition errors fairly often, especially when listening to noisy audio (Butterfield and Cutler, 1988). Such errors are more common when the misrecognized word belongs to a prosodically rare class and when the incorrectly hypothesized string contains frequent words (Cutler, 1990); phonetically ambiguous words are also more commonly recognized as the more frequent of two options (Connine et al., 1993). For the indefinite article "a" (often reduced to *[ə]*), lexical context is the main factor in deciding between ambiguous interpretations (Kim et al., 2012). In rapid speech, listeners have few phonetic cues to indicate whether it is present at all (Dilley and Pitt, 2010). Below, we analyze various misrecognitions made by our system (using the EM transducer), and find some similar effects.

System	x	top 4 outputs s							
Oracle	*u*	*u*	.68	*ə*	.05	*a*	.04	*ʊ*	.04
	i	*i*	.85	*ɪ*	.03	*ə*	.03	*ɛ*	.02
	ð	*ð*	.69	*s*	.07	[*ϕ*]	.07	*z*	.04
	k	*k*	.93	*d*	.02	*g*	.02		
	[*ϕ*]	*r*	.21	*h*	.11	*d*	.01	*ə*	.07
EM (full)	*u*	*u*	.75	*ə*	.08	*ɪ*	.04	*ʊ*	.03
	i	*i*	.90	*ɪ*	.04	*ɛ*	.02		
	ð	*ð*	.91	*s*	.03	*z*	0.1		
	k	*k*	.98						
	[*ϕ*]	*ə*	.32	*ɪ*	.14	*n*	.13	*t*	.13
EM (only 1000 utts)	*u*	*u*	.82	*ɪ*	.04	*ə*	.04	*a*	.02
	i	*i*	.97						
	ð	*ð*	.95						
	k	*k*	.99						
	[*ϕ*]	*ə*	.21	*ɪ*	.18	*t*	.12	*s*	.12

Table 5: Learned phonetic alternations: top 4 outputs s with $p > .001$ for inputs $x =$ uw (/u/), iy (/i/), dh (/ð/), k (/k/) and [ϕ], the null character. Outputs from [ϕ] are insertions. The oracle allows [ϕ] as an output (deletion) but for computational reasons, the model does not.

The easiest cases to analyze are those with no missegmentation: the proposed boundaries are correct, and the proposed lexical entry corresponds to a real word[7], but not the correct one. Most of them correspond to homophones (Table 6).

Common cases with a missegmentation include *it* and *is*, *a* and *is*, *it's* and *is*, *who*, *who's* and *whose*, *that's* and *what's*, and *there* and *there's*. In general, these errors involve words which sometimes appear

[7]The one-to-one mapping can be misleading, as it may map a large cluster to a real word on the basis of one or two tokens if all other tokens correspond to a different word already used for another cluster. We manually filter out a few cases like this.

Actual	proposed	count
/*tu*/ "two"	/*tə*/ "to"	95
/*kin*/ "can"	/*kænt*/ "can't"	67
/*ɛn*/ "and"	/*æn*/ "an"	61
/*hɪz*/ "his"	/*ɪz*/ "is"	57
/*ðə*/ "the"	/*ə*/ "ah"	51
/*wəts*/ "what's"	/*wants*/ "wants"	40
/*wan*/ "want"	/*won*/ "won't"	39
/*yu*/ "you"	/*yæ*/ "yeah"	39
/*fɚ*/ "for"	/*fɔr*/ "four"	30
/*hir*/ "here"	/*hil*/ "he'll"	28

Table 6: Top ten errors involving confusion between real, correctly segmented words: the most common pronunciation of the actual token and its orthographic form, the same for the proposed token, and the frequency.

with a morpheme or clitic (which can easily be missegmented as part of something else), words which differ by one segment, and frequent function words which often appear in similar contexts. These tendencies match those shown by adult human listeners.

A particularly distinctive set of joint recognition and segmentation errors are those where an entire real token is treated as phonetic "noise"— that is, it is segmented along with an adjacent word, and the system clusters the whole sequence as a token of that word. The most common examples are "that's a" identified as "that's", "have a" identified as "have", "sees a" identified as "sees" and other examples involving "a", a word which also frequently confuses humans (Kim et al., 2012; Dilley and Pitt, 2010). However, there are also instances of "who's in" as "who's", "does it" as "does", and "can you" as "can".

6 Conclusion

We have presented a model that jointly infers word segmentation, lexical items, and a model of phonetic variability; we believe this is the first model to do so on a broad-coverage naturalistic corpus[8]. Our results show a small improvement in both segmentation and normalization over a pipeline model, providing evidence for a synergistic interaction between these learning tasks and supporting claims of interactive learning from the developmental literature on infants. We also reproduced several experimental findings; our results suggest that two vowel-consonant asymmetries, one from the word segmentation literature and another from the phonetic learning literature, are linked to the large variability in vowels found in natural corpora. The model's correspondence with human behavioral results is by no means exact, but we believe these kinds of predictions might help guide future research on infant phonetic and word learning.

[8]Software is available from the ACL archive; updated versions may be posted at `https://bitbucket.org/melsner/beamseg`.

Acknowledgements

Thanks to Mary Beckman for comments. This work was supported by EPSRC grant EP/H050442/1 to the second author.

References

Lalit Bahl, Raimo Bakis, Frederick Jelinek, and Robert Mercer. 1980. Language-model/acoustic-channel-model balance mechanism. Technical disclosure bulletin Vol. 23, No. 7b, IBM, December.

Matthew J. Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. 2001. The infinite Hidden Markov Model. In *NIPS*, pages 577–584.

Elika Bergelson and Daniel Swingley. 2012. At 6-9 months, human infants know the meanings of many common nouns. *Proceedings of the National Academy of Sciences*, 109:3253–3258.

Nan Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.

Catherine T. Best, Michael D. Tyler, Tiffany N. Gooding, Corey B. Orlando, and Chelsea A. Quann. 2009. Development of phonological constancy: Toddlers' perception of native- and jamaican-accented words. *Psychological Science*, 20(5):539–542.

Benjamin Börschinger and Mark Johnson. 2012. Using rejuvenation to improve particle filtering for Bayesian word segmentation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–89, Jeju Island, Korea, July. Association for Computational Linguistics.

Benjamin Börschinger, Mark Johnson, and Katherine Demuth. 2013. A joint model of word segmentation and phonological variation for English word-final /t/-deletion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Luc Boruta, Sharon Peperkamp, Benoît Crabbé, and Emmanuel Dupoux. 2011. Testing the robustness of online word segmentation: Effects of linguistic diversity and

phonetic variation. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 1–9.

Laura Bosch and Núria Sebastián-Gallés. 2003. Simultaneous bilingualism and the perception of a language-specific vowel contrast in the first year of life. *Language and Speech*, 46(2-3):217–243.

Michael R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105, February.

Sally Butterfield and Anne Cutler. 1988. Segmentation errors by human listeners: Evidence for a prosodic segmentation strategy. In *Proceedings of SPEECH '88: Seventh Symposium of the Federation of Acoustic Societies of Europe, vol. 3*, pages 827–833, Edinburgh.

Morten H. Christiansen, Joseph Allen, and Mark S. Seidenberg. 1998. Learning to Segment Speech Using Multiple Cues: A Connectionist Model. *Language and Cognitive Processes*, 13(2/3):221–269.

C. M. Connine, D. Titone, and J. Wang. 1993. Auditory word recognition: Extrinsic and intrinsic effects of word frequency. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 19:81–94.

Anne Cutler. 1990. Exploiting prosodic probabilities in speech segmentation. In G. A. Altmann, editor, *Cognitive models of speech processing: Psycholinguistic and computational perspectives*, pages 105–121. MIT Press, Cambridge, MA.

Robert Daland and Janet B. Pierrehumbert. 2011. Learning diphone-based segmentation. *Cognitive Science*, 35(1):119–155.

Laura C. Dilley and Mark Pitt. 2010. Altering context speech rate can cause words to appear or disappear. *Psychological Science*, 21(11):1664–1670.

Markus Dreyer, Jason R. Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1080–1089, Stroudsburg, PA, USA. Association for Computational Linguistics.

Emmanuel Dupoux, Guillaume Beraud-Sudreau, and Shigeki Sagayama. 2011. Templatic features for modeling phoneme acquisition. In *Proceedings of the 33rd Annual Cognitive Science Society*.

Micha Elsner, Sharon Goldwater, and Jacob Eisenstein. 2012. Bootstrapping a unified model of lexical and phonetic acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 184–193, Jeju Island, Korea, July. Association for Computational Linguistics.

Naomi Feldman, Thomas Griffiths, and James Morgan. 2009. Learning phonetic categories by learning a lexicon. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Naomi H. Feldman, Emily B. Myers, Katherine S. White, Thomas L. Griffiths, and James L. Morgan. 2013. Word-level information influences phonetic learning in adults and infants. *Cognition*, 127(3):427–438.

Naomi H. Feldman, Thomas L. Griffiths, Sharon Goldwater, and James L. Morgan. in press. A role for the developing lexicon in phonetic category acquisition. *Psychological Review*.

Margaret M. Fleck. 2008. Lexicalized phonotactic word segmentation. In *Proceedings of ACL-08: HLT*, pages 130–138, Columbus, Ohio, June. Association for Computational Linguistics.

Michael C. Frank, Sharon Goldwater, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2010. Modeling human performance in statistical word segmentation. *Cognition*, 117(2):107–125.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Rebecca Gómez and Jessica Maye. 2005. The developmental trajectory of nonadjacent dependency learning. *Infancy*, 7:183–206.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Marieke van Heugten and Elizabeth K. Johnson. in press. Learning to contend with accents in infancy: Benefits of brief speaker exposure. *Journal of Experimental Psychology: General*.

Derek M. Houston and Peter W. Jusczyk. 2000. The role of talker-specific information in word segmentation by infants. *Journal of Experimental Psychology: Human Perception and Performance*, 26:1570–1582.

Jonathan Huggins and Frank Wood. 2013. Infinite structured hidden semi-Markov models. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, to appear, September.

Aren Jansen, Emmanuel Dupoux, Sharon Goldwater, Mark Johnson, Sanjeev Khudanpur, Kenneth Church, Naomi Feldman, Hynek Hermansky, Florian Metze, Richard Rose, Mike Seltzer, Pascal Clark, Ian McGraw, Balakrishnan Varadarajan, Erin Bennett, Benjamin Borschinger, Justin Chiu, Ewan Dunbar, Abdellah Fourtassi, David Harwath, Chia-ying Lee, Keith Levin, Atta Norouzian, Vijay Peddinti, Rachael Richardson, Thomas Schatz, and Samuel Thomas. 2013. A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and early language acquisition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Mark Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL-08: HLT*, pages 398–406, Columbus, Ohio, June. Association for Computational Linguistics.

Peter W. Jusczyk and Richard N. Aslin. 1995. Infants' detection of the sound patterns of words in fluent speech. *Cognitive Psychology*, 29:1–23.

Peter W. Jusczyk, Derek M. Houston, and Mary Newsome. 1999. The beginnings of word segmentation in English-learning infants. *Cognitive Psychology*, 39:159–207.

Dahee Kim, Joseph D.W. Stephens, and Mark A. Pitt. 2012. How does context play a part in splitting words apart? Production and perception of word boundaries in casual speech. *Journal of Memory and Language*, 66(4):509 – 529.

Patricia K. Kuhl, Karen A. Williams, Francisco Lacerda, Kenneth N. Stevens, and Bjorn Lindblom. 1992. Linguistic experience alters phonetic perception in infants by 6 months of age. *Science*, 255(5044):606–608.

Leigh Lisker and Arthur S. Abramson. 1964. A cross-language study of voicing in initial stops: Acoustical measurements. *Word*, 20:384–422.

Andrew Martin, Sharon Peperkamp, and Emmanuel Dupoux. 2013. Learning phonemes with a proto-lexicon. *Cognitive Science*, 37:103–124.

Sven L. Mattys and Peter W. Jusczyk. 2001. Do infants segment words or recurring contiguous patterns? *Journal of Experimental Psychology: Human Perception and Performance*, 27(3):644–655+.

Jessica Maye, Janet F. Werker, and LouAnn Gerken. 2002. Infant sensitivity to distributional information can affect phonetic discrimination. *Cognition*, 82(3):B101–11.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore, August. Association for Computational Linguistics.

Mehryar Mohri, 2004. *Weighted Finite-State Transducer Algorithms: An Overview*, chapter 29, pages 551–564. Physica-Verlag.

Thierry Nazzi, Laura C. Dilley, Ann Marie Jusczyk, Stefanie Shattuck-Hufnagel, and Peter W. Jusczyk. 2005. English-learning infants' segmentation of verbs from fluent speech. *Language and Speech*, 48(3):279–298+.

Graham Neubig, Masato Mimura, Shinsuke Mori, and Tatsuya Kawahara. 2010. Learning a language model from continuous speech. In *11th Annual Conference of the International Speech Communication Association (InterSpeech 2010)*, pages 1053–1056, Makuhari, Japan, 9.

Sharon Peperkamp, Rozenn Le Calvez, Jean-Pierre Nadal, and Emmanuel Dupoux. 2006. The acquisition of allophonic rules: Statistical learning with linguistic constraints. *Cognition*, 101(3):B31–B41.

Ann M. Peters. 1983. *The Units of Language Acquisition*. Cambridge Monographs and Texts in Applied Psycholinguistics. Cambridge University Press.

Gordon E. Peterson and Harold L. Barney. 1952. Control methods used in a study of the vowels. *Journal of the Acoustical Society of America*, 24(2):175–184.

Mark A. Pitt, Laura Dilley, Keith Johnson, Scott Kiesling, William Raymond, Elizabeth Hume, and Eric Fosler-Lussier. 2007. Buckeye corpus of conversational speech (2nd release).

Kim Plunkett. 2005. Learning how to be flexible with words. *Attention and Performance*, XXI:233–248.

Anton Rytting. 2007. *Preserving Subsegmental Variation in Modeling Word Segmentation (Or, the Raising of Baby Mondegreen)*. Ph.D. thesis, The Ohio State University.

Amanda Seidl and Elizabeth Johnson. 2006. Infant word segmentation revisited: Edge alignment facilitates target extraction. *Developmental Science*, 9:565–573.

Amanda Seidl and Elizabeth Johnson. 2008. Perceptual factors influence infants' extraction of onsetless words from continuous speech. *Journal of Child Language*, 34.

Leher Singh, James Morgan, and Katherine White. 2004. Preference and processing: The role of speech affect in early spoken word recognition. *Journal of Memory and Language*, 51:173–189.

Daniel Swingley. 2005. Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*, 50:86–132.

Daniel Swingley. 2009. Contributions of infant word learning to language development. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1536):3617–3632, December.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Michael Tomasello. 2000. The item-based nature of children's early syntactic development. *Trends in Cognitive Sciences*, 4(4):156 – 163.

Gautam K. Vallabha, James L. McClelland, Ferran Pons, Janet F. Werker, and Shigeaki Amano. 2007. Unsupervised learning of vowel categories from infant-directed speech. *Proceedings of the National Academy of Sciences*, 104(33):13273–13278.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008. Beam sampling for the infinite Hidden Markov model. In *Proceedings of the 25th International Conference on Machine learning*,

ICML '08, pages 1088–1095, New York, NY, USA. ACM.

Balakrishnan Varadarajan, Sanjeev Khudanpur, and Emmanuel Dupoux. 2008. Unsupervised learning of acoustic sub-word units. In *Proceedings of the Association for Computational Linguistics: Short Papers*, pages 165–168.

Anand Venkataraman. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):351–372.

Janet F. Werker and Richard C. Tees. 1984. Cross-language speech perception: Evidence for perceptual reorganization during the first year of life. *Infant Behavior and Development*, 7(1):49 – 63.

Katherine S. White and Richard N. Aslin. 2011. Adaptation to novel accents by toddlers. *Developmental Science*, 14(2):372–384.

Animacy Detection with Voting Models

Joshua L. Moore*
Dept. Computer Science
Cornell University
Ithaca, NY 14853
jlmo@cs.cornell.edu

Christopher J.C. Burges Erin Renshaw Wen-tau Yih
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{cburges, erinren, scottyih}
@microsoft.com

Abstract

Animacy detection is a problem whose solution has been shown to be beneficial for a number of syntactic and semantic tasks. We present a state-of-the-art system for this task which uses a number of simple classifiers with heterogeneous data sources in a voting scheme. We show how this framework can give us direct insight into the behavior of the system, allowing us to more easily diagnose sources of error.

1 Introduction

Animacy detection has proven useful for a variety of syntactic and semantic tasks, such as anaphora and coreference resolution (Orăsan and Evans, 2007; Lee et al., 2013), verb argument disambiguation (Dell'Orletta et al., 2005) and dependency parsing (Øvrelid and Nivre, 2007). Existing approaches for animacy detection typically rely on two types of information: linguistic databases, and syntactic cues observed from the corpus. They usually combine two types of approaches: rule based systems, and machine learning techniques. In this paper we explore a slightly different angle: we wish to design an animacy detector whose decisions are *interpretable and correctable*, so that downstream semantic modeling systems can revisit those decisions as needed. Thus here, we avoid defining a large number of features and then using a machine learning method such as boosted trees, since such methods, although powerful, result in hard-to-interpret systems. Instead, we explore combining interpretable voting models using machine learning only to reweight their votes. We show that such an approach can indeed result in a high performing system, with animacy detection accuracies in the mid 90% range, which compares well with other reported rates. Ensemble methods are well known (see for example, Dietterich (2000)) but our focus here is on using them for interpretability while still maintaining accuracy.

* Work performed while visiting Microsoft Research.

2 Previous Work

2.1 Definitions of Animacy

Previous work uses several different definitions of animacy. Orăsan and Evans (2007) define animacy in the service of anaphora resolution: an NP is considered animate "*if its referent can also be referred to using one of the pronouns he, she, him, her, his, hers, himself, herself, or a combination of such pronouns (e.g. his/her)*". Although useful for the task at hand, this has counterintuitive consequences: for example, *baby* may be considered animate or inanimate, and *ant* is considered inanimate (Ibid., Figure 1). Others have argued that animacy should be captured by a hierarchy or by categories (Aissen, 2003; Silverstein, 1986). For instance, Zaenen et al. (2004) propose three levels of animacy (*human*, *other animate* and *inanimate*), which cover ten categories of noun phrases, with categories like ORG (organization), ANIM (animal) and MAC (intelligent machines such as robots) categorised as *other animate*. Bowman and Chopra (2012) report results for animacy defined both this way and with the categories collapsed to a binary (*animate, inanimate*) definition.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 55–60,
Seattle, Washington, USA, 18-21 October 2013.

2.2 Methods for Animacy Detection

Evans and Orăsan (2000) propose a rule-based system based on the WordNet taxonomy (Fellbaum, 1998). Each synset is ascribed a binary animacy label based on its unique beginner. A given noun is then associated with the fraction of its animate synsets (where all synsets are taken to be animate or inanimate) and one minus that fraction, similarly for a given verb. Animacy is then ascribed by applying a series of rules imposing thresholds on those fractions, together with rules (and a gazetteer) to detect names and acronyms, and a rule triggered by the occurrence of *who*, or reflexives, in the NP. In later work, Orăsan and Evans (2007) extend the algorithm by propagating animacy labels in the WordNet graph using a chi-squared test, and then apply a k-nearest neighbor classifier based on four lexical features. In their work, the only context used was the animacy of the verb in the NP, for heads of subject NPs (e.g., the subject of *eat* is typically animate). Øvrelid (2009) and Bowman and Chopra (2012) extend this idea by using dependency relations to generate features for their classifier, enabled by corpora created by Zaenen et al. (2004). In another approach, Ji and Lin (2009) apply a simple "relative-pronoun" pattern to the Google n-gram corpus (Brants and Franz, 2006) to assign animacy (see the List model in Section 5 for details). Although the animacy decision is again context-independent, such a list provides a strong baseline and thus benefit applications like anaphora resolution (Lee et al., 2013).

3 The Task

We adopt a definition of animacy closest to the binary version in Bowman and Chopra (2012): we define an entity to be animate if it is alive and has the ability to move under its own will. We adopt this simple definition because it fits well with the common meaning and is therefore less error prone, both in terms of incorporation into higher level models, and for labeling (Orăsan and Evans (2007) report that the labeling of animacy tuned for anaphora proved challenging for the judges). We also apply the label to single noun tokens where possible: the only exceptions are compound names ("*Sarah Jones*") which are treated as single units. Thus, for example, "*puppy food*" is treated as two words, with *puppy* animate and *food* inanimate. A more complete definition would extend this to all noun phrases, so that *puppy food* as a unit would be inanimate, a notion we plan to revisit in future work. Note that even this simple definition presents challenges, so that a binary label must be applied depending on the predominant meaning. In "*A plate of chicken*," *chicken* is treated as inanimate since it refers to food. In "*Caruso (1873-1921) is considered one of the world's best opera singers. He...*," although at the time of writing clearly *Caruso* was not alive, the token is still treated as animate here because the subsequent writing refers to a live person.

4 The Data

We used the MC160 dataset, which is a subset of the MCTest dataset and which is composed of 160 grade level reading comprehension stories generated using crowd sourcing (Richardson et al., 2013). Workers were asked to write a short story (typically less than 300 words) with a target audience of 5 to 7 year olds. The available vocabulary was limited to approximately 8000 words, to model the reading ability of a first or second grader. We labeled this data for animacy using the definition given above. The first 100 of the 160 stories were used as the training set, and the remaining 60 were used for the test set. These animacy labels will be made available on the web site for MCTest (Richardson et al., 2013).

5 The Models

Since one of our key goals is interpretability we chose to use an ensemble of simple voting models. Each model is able to vote for the categories *Animal*, *Person*, *Inanimate*, or to abstain. The distinction between *Animal* and *Person* is only used when we combine votes, where *Animal* and *Person* votes appear as distinct inputs for the final voting combination model. Some voters do not distinguish between *Person* and *Animal*, and vote for *Animate* or *Inanimate*. Our models are:

List: The n-gram list method from (Ji and Lin, 2009). Here, the frequencies with which the relative pronouns *who*, *where*, *when*, and *which* occur are considered. Any noun followed most frequently by *who* is classified as *Animate*, and any other noun

in the list is classified as *Inanimate*. This voter abstains when the noun is not present in the list.

Anaphora Design: The WordNet-based approach of Evans and Orăsan (2000).

WordNet: A simple approach using WordNet. This voter chooses *Animal* or *Person* if the unique beginner of the first synset of the noun is either of these, and *Inanimate* otherwise.

WordSim: This voter uses the contextual vector space model of Yih and Qazvinian (2012) computed using Wikipedia and LA Times data. It uses short lists of hand-chosen signal words for the categories *Animal*, *Person*, and *Inanimate* to produce a "response" of the word to each category. This response is equal to the maximum cosine similarity in the vector space of the query word to any signal word in the category. The final vote goes to the category with the highest response.

Name: We used an in-house named entity tagger. This voter can recognize some inanimate entities such as cities, but does not distinguish between people and animals, and so can only vote *Animate*, *Inanimate* or *Abstain*.

Dictionaries: We use three different dictionary sources (Simple English Wiktionary, Full English Wiktionary, and the definitions found in WordNet) with a recursive dictionary crawling algorithm. First, we fetch the first definition of the query and use a dependency tree and simple heuristics to find the head noun of the definition, ignoring qualification NPs like "piece" or "member." If this noun belongs to a list of per-category signal words, the voter stops and votes for that category. Otherwise, the voter recursively runs on the found head noun. To prevent cycling, if no prediction is made after 10 recursive lookups, the voter abstains.

Transfer: For each story, we first process each sentence and detect instances of the patterns *x am/is/was/are/were y* and *y named x*. In each of these cases, we use majority vote of the remaining voters to predict the animacy of *y* and transfer its vote to *x*, applying this label (as a vote) to all instances of *x* in the text.

The *WordSim* and *Dictionaries* voters share lists of signal words, which were chosen early in the experimental process using the training set. The signal words for the *Animal* category were *animal* and *mammal*[1]. *Person* contains *person* and *people*. Finally, *Inanimate* uses *thing*, *object*, *space*, *place*, *symbol*, *food*, *structure*, *sound*, *measure*, and *unit*.

We considered two methods for combining voters: majority voting (where the reliable *Name* voter overrides the others if it does not abstain) and a linear reweighting of votes. In the reweighting method, a feature vector is formed from the votes. Except for *WordSim*, this vector is an indicator vector of the vote – either *Animal*, *Person*, *Animate* (if the voter doesn't distinguish between animals and people), *Inanimate*, or *Abstain*.

For *Dictionaries*, the vector's non-zero component is multiplied by the number of remaining allowed recursive calls that can be performed, plus one (so that a success on the final lookup gives a 1). For example, if the third lookup finds a signal word and chooses *Animal*, then the component corresponding to *Animal* will have a value of 9.

For *WordSim*, instead of an indicator vector, the responses to each category are used, or an indicator for abstain if the model does not contain the word. If the word is in the model, a second vector is appended containing the ratio of the maximum response to the second-largest response in the component for the maximum response category. These per-voter feature vectors are concatenated to form a 35 dimensional vector, and a linear SVM is trained to obtain the weights for combining the votes.

6 Results

We used the POS tagger in MSR SPLAT (Quirk et al., 2012) to extract nouns from the stories in the MC160 dataset and used these as labeled examples for the SVM. This resulted in 5,120 extracted nouns in the 100 training stories and 3,009 in the 60 test stories. We use five-fold cross-validation on the training set to select the SVM parameters. 57.2% of the training examples were inanimate, as were 58.1% of the test examples.

Table 1 gives the test accuracy of each voter. *List*

[1]This was found to work well given typical dictionary definitions despite the fact that people are also mammals.

List	Anaphora	WNet	WSim	Dict	Name
84.6	77.1	78.8	57.6	74.3	16.0

Table 1: Accuracy of various individual voters on the test set. Abstentions are counted as errors. Note that *Transfer* depends on a secondary source for classification, and is therefore not listed here.

	Majority	SVM
N+WN+D+WS+AD+L	87.7	95.0
N+WN+WS	80.1	95.0
N+WN+D+WS+AD+L+T	87.4	95.0
N+WN+D+WS	86.4	94.8
N+WN+WS+AD+L	86.5	94.7
N+WN+D+WS+T	86.8	94.0
N+WN+D	86.1	93.7
N+WN	89.3	93.0
N+D	82.6	93.0
N+AD	87.6	89.4
N+L	85.4	88.9

Table 2: Accuracy of various combinations of voters among Name (N), Anaphora Design (AD), List (L), WordNet (WN), WordSim (WS), Dictionary (D), and Transfer (T) under majority voting and SVM schemes. Bold indicates a statistically significant difference over the next lower bolded entry with $p < 0.01$, for the SVM.

comes out on top when taken alone, but we see in later results that it is less critical when used with other voters. *Name* performs poorly on its own, but later we will see that it is a very accurate voter which frequently abstains.

Table 2 gives the test performance of various combinations of voters, both under majority vote and reweighting. Statistical significance was tested using a paired t-test, and bold indicates a method was significant over the next lower bold line with p value $p < 0.01$. We see a very large gain from the SVM reweighting: 14.9 points in the case of *Name+WordNet+WordSim*.

In Table 3, we show the results of ablation experiments on the voters. We see that the most valuable sources of information are *WordSim* and *Dictionaries*.

Finally, in Table 4, we show a breakdown of which voters cause the most errors, for the majority vote system. In this table, we considered only "final errors," i.e. errors that the entire system makes. Over all such errors, we counted the number of times

	Majority	SVM
WordSim	87.6	93.7
SimpleWikt (dict)	87.3	94.1
FullWikt (dict)	86.4	94.3
Dict	87.4	94.5
Name	86.6	94.7
List	86.4	94.8
WordNet (dict)	88.7	94.8
WordNet	87.5	94.9
Anaphora Design	88.6	94.9
Transfer	87.7	95.0

Table 3: Test accuracy when leaving out various voters, using both majority vote and and reweighting. Bold indicates statistical significance over the next lower bold line with $p < 0.01$.

each voter chose incorrectly, giving a count of how many times each voter contributed to a final error. We see that the *Anaphora Design* system has the largest number of errors on both train and test sets. After this, *WordNet*, *List*, and *WordNet (dict)* are also large sources of error. On the other hand, *Name* and *WordSim* have very few errors, indicating high reliability. The table also gives the number of *critical errors*, where the voter selected the wrong category *and* was a deciding vote (that is, when changing its vote would have resulted in a correct overall classification). We see a similar pattern here, with *Anaphora Design* causing the most errors and *WordSim* and *Name* among the most reliable. We included *Anaphora Design* even though it uses a different definition of animacy, to determine if its vote was nevertheless valuable.

Error tables such as these show how voting models are more interpretable *and therefore correctable* compared to more complex learned models. The tables indicate the largest sources of error and suggest changes that could be made to increase accuracy. For example, we could make significant gains by improving *WordNet*, *WordNet (dictionary)*, or *List*, whereas there is relatively little reason to adjust *WordSim* or *Name*.

7 Conclusions

We have shown that linear combinations of voting models can give animacy detection rates in the mid 90% range. This is well above the accuracy found

	Errors		Critical	
	Train	Test	Train	Test
Anaphora Design	555	266	117	76
WordNet	480	228	50	45
List	435	195	94	45
Transfer	410	237	54	58
WordNet (dict)	385	194	84	65
SimpleWikt (dict)	175	111	39	16
FullWikt (dict)	158	67	1	5
WordSim	107	89	11	19
Name	71	55	27	19

Table 4: *Errors* column: number of errors on train and test where a source voted incorrectly, and was thus at least in part responsible for an error of the overall system. *Critical* column: number of errors on train and test where a source voted incorrectly, and in addition cast a deciding vote. Results are for majority vote.

by using the n-gram method of (Ji and Lin, 2009), which is used as an animacy detection component in other systems. In this sense the work presented here improves upon the state of the art, but there are caveats, since other workers define animacy differently and so a direct comparison with their work is not possible. Our method has the added advantage of interpretability, which we believe will be useful when using it as a component in a larger system.

Acknowledgments

We wish to thank Andrzej Pastusiak for his help with the labeling tool.

References

Judith Aissen. 2003. Differential object marking: Iconicity vs. economy. *Natural Language & Linguistic Theory*, 21(3):435–483.

Samuel Bowman and Harshit Chopra. 2012. Automatic animacy classification. In *Proceedings of the NAACL-HLT 2012 Student Research Workshop*.

Thorsten Brants and Alex Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium.

Felice Dell'Orletta, Alessandro Lenci, Simonetta Montemagni, and Vito Pirrelli. 2005. Climbing the path to grammar: a maximum entropy model of subject/object learning. In *Proceedings of the Workshop on Psychocomputational Models of Human Language Acquisition*, PMHLA '05, pages 72–81, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15.

Richard Evans and Constantin Orăsan. 2000. Improving anaphora resolution by identifying animate entities in texts. In *Proceedings of the Discourse Anaphora and Reference Resolution Conference (DAARC2000)*, pages 154–162.

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

Heng Ji and Dekang Lin. 2009. Gender and animacy knowledge discovery from Web-scale n-grams for unsupervised person mention detection. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*, pages 220–229, Hong Kong, December. City University of Hong Kong.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

Constantin Orăsan and Richard J. Evans. 2007. NP animacy identification for anaphora resolution. *Journal of Artificial Intelligence Research (JAIR)*, 29:79–103.

Lilja Øvrelid and Joakim Nivre. 2007. When word order and part-of-speech tags are not enough – Swedish dependency parsing with rich linguistic features. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 447–451.

Lilja Øvrelid. 2009. Empirical evaluations of animacy annotation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wen-tau Yih, Colin Cherry, and Lucy Vanderwende. 2012. MSR SPLAT, a language analysis toolkit. In *Proceedings of the Demonstration Session at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 21–24, Montréal, Canada, June. Association for Computational Linguistics.

Matthew Richardson, Chris Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Michael Silverstein. 1986. Hierarchy of features and ergativity. In P. Muysken and H. van Riemsdijk, editors, *Features and Projections*, pages 163–232. Foris Publications Holland.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space

models. In *Proceedings of NAACL-HLT*, pages 616–620, Montréal, Canada, June.

Annie Zaenen, Jean Carletta, Gregory Garretson, Joan Bresnan, Andrew Koontz-Garboden, Tatiana Nikitina, M. Catherine O'Connor, and Tom Wasow. 2004. Animacy encoding in English: Why and how. In Bonnie Webber and Donna K. Byron, editors, *ACL 2004 Workshop on Discourse Annotation*, pages 118–125, Barcelona, Spain, July. Association for Computational Linguistics.

A Log-Linear Model for Unsupervised Text Normalization

Yi Yang
School of Interactive Computing
Georgia Institute of Technology
yiyang@gatech.edu

Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
jacobe@gatech.edu

Abstract

We present a unified unsupervised statistical model for text normalization. The relationship between standard and non-standard tokens is characterized by a log-linear model, permitting arbitrary features. The weights of these features are trained in a maximum-likelihood framework, employing a novel sequential Monte Carlo training algorithm to overcome the large label space, which would be impractical for traditional dynamic programming solutions. This model is implemented in a normalization system called UNLOL, which achieves the best known results on two normalization datasets, outperforming more complex systems. We use the output of UNLOL to automatically normalize a large corpus of social media text, revealing a set of coherent orthographic styles that underlie online language variation.

1 Introduction

Social media language can differ substantially from other written text. Many of the attempts to characterize and overcome this variation have focused on *normalization*: transforming social media language into text that better matches standard datasets (Sproat et al., 2001; Liu et al., 2011). Because there is little available training data, and because social media language changes rapidly (Eisenstein, 2013b), fully supervised training is generally not considered appropriate for this task. However, due to the extremely high-dimensional output space — arbitrary sequences of words across the vocabulary — it is a very challenging problem for unsupervised learning. Perhaps it is for these reasons that the most successful systems are pipeline architectures that cobble together a diverse array of techniques and resources, including statistical language models, dependency parsers, string edit distances, off-the-shelf spellcheckers, and curated slang dictionaries (Liu et al., 2011; Han and Baldwin, 2011; Han et al., 2013).

We propose a different approach, performing normalization in a maximum-likelihood framework. There are two main sources of information to be exploited: local context, and surface similarity between the observed strings and normalization candidates. We treat the local context using standard language modeling techniques; we treat string similarity with a log-linear model that includes features for both surface similarity and word-word pairs.

Because labeled examples of normalized text are not available, this model cannot be trained in the standard supervised fashion. Nor can we apply dynamic programming techniques for unsupervised training of locally-normalized conditional models (Berg-Kirkpatrick et al., 2010), as their complexity is quadratic in the size of label space; in normalization, the label space is the vocabulary itself, with at least 10^4 elements. Instead, we present a new training approach using Monte Carlo techniques to compute an approximate gradient on the feature weights. This training method may be applicable in other unsupervised learning problems with a large label space.

This model is implemented in a normalization system called UNLOL (**u**nsupervised **n**ormalization in a **LO**g-**L**inear model). It is a lightweight proba-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 61–72,
Seattle, Washington, USA, 18-21 October 2013.

bilistic approach, relying only on a language model for the target domain; it can be adapted to new corpora text or new domains easily and quickly. Our evaluations show that UNLOL outperforms the state-of-the-art on standard normalization datasets. In addition, we demonstrate the linguistic insights that can be obtained from normalization, using UNLOL to identify classes of orthographic transformations that form coherent linguistic styles.

2 Background

The text normalization task was introduced by Sproat et al. (2001), and attained popularity in the context of SMS messages (Choudhury et al., 2007b). It has become still more salient in the era of widespread social media, particularly Twitter. Han and Baldwin (2011) formally define a normalization task for Twitter, focusing on normalizations between single tokens, and excluding multi-word tokens like `lol` (*laugh out loud*). The normalization task has been criticized by Eisenstein (2013b), who argues that it strips away important social meanings. In recent work, normalization has been shown to yield improvements for part-of-speech tagging (Han et al., 2013), parsing (Zhang et al., 2013), and machine translation (Hassan and Menezes, 2013). As we will show in Section 7, accurate automated normalization can also improve our understanding of the nature of social media language.

Supervised methods Early work on normalization focused on labeled SMS datasets, using approaches such as noisy-channel modeling (Choudhury et al., 2007a) and machine translation (Aw et al., 2006), as well as hybrid combinations of spelling correction and speech recognition (Kobus et al., 2008; Beaufort et al., 2010). This work sought to balance language models (favoring words that fit in context) with transformation models (favoring words that are similar to the observed text). Our approach can also be seen as a noisy channel model, but unlike this prior work, no labeled data is required.

Unsupervised methods Cook and Stevenson (2009) manually identify several word formation types within a noisy channel framework. They parametrize each formation type with a small number of scalar values, so that all legal transformations of a given type are equally likely. The scalar parameters are then estimated using expectation maximization. This work stands apart from most of the other unsupervised models, which are pipelines.

Contractor et al. (2010) use string edit distance to identify closely-related candidate orthographic forms and then decode the message using a language model. Gouws et al. (2011) refine this approach by mining an "exception dictionary" of strongly-associated word pairs such as *you*/`u`. Like Contractor et al. (2010), we apply string edit distance, and like Gouws et al. (2011), we capture strongly related word pairs. However, rather than applying these properties as filtering steps in a pipeline, we add them as features in a unified log-linear model.

Recent approaches have sought to improve accuracy by bringing more external resources and complex architectures to bear. Han and Baldwin (2011) begin with a set of string similarity metrics, and then apply dependency parsing to identify contextually-similar words. Liu et al. (2011) extract noisy training pairs from the search snippets that result from carefully designed queries to Google, and then train a conditional random field (Lafferty et al., 2001) to estimate a character-based translation model. They later extend this work by adding a model of visual priming, an off-the-shelf spell-checker, and local context (Liu et al., 2012a). Hassan and Menezes (2013) use a random walk framework to capture contextual similarity, which they then interpolate with an edit distance metric. Rather than seeking additional external resources or designing more complex metrics of context and similarity, we propose a unified statistical model, which learns feature weights in a maximum-likelihood framework.

3 Approach

Our approach is motivated by the following criteria:

- **Unsupervised**. We want to be able to train a model without labeled data. At present, labeled data for Twitter normalization is available only in small quantities. Moreover, as social media language is undergoing rapid change (Eisenstein, 2013b), labeled datasets may become stale and increasingly ill-suited to new spellings and words.

- **Low-resource**. Other unsupervised approaches take advantage of resources such as slang dictionaries and spell checkers (Han and Baldwin, 2011; Liu et al., 2011). Resources that characterize the current state of internet language risk becoming outdated; in this paper we investigate whether high-quality normalization is possible without any such resources.

- **Featurized**. The relationship between any pair of words can be characterized in a number of different ways, ranging from simple character-level rules (e.g., *going*/`goin`) to larger substitutions (e.g., *someone*/`sum1`), and even to patterns that are lexically restricted (e.g., *you*/`u`, *to*/`2`). For these reasons, we seek a model that permits many overlapping features to describe candidate word pairs. These features may include simple string edit distance metrics, as well as lexical features that memorize specific pairs of standard and nonstandard words.

- **Context-driven**. Learning potentially arbitrary word-to-word transformations without supervision would be impossible without the strong additional cue of local context. For example, in the phrase

  `give me suttin to believe in`,

  even a reader who has never before seen the word `suttin` may recognize it as a phonetic transcription of *something*. The relatively high string edit distance is overcome by the strong contextual preference for the word *something* over orthographically closer alternatives such as *button* or *suiting*. We can apply an arbitrary target language model, leveraging large amounts of unlabeled data and catering to the desired linguistic characteristics of the normalized content.

- **Holistic**. While several prior approaches — such as normalization dictionaries — operate at the token level, our approach reasons over the scope of the entire message. The necessity for such holistic, joint inference and learning can be seen by changing the example above to:

  `gimme suttin 2 beleive innnn`.

  None of these tokens are standard (except `2`, which appears in a nonstandard sense here), so without joint inference, it would not be possible to use context to help normalize `suttin`. Only by jointly reasoning over the entire message can we obtain the correct normalization.

These desiderata point towards a featurized sequence model, which must be trained without labeled examples. While there is prior work on training sequence models without supervision (Smith and Eisner, 2005; Berg-Kirkpatrick et al., 2010), there is an additional complication not faced by models for tasks such as part-of-speech tagging and named entity recognition: the potential label space of standard words is large, on the order of at least 10^4. Naive application of Viterbi decoding — which is a component of training for both Contrastive Estimation (Smith and Eisner, 2005) and the locally-normalized sequence labeling model of Berg-Kirkpatrick et al. (2010) — will be stymied by Viterbi's quadratic complexity in the dimension of the label space. While various pruning heuristics may be applied, we instead look to Sequential Monte Carlo (SMC), a randomized algorithm which approximates the necessary feature expectations through weighted samples.

4 Model

Given a set of source-language sentences $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots\}$ (e.g., Tweets), our goal is to transduce them into target-language sentences $T = \{\mathbf{t}_1, \mathbf{t}_2, \ldots\}$ (standard English). We are given a target language model $P(\mathbf{t})$, which can be estimated from some large set of unlabeled target-language sentences. We denote the vocabularies of source language and target language as ν_S and ν_T respectively.

We define a log-linear model that scores source and target strings, with the form

$$P(\mathbf{s}|\mathbf{t};\theta) \propto \exp\left(\theta^{\mathsf{T}}\mathbf{f}(\mathbf{s},\mathbf{t})\right). \quad (1)$$

The desired conditional probability $P(\mathbf{t}|\mathbf{s})$ can be obtained by combining this model with the target language model, $P(\mathbf{t}|\mathbf{s}) \propto P(\mathbf{s}|\mathbf{t};\theta)P(\mathbf{t})$. Since no labeled data is available, the parameters θ must be estimated by maximizing the log-likelihood of the source-language data. We define the log-likelihood

$\ell_\theta(\mathbf{s})$ for a source-language sentence $\mathbf{s}$ as follows:

$$\ell_\theta(\mathbf{s}) = \log P(\mathbf{s}) = \log \sum_{\mathbf{t}} P(\mathbf{s}|\mathbf{t};\theta)P(\mathbf{t})$$

We would like to maximize this objective by making gradient-based updates.

$$\begin{aligned}\frac{\partial \ell_\theta(\mathbf{s})}{\partial \theta} &= \frac{1}{P(\mathbf{s})}\sum_{\mathbf{t}} P(\mathbf{t})\frac{\partial}{\partial\theta}P(\mathbf{s}|\mathbf{t};\theta) \\ &= \sum_{\mathbf{t}} P(\mathbf{t}|\mathbf{s})\left(\mathbf{f}(\mathbf{s},\mathbf{t}) - \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{t})\mathbf{f}(\mathbf{s}',\mathbf{t})\right) \\ &= E_{\mathbf{t}|\mathbf{s}}[\mathbf{f}(\mathbf{s},\mathbf{t}) - E_{\mathbf{s}'|\mathbf{t}}[\mathbf{f}(\mathbf{s}',\mathbf{t})]]\end{aligned} \tag{2}$$

We are left with a difference in expected feature counts, as is typical in log-linear models. However, unlike the supervised case, here *both* terms are expectations: the outer expectation is over all target sequences (given the observed source sequence), and the nested expectation is over all source sequences, given the target sequence. As the space of possible target sequences $\mathbf{t}$ grows exponentially in the length of the source sequence, it will not be practical to compute this expectation directly.

Dynamic programming is the typical solution for computing feature expectations, and can be applied to sequence models when the feature function decomposes locally. There are two reasons this will not work in our case. First, while the forward-backward algorithm would enable us to compute $E_{\mathbf{t}|\mathbf{s}}$, it would not give us the nested expectation $E_{\mathbf{t}|\mathbf{s}}[E_{\mathbf{s}'|\mathbf{t}}]$; this is the classic challenge in training globally-normalized log-linear models without labeled data (Smith and Eisner, 2005). Second, both forward-backward and the Viterbi algorithm have time complexity that is quadratic in the dimension of the label space, at least 10^4 or 10^5. As we will show, Sequential Monte Carlo (SMC) algorithms have a number of advantages in this setting: they permit the efficient computation of both the outer and inner expectations, they are trivially parallelizable, and the number of samples provides an intuitive tuning tradeoff between accuracy and speed.

4.1 Sequential Monte Carlo approximation

Sequential Monte Carlo algorithms are a class of sampling-based algorithms in which latent variables are sampled sequentially (Cappe et al., 2007). They are particularly well-suited to sequence models, though they can be applied more broadly. SMC algorithms maintain a set of weighted hypotheses; the weights correspond to probabilities, and in our case, the hypotheses correspond to target language word sequences. Specifically, we approximate the conditional probability,

$$P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n}) \approx \sum_{k=1}^{K} \omega_n^k \delta_{\mathbf{t}_{1:n}^k}(\mathbf{t}_{1:n}),$$

where ω_n^k is the normalized weight of sample k at word n ($\tilde{\omega}_n^k$ is the unnormalized weight), and $\delta_{\mathbf{t}_{1:n}^k}$ is a delta function centered at $\mathbf{t}_{1:n}^k$.

At each step, and for each hypothesis k, a new target word is sampled from a *proposal distribution*, and the weight of the hypothesis is then updated. We maintain feature counts for each hypothesis, and approximate the expectation by taking a weighted average using the hypothesis weights. The proposal distribution will be described in detail later.

We make a Markov assumption, so that the emission probability $P(\mathbf{s}|\mathbf{t})$ decomposes across the elements of the sentence $P(\mathbf{s}|\mathbf{t}) = \prod_n^N P(s_n|t_n)$. This means that the feature functions $\mathbf{f}(\mathbf{s},\mathbf{t})$ must decompose on each $\langle s_n, t_n\rangle$ pair. We can then rewrite (1) as

$$P(\mathbf{s}|\mathbf{t};\theta) = \prod_n^N \frac{\exp\left(\theta^\mathsf{T}\mathbf{f}(s_n,t_n)\right)}{Z(t_n)} \tag{3}$$

$$Z(t_n) = \sum_s \exp\left(\theta^\mathsf{T}\mathbf{f}(s,t_n)\right). \tag{4}$$

In addition, we assume that the target language model $P(\mathbf{t})$ can be written as an N-gram language model, $P(\mathbf{t}) = \prod_n P(t_n|t_{n-1},\ldots t_{n-k+1})$. With these assumptions, we can view normalization as a finite state-space model in which the target language model defines the prior distribution of the process and Equation 3 defines the likelihood function. We are able to compute the the posterior probability $P(\mathbf{t}|\mathbf{s})$ using *sequential importance sampling*, a member of the SMC family.

The crucial idea in sequential importance sampling is to update the hypotheses $\mathbf{t}_{1:n}^k$ and their weights ω_n^k so that they approximate the posterior distribution at the next time step, $P(\mathbf{t}_{1:n+1}|\mathbf{s}_{1:n+1})$.

Assuming the proposal distribution has the form $Q(\mathbf{t}^k_{1:n}|\mathbf{s}_{1:n})$, the importance weights are given by

$$\omega^k_n \propto \frac{P(\mathbf{t}^k_{1:n}|\mathbf{s}_{1:n})}{Q(\mathbf{t}^k_{1:n}|\mathbf{s}_{1:n})} \tag{5}$$

In order to update the hypotheses recursively, we rewrite $P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n})$ as:

$$\begin{aligned}P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n}) &= \frac{P(s_n|\mathbf{t}_{1:n},\mathbf{s}_{1:n-1})P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n-1})}{P(s_n|\mathbf{s}_{1:n-1})}\\ &= \frac{P(s_n|t_n)P(t_n|\mathbf{t}_{1:n-1},\mathbf{s}_{1:n-1})P(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1})}{P(s_n|\mathbf{s}_{1:n-1})}\\ &\propto P(s_n|t_n)P(t_n|t_{n-1})P(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1}),\end{aligned}$$

assuming a bigram language model. We further assume the proposal distribution Q can be factored as:

$$\begin{aligned}Q(\mathbf{t}_{1:n}|\mathbf{s}_{1:n}) &= Q(t_n|\mathbf{t}_{1:n-1},\mathbf{s}_{1:n})Q(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1})\\ &= Q(t_n|t_{n-1},s_n)Q(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1}).\end{aligned} \tag{6}$$

Then the unnormalized importance weights simplify to a recurrence:

$$\tilde{\omega}^k_n = \frac{P(s_n|t^k_n)P(t^k_n|t^k_{n-1})P(\mathbf{t}^k_{1:n-1}|\mathbf{s}_{1:n-1})}{Q(t_n|t_{n-1},s_n)Q(\mathbf{t}^k_{1:n-1}|\mathbf{s}_{1:n-1})} \tag{7}$$

$$= \omega^k_{n-1}\frac{P(s_n|t^k_n)P(t^k_n|t^k_{n-1})}{Q(t_n|t_{n-1},s_n)} \tag{8}$$

Therefore, we can approximate the posterior distribution $P(t_n|\mathbf{s}_{1:n}) \approx \sum_{k=1}^K \omega^k_n \delta_{t^k_n}(t_n)$, and compute the outer expectation as follows:

$$E_{\mathbf{t}|\mathbf{s}}[\mathbf{f}(\mathbf{s},\mathbf{t})] = \sum_{k=1}^K \omega^k_N \sum_{n=1}^N \mathbf{f}(s_n, t^k_n) \tag{9}$$

We compute the nested expectation using a non-sequential Monte Carlo approximation, assuming we can draw $s^{\ell,k} \sim P(s|t^k_n)$.

$$E_{\mathbf{s}|\mathbf{t}^k}[\mathbf{f}(\mathbf{s},\mathbf{t}^k)] = \frac{1}{L}\sum_{n=1}^N \sum_{\ell=1}^L \mathbf{f}(s^{\ell,k}_n, t^k_n)$$

This gives the overall gradient computation:

$$\begin{aligned}E_{\mathbf{t}|\mathbf{s}}[\mathbf{f}(\mathbf{s},\mathbf{t}) - E_{\mathbf{s}'|\mathbf{t}}[\mathbf{f}(\mathbf{s}',\mathbf{t})]] &= \frac{1}{\sum_{k=1}^K \tilde{\omega}^k_N}\sum_{k=1}^K \tilde{\omega}^k_N\\ &\times \sum_{n=1}^N \left(\mathbf{f}(s_n,t^k_n) - \frac{1}{L}\sum_{\ell=1}^L \mathbf{f}(s^{\ell,k}_n, t^k_n)\right)\end{aligned} \tag{10}$$

where we sample t^k_n and update ω^k_n while moving from left-to-right, and sample $s^{\ell,k}_n$ at each n. Note that although the sequential importance sampler moves left-to-right like a filter, we use only the final weights ω_N to compute the expectation. Thus, the resulting expectation is based on the distribution $P(\mathbf{s}_{1:N}|\mathbf{t}_{1:N})$, so that no backwards "smoothing" pass (Godsill et al., 2004) is needed to eliminate bias. Other applications of sequential Monte Carlo make use of resampling (Cappe et al., 2007) to avoid degeneration of the hypothesis weights, but we found this to be unnecessary due to the short length of Twitter messages.

4.2 Proposal distribution

The major computational challenge for dynamic programming approaches to normalization is the large label space, equal to the size of the target vocabulary. It may appear that all we have gained by applying sequential Monte Carlo is to convert a computational problem into a statistical one: a naive sampling approach will have little hope of finding the small high-probability region of the high-dimensional label space. However, sequential importance sampling allows us to address this issue through the proposal distribution, from which we sample the candidate words t_n. Careful design of the proposal distribution can guide sampling towards the high-probability space. In the asymptotic limit of an infinite number of samples, any non-pathological proposal distribution will ultimately arrive at the desired estimate, but a good proposal distribution can greatly reduce the number of samples needed.

Doucet et al. (2001) note that the optimal proposal — which minimizes the variance of the importance weights conditional on $\mathbf{t}_{1:n-1}$ and $\mathbf{s}_{1:n}$ — has the following form:

$$Q(t^k_n|s_n,t^k_{n-1}) = \frac{P(s_n|t^k_n)P(t^k_n|t^k_{n-1})}{\sum_{t'}P(s_n|t')P(t'|t^k_{n-1})} \tag{11}$$

Sampling from this proposal requires computing the normalized distribution $P(s_n|t_n^k)$; similarly, the update of the hypothesis weights (Equation 8) requires the calculation of Q in its normalized form. In each case, the total cost is the product of the vocabulary sizes, $\mathcal{O}(\#|\nu_T|\#|\nu_S|)$, which is not tractable as the vocabularies become large.

In low-dimensional settings, a convenient solution is to set the proposal distribution equal to the transition distribution, $Q(t_n^k|s_n, t_{n-1}^k) = P(t_n^k|t_{n-1}^k, \ldots, t_{n-k+1}^k)$. This choice is called the "bootstrap filter," and it has the advantage that the weights $\omega^{(k)}$ are exactly identical to the product of emission likelihoods $\prod_n P(s_n|t_n^k)$. The complexity of computing the hypothesis weights is thus $\mathcal{O}(\#|\nu_S|)$. However, because this proposal ignores the emission likelihood, the bootstrap filter has very little hope of finding a high-probability sample in high-entropy contexts.

We strike a middle ground between efficiency and accuracy, using a proposal distribution that is closely related to the overall likelihood, yet is tractable to sample and compute:

$$\begin{aligned} Q(t_n^k|s_n, t_{n-1}^k) &\stackrel{def}{=} \\ &\frac{P(s_n|t_n^k)Z(t_n^k)P(t_n^k|t_{n-1}^k)}{\sum_{t'} P(s_n|t')Z(t')P(t'|t_{n-1}^k)} \\ =&\frac{\exp\left(\theta^\mathsf{T}\mathbf{f}(s_n, t_n)\right) P(t_n^k|t_{n-1}^k)}{\sum_{t'} \exp\left(\theta^\mathsf{T}\mathbf{f}(s_n, t')\right) P(t'|t_{n-1}^k)} \end{aligned} \quad (12)$$

Here, we simply replace the likelihood distribution in (11) by its unnormalized version.

To update the unnormalized hypothesis weights $\tilde{\omega}_n^k$, we have

$$\tilde{\omega}_n^k = \omega_{n-1}^k \frac{\sum_{t'} \exp\left(\theta^\mathsf{T}\mathbf{f}(s_n, t')\right) P(t'|t_{n-1}^k)}{Z(t_n^k)} \quad (13)$$

The numerator requires summing over all elements in ν_T and the denominator $Z(t_n^k)$ requires summing over all elements in ν_S, for a total cost of $\mathcal{O}(\#|\nu_T| + \#|\nu_S|)$.

4.3 Decoding

Given an input source sentence $\mathbf{s}$, the decoding problem is to find a target sentence $\mathbf{t}$ that maximizes $P(\mathbf{t}|\mathbf{s}) \propto P(\mathbf{s}|\mathbf{t})P(\mathbf{t}) = \prod_n^N P(s_n|t_n)P(t_n|t_{n-1})$.

Feature name	Description
word-word pair	A set of binary features for each source/target word pair $\langle s, t\rangle$
string similarity	A set of binary features indicating whether s is one of the top N string similar non-standard words of t, for $N \in \{5, 10, 25, 50, 100, 250, 500, 1000\}$

Table 1: The feature set for our log-linear model

As with learning, we cannot apply the usual dynamic programming algorithm (Viterbi), because of its quadratic cost in the size of the target language vocabulary. This must be multiplied by the cost of computing the normalized probability $P(s_n|t_n)$, resulting in a prohibitive time complexity of $\mathcal{O}(\#|\nu_S|\#|\nu_T|^2 N)$.

We consider two approximate decoding algorithms. The first is to simply apply the proposal distribution, with linear complexity in the size of the two vocabularies. However, this decoder is not identical to $P(\mathbf{t}|\mathbf{s})$, because of the extra factor of $Z(t)$ in the numerator. Alternatively, we can apply the proposal distribution for selecting target word candidates, then apply the Viterbi algorithm only within these candidates. The total cost is $\mathcal{O}(\#|\nu_S|T^2 N)$, where T is the number of target word candidates we consider; this will asymptotically approach $P(\mathbf{t}|\mathbf{s})$ as $T \to \#|\nu_T|$. Our evaluations use the more expensive proposal+Viterbi decoding, but accuracy with the more efficient proposal-based decoding is very similar.

4.4 Features

Our system uses the feature types described in Table 1. The word pair features are designed to capture lexical conventions, e.g. *you*/u. We only consider word pair features that fired during training. The string similarity features rely on the similarity function proposed by Contractor et al. (2010), which has proven effective for normalization in prior work. We bin this similarity to create binary features indicating whether a string s is in the top-N most similar strings to t; this binning yields substantial speed improvements without negatively impacting accuracy.

5 Implementation and data

The model and inference described in the previous section are implemented in a software system for normalizing text on twitter, called UNLOL: **u**nsupervised **n**ormalization in a **LO**g-**L**inear model. The final system can process roughly 10,000 Tweets per hour. We now describe some implementation details.

5.1 Normalization candidates

Most tokens in tweets do not require normalization. The question of how to identify which words are to be normalized is still an open problem. Following Han and Baldwin (2011), we build a dictionary of words which are permissible in the target domain, and make no attempt to normalize source strings that match these words. As with other comparable approaches, we are therefore unable to normalize strings like `ill` into *I'll*. Our set of "in-vocabulary" (IV) words is based on the GNU aspell dictionary (v0.60.6), containing 97,070 words. From this dictionary, we follow Liu et al. (2012a) and remove all the words with a count of less than 20 in the Edinburgh Twitter corpus (Petrović et al., 2010) — resulting in a total of 52,449 target words. All single characters except `a` and `i` are excluded, and `rt` is treated as in-vocabulary. For all in-vocabulary words, we define $P(s_n|t_n) = \delta(s_n, t_n)$, taking the value of zero when $s_n \neq t_n$. This effectively prevents our model from attempting to normalize these words.

In addition to words that are in the target vocabulary, there are many other strings that should not be normalized, such as names and multiword shortenings (e.g. *going to*/`gonna`).[1] We follow prior work and assume that the set of normalization candidates is known in advance during test set decoding (Han et al., 2013). However, the unlabeled training data has no such information. Thus, during training we attempt to normalize all tokens that (1) are not in our lexicon of IV words, and (2) are composed of letters, numbers and the apostrophe. This set includes contractions like "gonna" and "gotta", which would not appear in the test set, but are nonetheless normalized during training. For each OOV token, we conduct a pre-normalization step by reducing any repetitions of more than two letters in the nonstandard words to exactly two letters (e.g., `cooool` → `cool`).

[1] Whether multiword shortenings should be normalized is arguable, but they are outside the scope of current normalization datasets (Han and Baldwin, 2011).

5.2 Language modeling

The Kneser-Ney smoothed trigram target language model is estimated with the SRILM toolkit Stolcke (2002), using Tweets from the Edinburgh Twitter corpus that contain no OOV words besides hashtags and username mentions (following (Han et al., 2013)). We use this language model for both training and decoding. We occasionally find training contexts in which the trigram $\langle t_n, t_{n-1}, t_{n-2} \rangle$ is unobserved in the language model data; features resulting from such trigrams are not considered when computing the weight gradients.

5.3 Parameters

The Monte Carlo approximations require two parameters: the number of samples for sequential Monte Carlo (K), and the number of samples for the non-sequential sampler of the nested expectation (L, from Equation 10). The theory of Monte Carlo approximation states that the quality of the approximation should only improve as the number of samples increases; we obtained good results with $K = 10$ and $L = 1$, and found relatively little improvement by increasing these values. The number of hypotheses considered by the decoder is set to $T = 10$; again, the performance should only improve with T, as we more closely approximate full Viterbi decoding.

6 Experiments

Datasets We use two existing labeled Twitter datasets to evaluate our approach. The first dataset — which we call LWWL11, based on the names of its authors Liu et al. (2011) — contains 3,802 individual "nonstandard" words (i.e., words that are not in the target vocabulary) and their normalized forms. The rest of the message in which the words is appear is not available. As this corpus does not provide linguistic context, its decoding must use a unigram target language model. The second dataset — which is called LexNorm1.1 by its authors Han and Baldwin (2011) — contains 549 complete tweets with 1,184 nonstandard tokens (558 unique word types).

Method	Dataset	Precision	Recall	F-measure
(Liu et al. 2011)	LMML11	68.88	68.88	68.88
(Liu et al. 2012)		69.81	69.81	69.81
UNLOL		**73.04**	**73.04**	**73.04**
(Han and Baldwin, 2011)	LexNorm 1.1	75.30	75.30	75.30
(Liu et al. 2012)		84.13	78.38	81.15
(Hassan et al. 2013)		**85.37**	56.4	69.93
UNLOL		82.09	**82.09**	**82.09**
UNLOL	LexNorm 1.2	82.06	82.06	82.06

Table 2: Empirical results

In this corpus, we can decode with a trigram language model.

Close analysis of LexNorm1.1 revealed some inconsistencies in annotation (for example, `y'all` and `2` are sometimes normalized to *you* and *to*, but are left unnormalized in other cases). In addition, several annotations disagree with existing resources on internet language and dialectal English. For example, `smh` is normalized to *somehow* in LexNorm1.1, but `internetslang.com` and `urbandictionary.com` assert that it stands for *shake my head*, and this is evident from examples such as `smh at this girl`. Similarly, `finna` is normalized to *finally* in LexNorm1.1, but from the literature on African American English (Green, 2002), it corresponds to *fixing to* (e.g., `i'm finna go home`). To address these issues, we have produced a new version of this dataset, which we call LexNorm1.2 (after consulting with the creators of LexNorm1.1). LexNorm1.2 differs from version 1.1 in the annotations for 172 of the 2140 OOV words. We evaluate on LexNorm1.1 to compare with prior work, but we also present results on LexNorm1.2 in the hope that it will become standard in future work on normalization in English. The dataset is available at `http://www.cc.gatech.edu/~jeisenst/lexnorm.v1.2.tgz`.

To obtain unlabeled training data, we randomly sample 50 tweets from the Edinburgh Twitter corpus Petrović et al. (2010) for each OOV word. Some OOV words appear less than 50 times in the corpus, so we obtained more training tweets for them through the Twitter search API.

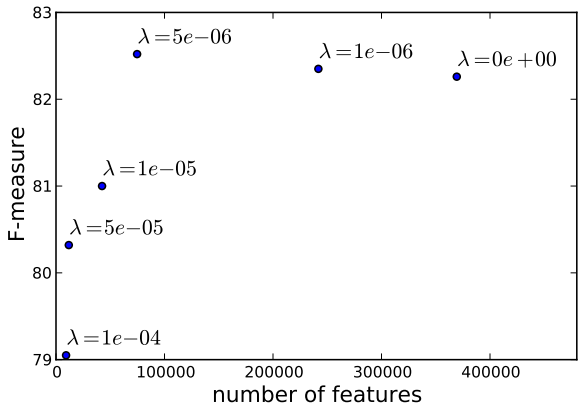
Metrics Prior work on these datasets has assumed perfect detection of words requiring normalization, and has focused on finding the correct normalization for these words (Han and Baldwin, 2011; Han et al., 2013). Recall has been defined as the proportion of words requiring normalization which are normalized correctly; precision is defined as the proportion of normalizations which are correct.

Results We run our training algorithm for two iterations (pass the training data twice). The results are presented in Table 2. Our system, UNLOL, achieves the highest published F-measure on both datasets. Performance on LexNorm1.2 is very similar to LexNorm1.1, despite the fact that roughly 8% of the examples were relabeled.

In the normalization task that we consider, the tokens to be normalized are specified in advance. This is the same task specification as in the prior work against which we compare. At test time, our system attempts normalizes all such tokens; every error is thus both a false positive and false negative, so precision equals to recall for this task; this is also true for Han and Baldwin (2011) and Liu et al. (2011).

It is possible to trade recall for precision by refusing to normalize words when the system's confidence falls below a threshold. A good setting of this threshold can improve the F-measure, but we did not report these results because we have no development set for parameter tuning.

Regularization One potential concern is that the number of non-zero feature weights will continually increase until the memory cost becomes overwhelming. Although we did not run up against mem-



λ	Dataset	F-measure	# of features
10^{-4}	LexNorm 1.1	79.05	9,281
5×10^{-5}		80.32	11,794
10^{-5}		81.00	42,466
5×10^{-6}		**82.52**	74,744
10^{-6}		82.35	241,820
0		82.26	369,366
5×10^{-6}	LexNorm 1.2	82.23	74,607

Figure 1: Effect of L1 regularization on the F-measure and the number of features with non-zero weights

ory limitations in the experiments producing the results in Table 2, this issue can be addressed through the application of L1 regularization, which produces sparse weight vectors by adding a penalty of $\lambda||\theta||_1$ to the log-likelihood. We perform online optimization of the L1-regularized log-likelihood by applying the truncated gradient method (Langford et al., 2009). We use an exponential decreasing learning rate $\eta_k = \eta_0 \alpha^{k/N}$, where k is the iteration counter and N is the size of training data. We set $\eta_0 = 1$ and $\alpha = 0.5$. Experiments were run until 300,000 training instances were observed, with a final learning rate of less than $1/32$. As shown in Figure 1, a small amount of regularization can dramatically decrease the number of active features without harming performance.

7 Analysis

We apply our normalization system to investigate the orthographic processes underlying language variation in social media. Using a dataset of 400,000 English language tweets, sampled from the month of August in each year from 2009 to 2012, we apply UNLOL to automatically normalize each token. We then treat these normalizations as labeled training data, and examine the Levenshtein alignment between the source and target tokens. This alignment gives approximate character-level transduction rules to explain each OOV token. We then examine which rules are used by each author, constructing a matrix of authors and rules.[2]

Factorization of the author-rule matrix reveals sets of rules that tend to be used together; we might call these rulesets "orthographic styles." We apply non-negative matrix factorization (Lee and Seung, 2001), which characterizes each author by a vector of k style loadings, and simultaneously constructs k style dictionaries, which each put weight on different orthographic rules. Because the loadings are constrained to be non-negative, the factorization can be seen as sparsely assigning varying amounts of each style to each author. We choose the factorization that minimizes the Frobenius norm of the reconstruction error, using the NIMFA software package (`http://nimfa.biolab.si/`).

The resulting styles are shown in Table 3, for $k = 10$; other values of k give similar overall results with more or less detail. The styles incorporate a number of linguistic phenomena, including: expressive lengthening (styles 7-9; see Brody and Diakopoulos, 2011); g- and t-dropping (style 5, see Eisenstein 2013a) ; th-stopping (style 6); and the dropping of several word-final vowels (styles 1-3). Some of these styles, such as t-dropping and th-stopping, have direct analogues in spoken language varieties (Tagliamonte and Temple, 2005; Green, 2002), while others, like expressive lengthening, seem more unique to social media. The relationships between these orthographic styles and social variables such as geography and demograph-

[2]We tried adding these rules as features and retraining the normalization system, but this hurt performance.

style	rules	examples
1. you; o-dropping	*y*/_ *ou*/_u **y*/⋆_ *o*/_	u, yu, 2day, knw, gud, yur, wud, yuh, u've, toda, everthing, everwhere, ourself
2. e-dropping, u/o	*be*/b_ *e*/_ *o*/u *e**/_⋆	b, r, luv, cum, hav, mayb, bn, remembr, btween, gunna, gud
3. a-dropping	*a*/_ **a*/⋆_ *re*/r_ *ar*/_r	r, tht, wht, yrs, bck, strt, gurantee, elementry, wr, rlly, wher, rdy, preciate, neway
4. g-dropping	*g**/_⋆ *ng*/n_ *g*/_	goin, talkin, watchin, feelin, makin
5. t-dropping	*t**/_⋆ *st*/s_ *t*/_	jus, bc, shh, wha, gota, wea, mus, firts, jes, subsistutes
6. th-stopping	*h*/_ **t*/⋆d *th*/d_ *t*/d	dat, de, skool, fone, dese, dha, shid, dhat, dat's
7. (kd)-lengthening	*i*_/id _/k _/d _*/k⋆	idk, fuckk, okk, backk, workk, badd, andd, goodd, bedd, elidgible, pidgeon
8. o-lengthening	*o*_/oo _*/o⋆ _/o	soo, noo, doo, oohh, loove, thoo, helloo
9. e-lengthening	_/i *e*_/ee _/e _*/e⋆	mee, ive, retweet, bestie, lovee, nicee, heey, likee, iphone, homie, ii, damnit
10. a-adding	_/a __/ma _/m _*/a⋆	ima, outta, needa, shoulda, woulda, mm, comming, tomm, boutt, ppreciate

Table 3: Orthographic styles induced from automatically normalized Twitter text

ics must be left to future research, but they offer a promising generalization of prior work that has focused almost exclusively on exclusively on lexical variation (Argamon et al., 2007; Eisenstein et al., 2010; Eisenstein et al., 2011), with a few exceptions for character-level features (Brody and Diakopoulos, 2011; Burger et al., 2011).

Note that style 10 is largely the result of mistaken normalizations. The tokens ima, outta, and needa all refer to multi-word expressions in standard English, and are thus outside the scope of the normalization task as defined by Han et al. (2013). UNLOL has produced incorrect single-token normalizations for these terms: *i*/ima, *out*/outta, and *need*/needa. But while these normalizations are wrong, the resulting style nonetheless captures a coherent orthographic phenomenon.

8 Conclusion

We have presented a unified, unsupervised statistical model for normalizing social media text, attaining the best reported performance on the two standard normalization datasets. The power of our approach comes from flexible modeling of word-to-word relationships through features, while exploiting contextual regularity to train the corresponding feature weights without labeled data. The primary technical challenge was overcoming the large label space of the normalization task; we accomplish this using sequential Monte Carlo. Future work may consider whether sequential Monte Carlo can offer similar advantages in other unsupervised NLP tasks. An additional benefit of our joint statistical approach is that it may be combined with other downstream language processing tasks, such as part-of-speech tagging (Gimpel et al., 2011) and named entity resolution (Liu et al., 2012b).

Acknowledgments

We thank the reviewers for thoughtful comments on our submission. This work also benefitted from discussions with Timothy Baldwin, Paul Cook, Frank Dellaert, Arnoud Doucet, Micha Elsner, and Sharon Goldwater. It was supported by NSF SOCS-1111142.

References

S. Argamon, M. Koppel, J. Pennebaker, and J. Schler. 2007. Mining the blogosphere: age, gender, and the varieties of self-expression. *First Monday*, 12(9).

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of ACL*, pages 33–40.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of ACL*, pages 770–779.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL*, pages 582–590.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs. In *Proceedings of EMNLP*.

John D. Burger, John C. Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of EMNLP*.

Olivier Cappe, Simon J. Godsill, and Eric Moulines. 2007. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, May.

M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu. 2007a. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007b. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174.

Danish Contractor, Tanveer A. Faruquie, and L. Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of COLING*, pages 189–196.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 71–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. Doucet, N.J. Gordon, and V. Krishnamurthy. 2001. Particle filters for state estimation of jump markov linear systems. *Trans. Sig. Proc.*, 49(3):613–624, March.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of EMNLP*.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of ACL*.

Jacob Eisenstein. 2013a. Phonological factors in social media writing. In *Proceedings of the NAACL Workshop on Language Analysis in Social Media*.

Jacob Eisenstein. 2013b. What to do about bad language on the internet. In *Proceedings of NAACL*, pages 359–369.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proceedings of ACL*.

Simon J. Godsill, Arnaud Doucet, and Mike West. 2004. Monte carlo smoothing for non-linear time series. In *Journal of the American Statistical Association*, pages 156–168.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11.

Lisa J. Green. 2002. *African American English: A Linguistic Introduction*. Cambridge University Press, September.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of ACL*, pages 368–378.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology*, 4(1):5.

Hany Hassan and Arul Menezes. 2013. Social text normalization using contextual graph random walks. In *Proceedings of ACL*.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing sms: are two metaphors better than one? In *Proceedings of COLING*, pages 441–448.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

John Langford, Lihong Li, and Tong Zhang. 2009. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research*, 10:777–801.

D. D. Lee and H. S. Seung. 2001. Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, pages 556–562.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of ACL*, pages 71–76.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012a. A broad-coverage normalization system for social media language. In *Proceedings of ACL*, pages 1035–1044.

Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. 2012b. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of ACL*.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. The edinburgh twitter corpus. In *Proceedings of the NAACL HLT Workshop on Computational Linguistics in a World of Social Media*, pages 25–26.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 354–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. Sproat, A.W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.

Sali Tagliamonte and Rosalind Temple. 2005. New perspectives on an ol' variable: (t,d) in british english. *Language Variation and Change*, 17:281–302, September.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of ACL*, pages 1159–1168.

Paraphrasing 4 Microblog Normalization

Wang Ling **Chris Dyer** **Alan W Black** **Isabel Trancoso**
L^{2}F Spoken Systems Lab, INESC-ID, Lisbon, Portugal
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
Instituto Superior Técnico, Lisbon, Portugal
{lingwang,cdyer,awb}@cs.cmu.edu
isabel.trancoso@inesc-id.pt

Abstract

Compared to the edited genres that have played a central role in NLP research, microblog texts use a more informal register with nonstandard lexical items, abbreviations, and free orthographic variation. When confronted with such input, conventional text analysis tools often perform poorly. Normalization — replacing orthographically or lexically idiosyncratic forms with more standard variants — can improve performance. We propose a method for learning normalization rules from machine translations of a parallel corpus of microblog messages. To validate the utility of our approach, we evaluate extrinsically, showing that normalizing English tweets and then translating improves translation quality (compared to translating unnormalized text) using three standard web translation services as well as a phrase-based translation system trained on parallel microblog data.

1 Introduction

Microblogs such as Twitter, Sina Weibo (a popular Chinese microblog service) and Facebook have received increasing attention in diverse research communities (Han and Baldwin, 2011; Hawn, 2009, *inter alia*). In contrast to traditional text domains that use carefully controlled, standardized language, microblog content is often informal, with less adherence to conventions regarding punctuation, spelling, and style, and with a higher proportion of dialect or pronouciation-derived orthography. While this diversity itself is an important resource for studying, e.g., sociolinguistic variation (Eisenstein et al., 2011; Eisenstein, 2013), it poses challenges to NLP applications developed for more formal domains. If retaining variation due to sociolinguistic or phonological factors is not crucial, **text normalization** can improve performance on downstream tasks (§2).

This paper introduces a data-driven approach to learning normalization rules by conceiving of normalization as a kind of paraphrasing and taking inspiration from the bilingual pivot approach to paraphrase detection (Bannard and Callison-Burch, 2005) and the observation that translation is an inherently "simplifying" process (Laviosa, 1998; Volansky et al., 2013). Starting from a parallel corpus of microblog messages consisting of English paired with several other languages (Ling et al., 2013), we use standard web machine translation systems to *re-translate* the non-English segment, producing $\langle$English original, English MT$\rangle$ pairs (§3). These are our normalization examples, with MT output playing the role of normalized English. Several techniques for identifying high-precision normalization rules are proposed, and we introduce a character-based normalization model to account for predictable character-level processes, like repetition and substitution (§4). We then describe our decoding procedure (§5) and show that our normalization model improve translation quality for English–Chinese microblog translation (§6).[1]

2 Why Normalize?

Consider the English tweet shown in the first row of Table 1 which contains several elements that NLP

[1]The datasets used in this paper are available from http://www.cs.cmu.edu/~lingwang/microtopia.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 73–84,
Seattle, Washington, USA, 18-21 October 2013.

Table 1: Translations of an English microblog message into Mandarin, using three web translation services.

orig.	*To DanielVeuleman yea iknw imma work on that*
MT[1]	啊iknw DanielVeuleman伊马工作，
MT[2]	DanielVeuleman 是iknw 凋谢关于工作，
MT[3]	到DanielVeuleman是的iknw imma这方面的工作

Table 2: Translations of Chinese original post to English using web-based service.

orig.	To DanielVeuleman yea iknw imma work on that
orig.	对DanielVeuleman说，是的，我知道，我正在向那方面努力
MT[1]	Right DanielVeuleman say, yes, I know, I'm Xiangna efforts
MT[2]	DanielVeuleman said, Yes, I know, I'm that hard
MT[3]	Said to DanielVeuleman, yes, I know, I'm to that effort

systems trained on edited domains may not handle well. First, it contains several nonstandard abbreviations, such as, *yea*, *iknw* and *imma* (abbreviations of *yes*, *I know* and *I am going to*). Second, there is no punctuation in the text although standard convention would dictate that it should be used. To illustrate the effect this can have, consider now the translations produced by Google Translate,[2] Microsoft Bing,[3] and Youdao,[4] shown in rows 2–4. Even with no knowledge of Chinese, it is not hard to see that all engines have produced poor translations: the abbreviation *iknw* is left translated by all engines, and *imma* is variously deleted, left untranslated, or transliterated into the meaningless sequence 伊马 (pronounced *yī mǎ*).

While normalization to a form like *To Daniel Veuleman: Yes, I know. I am going to work on that.* does indeed lose some information (information important for an analysis of sociolinguistic or phonological variation clearly goes missing), it expresses the propositional content of the original in a form that is more amenable to processing by traditional tools. Translating the normalized form with Google Translate produces 要丹尼尔*Veuleman*：是的，我知道。我打算在那工作。, which is a substantial improvement over all translations in Table 1.

3 Obtaining Normalization Examples

We want to treat normalization as a supervised learning problem akin to machine translation, and to do so, we need to obtain pairs of microblog posts and their normalized forms. While it would be possible to ask annotators to create such a corpus, it would be quite expensive to obtain large numbers of examples. In this section, we propose a method for creating normalization examples without any human annotation, by leveraging existing tools and data resources.

The English example sentence in Table 1 was selected from the **μtopia parallel corpus** (Ling et al., 2013), which consists of *self-translated* messages from Twitter and Sina Weibo (i.e., each message contains a translation of itself). Row 2 of Table 2 shows the Mandarin self-translation from the corpus. The key observation is what happens when we *automatically* translate the Mandarin version back into English. Rows 3–5 shows automatic translations from three standard web MT engines. While not perfect, the translations contain several correctly normalized subphrases. We will use such *re-translations* as a source of (noisy) normalization examples. Since such self-translations are relatively numerous on microblogs, this technique can provide a large amount of data.

Of course, to motivate this paper, we argued that NLP tools — like the very translation systems we propose to use — often fail on unnormalized input. Is this a problem? We argue that it is not for the following two reasons.

Normalization in translation. Work in translation studies has observed that translation tends to be a *generalizing* process that "smooths out" author- and work-specific idiosyncrasies (Laviosa, 1998; Volansky et al., 2013). Assuming this observation is robust, we expect that dialectal variant forms found in microblogs to be normalized in translation. Therefore, if the parallel segments in our microblog parallel corpus did indeed originate through a translation process (rather than, e.g., being generated as two independent utterances from a bilingual), we may then state the following assumption about the distribution of variant forms in a parallel segment

[2] http://translate.google.com/
[3] http://www.bing.com/translator
[4] http://fanyi.youdao.com/

$\langle \mathbf{e}, \mathbf{f} \rangle$: *if **e** contains nonstandard lexical variants, then **f** is likely to be a normalized translation using with fewer nonstandard lexical variants (and vice-versa).*

Uncorrelated orthographic variants. Any written language has the potential to make creative use of orthography: alphabetic scripts can render approximations of pronunciation variants; logographic scripts can use homophonic substitutions. However, the kinds of innovations used in particular languages will be language specific (depending on details of the phonology, lexicon, and orthography of the language). However, for language pairs that differ substantially in these dimensions, it may not always be possible (or at least easy) to preserve particular kinds of nonstandard orthographic forms in translation. Consider the (relatively common) pronoun-verb compounds like *iknw* and *imma* from our motivating example: since Chinese uses a logographic script without spaces, there is no obvious equivalent.

3.1 Variant–Normalized Parallel Corpus

For the two reasons outlined above, we argue that we will be able to translate back into English using MT, even when the underlying English part of the parallel corpus has a great deal of nonstandard content. We leverage this fact to build the normalization corpus, where the original English tweet is treated as the variant form, and the automatic translation obtained from another language is considered a potential normalization.[5]

Our process is as follows. The microblog corpus of Ling et al. (2013) contains sentence pairs extracted from Twitter and Sina Weibo, for multiple language pairs. We use all corpora that include English as one of the languages in the pair. The respective non-English side is translated into English using different translation engines. The different sets we used and the engines we used to translate are shown in Table 3. Thus, for each original English post **o**, we obtain n paraphrases $\{\mathbf{p}_i\}_{i=1}^{n}$, from n different translation engines.

[5]We additionally assume that the translation engines are trained to output more standardized data, so there will be additional normalizing effect from the machine translation system.

Table 3: Corpora Used for Paraphrasing.

Lang. Pair	Source	Segs.	MT Engines
ZH-EN	Weibo	800K	Google, Bing, Youdao
ZH-EN	Twitter	113K	Google, Bing, Youdao
AR-EN	Twitter	114K	Google, Bing
RU-EN	Twitter	119K	Google, Bing
KO-EN	Twitter	78K	Google, Bing
JA-EN	Twitter	75K	Google, Bing

3.2 Alignment and Filtering

Our parallel microblog corpus was crawled automatically and contains many misaligned sentences. To improve precision, we attempt to find the similarity between the (unnormalized) original and each of the normalizations using an alignment based on the one used in METEOR (Denkowski and Lavie, 2011), which computes the best alignment between the original tweet and each of the normalizations but modified to permit domain-specific approximate matches. To address lexical variants, we allow fuzzy word matching, that is, we allow lexically similar, such as *yea* and *yes* to be aligned (similarity is determined by the Levenshtein distance). We also perform phrasal matchings, such as *ikwn* to *i know*. To do so, we extend the alignment algorithm from word to phrasal alignments. More precisely, given the original post **o** and a candidate normalization **n**, we wish to find the optimal segmentation producing a good alignment. A segmentation $\mathbf{s} = \langle s_1, \ldots, s_{|\mathbf{s}|} \rangle$ is a sequence of segments that aligns as a block to a source word. For instance, for the sentence *yea iknw imma work on that*, one possible segmentation could be s_1 =*yea ikwn*, s_2 =*imma* and s_3 =*work on that*.

Model. We define the score of an alignment **a** and segmentation **s** in using a model that makes semi-Markov independence assumptions, similar to the work in (Bansal et al., 2011), $u(\mathbf{a}, \mathbf{s} \mid \mathbf{o}, \mathbf{n}) =$

$$\prod_{i=1}^{|\mathbf{s}|} \Big[u_e(s_i, a_i \mid \mathbf{n}) \times u_t(a_i \mid a_{i-1}) \times u_\ell(|s_i|) \Big]$$

In this model, the maximal scoring segmentation and alignment can be found using a polynomial time dynamic programming algorithm. Each segment can be aligned to any word or segment in **o**. The aligned segment for s_k is defined as a_k. For the

score of a segment correspondence $u_e(s, a \mid \mathbf{n})$, we assume that this can be estimated using the lexical similarity between segments, which we define to be $1 - \frac{L(s_k,a_k)}{\max\{|s_k|,|a_k|\}}$, where $L(x, y)$ denotes the Levenshtein distance between strings x and y, normalized by the highest possible distance between those segments.

For the alignment score u_t, we assume that the relative order of the two sequences will be mostly monotonous. Thus, we approximate u_t with the following density $pos_s(a_k) - pos_e(a_{k-1}) \sim \mathcal{N}(1, 1)$, where the pos_s is the index of the first word in the segment and pos_e the one of the last word.

After finding the Viterbi alignments, we compute the similarity measure $\tau = \frac{|A|}{|A|+|U|}$, used in (Resnik and Smith, 2003), where $|A|$ and $|U|$ are the number of words that were aligned and unaligned, respectively. In this work, we extract the pair if $\tau > 0.2$.

4 Normalization Model

From the normalization corpus, we learn a normalization model that generalizes the normalization process. That is, from the data we observe that *To DanielVeuleman yea iknw imma work on that* is normalized to *To Daniel Veuleman: yes, I know. I am going to work on that.* However, this is not useful, since the chances of the exact sentence *To DanielVeuleman yea iknw imma work on that* occurring in the data is low. We wish to learn a process to convert the original tweet into the normalized form.

There are two mechanisms that we use in our model. The first (§4.1) learns word–word and phrase–phrase mappings. That is, we wish to find that *DanielVeuleman* is normalized to *Daniel Veuleman*, that *iknw* is normalized to *I know* and that *imma* is normalized to *I am going*. These mappings are more useful, since whenever *iknw* occurs in the data, we have the option to normalize it to *I know*. The second (§4.2) learns character sequence mappings. If we look at the normalization *DanielVeuleman* to *Daniel Veuleman*, we can see that it is only applicable when the exact word *DanielVeuleman* occurs. However, we wish to learn that it is uncommon for the letters *l* and *v* to occur in the same word sequentially, so that be can add missing spaces in words that contain the *lv* character sequence, such as normalizing *phenomenalvoter* to *phenomenal voter*.

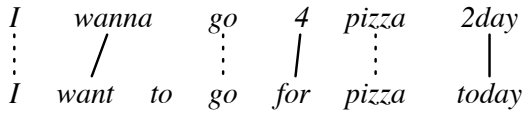


Figure 1: Variant–normalized alignment with the variant form above and the normalized form below; solid lines show potential normalizations, while dashed lines represent identical translations.

However, there are also cases where this is not true, for instance, in the word *velvet*, we do not wish to separate the letters *l* and *v*. Thus, we shall describe the process we use to decide when to apply these transformations.

4.1 From Sentences To Phrases

The process to find phrases from sentences has been throughly studied in Machine Translation. This is generally done in two steps, Word Alignments and Phrase Extraction.

Alignment. The first step is to find the word-level alignments between the original post and its normalization. This is a well studied problem in MT, referred as Word Alignment (Brown et al., 1993). Many alignment models have been proposed, such as, the HMM-based word alignment models (Vogel et al., 1996) and the IBM models (Och and Ney, 2003). Generally, a symmetrization step is performed, where the bidirectional alignments are combined heuristically. In our work, we use the fast aligner proposed in (Dyer et al., 2013) to obtain the word alignments. Figure 1 shows an example of an word aligned pair of a tweet and its normalization.

Phrase Extraction. The phrasal extraction step (Ling et al., 2010), uses the word aligned sentences and extracts phrasal mappings between the original tweet and its normalization, named phrase pairs. For instance, in Figure 1, we would like to extract the phrasal mapping from *go 4* to *go for*, so that we learn that the word *4* in the context of *go* is normalized to the proposition *for*. To do this, the most common approach is to use the template proposed in (Och and Ney, 2004), which allows phrase pairs to be extracted, if there is at least one word alignment within the pair, and there are no

Table 4: Fragment of the phrase normalization model built, for each original phrase **o**, we present the top-3 normalized forms ranked by $f(\mathbf{n} \mid \mathbf{o})$.

Original (**o**)	Normalization (**n**)	$f(\mathbf{n} \mid \mathbf{o})$
wanna	want to	0.4679
wanna	will	0.0274
wanna	going to	0.0114
4	4	0.5641
4	for	0.01795
go 4	go for	1.0000

words inside the pair that are aligned to words not in the pair. For instance, in the example above, the phrase pair that normalizes *wanna* to *want to* would be extracted, but the phrase pair normalizing *wanna* to *want to go* would not, because the word *go* in the normalization is aligned to a word not in the pair.

Phrasal Features. After extracting the phrase pairs, a model is produced with features derived from phrase pair occurrences during extraction. This model is equivalent to phrasal translation model in MT, but we shall refer to it as the normalization model. For a phrase pair $\langle \mathbf{o}, \mathbf{n} \rangle$, where **o** is the original phrase, and **n** is the normalized phrase, we compute the normalization relative frequency $f(\mathbf{n} \mid \mathbf{o}) = \frac{C(\mathbf{n},\mathbf{o})}{C(\mathbf{o})}$, where $C(\mathbf{n}, \mathbf{o})$ denotes the number of times **o** was normalized to **n** and $C(\mathbf{o})$ denotes the number of times **o** was seen in the extracted phrase pairs. Table 4 gives a fragment of the normalization model. The columns represent the original phrase, its normalization and the probability, respectively.

In Table 4, we observe that the abbreviation *wanna* is normalized to *want to* with a relatively high probability, but it can also be normalized to other equivalent expressions, such as *will* and *going to*. The word *4* by itself has a low probability to be normalized to the preposition *for*. This is expected, since this decision cannot be made without context. However, we see that the phrase *go 4* is normalized to *go for* with a high probability, which specifies that within the context of *go*, *4* is generally used as a preposition.

4.2 From Phrases to Characters

While we can learn lexical variants that are in the corpora using the phrase model, we can only address word forms that have been observed in the corpora.

Table 5: Fragment of the character normalization model where examples representative of the lexical variant generation process are encoded in the model.

Original (**o**)	Normalization (**n**)	$f(\mathbf{n} \mid \mathbf{o})$
o o o	o o	0.0223
o o o	o	0.0439
s	c	0.0331
z	s	0.0741
s h	c h	0.019
2	t o	0.014
4	f o r	0.0013
0	o	0.0657
i n g f o r	i n g <space> f o r	0.4545
g f	g <space> f	0.01028

This is quite limited, since we cannot expect all the word forms to be present, such as all the possible orthographic errors for the word *cat*, such as *catt*, *kat* and *caaaat*. Thus, we will build a character-based model that learns the process lexical variants are generated at the subword level.

Our character-based model is similar to the phrase-based model, except that, rather than learning word-based mappings from the original tweet and the normalization sentences, we learn character-based mappings from the original phrases to the normalizations of those phrases. Thus, we extract the phrase pairs in the phrasal normalization model, and use them as a training corpora. To do this, for each phrase pair, we add a start token, *<start>*, and a end token, *<end>*, at the beginning and ending of the phrase pair. Afterwards, we separate all characters by space and add a space token *<space>* where spaces were originally. For instance, the phrase pair normalizing *DanielVeuleman* to *Daniel Veuleman* would be converted to *<start> d a n i e l v e u l e m a n <end>* and *<start> d a n i e l <space> v e u l e m a n <end>*.

Character-based Normalization Model - To build the character-based model, we proceed using the same approach as in the phrasal normalization model. We first align characters using Word Alignment Models, and then we perform phrase extraction to retrieve the phrasal character segments, and build the character-based model by collecting statistics. Once again, we provide examples of entries in the model in Table 5.

We observe that many of the normalizations dealt with in the previous model by memorizing phrases are captured with string transformations. For instance, from phrase pairs such as *tooo* to *too* and *sooo* to *so*, we learn that sequences of *o*'s can be reduced to 2 or 1 *o*. Other examples include orthographic substitutions, such as *2* for *to* and *4* for *for* (as found in *2gether*, *2morrow*, *4ever* and *4get*). Moreover, orthographic errors can be generated from mistaking characters with similar phonetic properties, such as, *s* to *c*, *z* to *s* and *sh* to *ch*, generating lexical variants such as *reprecenting*. Finally, we learn that the number *0* that resembles the letter *o*, can be used as a replacement, as in *g00d*. Finally, we can see that the rule *ingfor* to *ing for* attempts to find segmentation errors, such as *goingfor*, where a space between *going* and *for* was omitted.[6]

5 Normalization Decoder

In section 4, we built two models to learn the process of normalization, the phrase-based model and the character-based model. In this section, we describe the decoder we used to normalize the sentences.

The advantage of the phrase-based model is that it can make decisions for normalization based on context. That is, it contains phrasal units, such as, *go 4*, that determine, when the word *4* should be normalized to the preposition *for* and when to leave it as a number. However, it cannot address words that are unseen in the corpora. For instance, if the word form *4ever* is not seen in the training corpora, it is not be able to normalize it, even if it has seen the word *4get* normalized to *forget*. On the other hand, the character-based model learns subword normalizations, for instance, if we see the word *nnnnno* normalized to *no*, we can learn that repetitions of the letter *n* are generally shorted to *n*, which allows it to generate new word forms. This model has strong generalization potential, but the weakness of the character-based model is that it fails to consider the context of the normalization that the phrase-based model uses to make normalization decisions. Thus, our goal in this section is describe a decoder that uses both models to improve the quality of the normalizations.

5.1 Phrasal Decoder

We use Moses, an off-the-shelf phrase-based MT system (Koehn et al., 2007), to "translate" the original tweet its normalized form using the phrasal model (§4.1). Aside form the normalization probability, we also use the common features used in MT. These are the reverse normalization probability, the lexical and reverse lexical probabilities and the phrase penalty. We also use the MSD reordering model proposed in (Koehn et al., 2005), which adds reordering features.[7] The final score of each phrase pair is given as a sum of weighted log features. The weights for these features are optimized using MERT (Och, 2003). In our work, we sampled 150 tweets randomly from Twitter and normalized them manually, and used these samples as development data for MERT. As for the character-based model features, we simply rank the training phrase pairs by their relative frequency the $f(\mathbf{n} \mid \mathbf{o})$, and use the top-1000 phrase pairs as development set. Finally, a language model is required during decoding as a prior, since it defines the type of language that is produced by the output. We wish to normalized to formal language, which is generally better processed by NLP tools. Thus, for the phrase model, we use the English NIST dataset composed of 8M sentences in English from the news domain to build a 5-gram Kneser-Ney smoothed language model.

5.2 Character and Phrasal Decoder

We now turn to how to apply the character-based (§4.2), together with the phrasal model. For this model, we again use Moses, treating each character as a "word". The simplest way to combine both methods is first to decode the input **o** sentence with the character-based decoder, normalizing each word independently and then normalizing the resulting output using the phrase-based decoder, which enables the phrase model to score the outputs of the character model in context.

[6] Note that this captures the context in which such transformations are likely to occur: there are not many words that contain the sequence *ingfor*, so the probability that these should be normalized by inserting a space is high. On the other hand, we cannot assume that if we observe the sequence *gf*, we can safely separate these with a space. This is because, there are many words that contain this sequence, such as the abbreviation of *gf*(*girlfriend*), *dogfight*, and *bigfoot*.

[7] Reordering helps find lexical variants that are generated by transposing characters, such as, *mabye* to *maybe*.

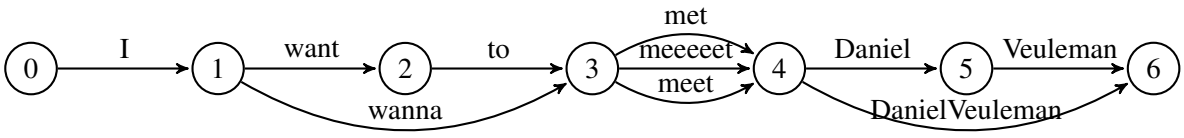


Figure 2: Example output lattice of the character-based decoder, for the sentence *I wanna meeeeet DanielVeuleman*.

Our process is as follows. Given the input sentence **o**, with the words $o_1, \ldots, o_m$, where m is the number of words in the input, we generate for each word o_i a list of n-best normalization candidates $z^1_{o_i}, \ldots, z^n_{o_i}$. We further filter the candidates using two criteria. We start by filtering each candidate $z^j_{o_i}$ that occurs less frequently than the original word o_i. This is motivated by our observation that lexical variants occur far less than the respective standard form. Second, we build a corpus of English language Twitter consisting of 70M tweets, extract the unigram counts, and perform Brown clustering (Brown et al., 1992) with $k = 3000$ clusters. Next, we calculate the cluster similarity between o_i and each surviving candidate, $z^j_{o_i}$. We filter the candidate if the similarity is less than 0.8. The similarity between two clusters represented as *bit strings*, $S[c(o_i), c(z^j_{o_i})]$, calculated as:

$$S(x, y) = \frac{2 \cdot |lpm\{x, y)\}|}{|x| + |y|},$$

where lpm computes the longest common prefix of the contexts and $|x|$ is the length of the bit string.[8] If a candidate contains more than one word (because a space was inserted), we set its count as the minimum count among its words. To find the cluster for multiple word units, we concatenate the words together, and find the cluster with the resulting word if it exists. This is motivated by the fact that it is common for missing spaces to exist in microblog corpora, generating new word forms, such as *wantto*, *goingfor*, and given a large enough corpora as the one we used, these errors occur frequently enough to be placed in the correct cluster. In fact, the variants such as *wanna* and *tmi*, occur in the same clusters as the words *wantto* and *toomuchinformation*.

Remaining candidates are combined into a word lattice, enabling us to perform lattice-based decoding with the phrasal model (Dyer et al., 2008). Figure 2, provides an example of such a lattice for the variant sentence *I wanna meeeet DanielVeuleman*.

5.3 Learning Variants from Monolingual Data

Until now, we learned normalizations from pairs of original tweets and their normalizations. We shall now describe a process to leverage monolingual documents to learn new normalizations, since the monolingual data is far easier to obtain than parallel data. This process is similar to the work in (Han et al., 2012), where confusion sets of contextually similar words are built initially as potential normalization candidates. We again use the $k = 3000$ Brown clusters,[9] and this time consider the contents of each cluster as a set of possible normalization variants. For instance, we find that the cluster that includes the word *never*, also includes the variant forms *neverrrr*, *neva* and *nevahhh*. However, the cluster also contains non-variant forms, such as *gladly* and *glady*. Thus, we want to find that *neverrrr* maps to *never*, while *glady* maps to *gladly* in the same cluster. Our work differs from previous work in that, rather than defining features manually, we use our character-based decoder to find the mappings between lexical variants and their normalizations.

For every word type w_i in cluster $c(w_i) = \{w_1, \ldots, w_n\}$, we generate a set of possible candidates for each word $w^1_i, \ldots, w^m_i$. Then, we build a directed acyclic graph (DAG), where every word. We add an edge between w_i and w_j, if w_i can be decoded into w_j using the character model from the previous section, and also if w_i occurs less than w_j; the second condition guarantees that the graph will be acyclic. Sample graphs are shown in Figure 3.

Afterwards, we find the number of paths between all nodes in the graph (this can be computed efficiently in $O(|V| + |E|)$ time). Then, for each word

[8] Brown clusters are organized such that more words with more similar distributions share common prefixes.

[9] The Brown clustering algorithm groups words together based on contextual similarity.

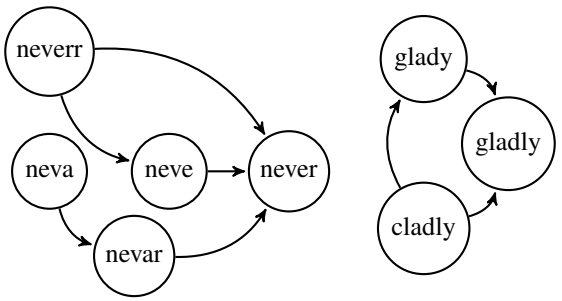


Figure 3: Example DAGs, built from the cluster containing the words *never* and *gladly*.

w_i, we find the w_j to which it has the highest number of paths to and extract the normalization of w_i to w_j. In case of a tie, we choose the word w_j that occurs more often in the monolingual corpora. This is motivated by the fact that normalizations are transitive. Thus, even if *neva* cannot be decoded directly to *never*, we can use *nevar* as an intermediate step to find the correct normalization. This is performed for all the clusters, and the resulting dictionary of lexical variants mapped to their standard forms is added to the training data of the character-based model.

6 Experiments

We evaluate our normalization model intrinsically by testing whether our normalizations more closely resemble standardized data, and then extrinsically by testing whether we can improve the translation quality of in-house as well as online Machine Translation systems by normalizing the input.

6.1 Setup

We use the gold standard by Ling et al. (2013), composed by 2581 English-Mandarin microblog sentence pairs. From this set, we randomly select 1290 pairs for development and 1291 pairs for testing.

The normalizer model is trained on the corpora extracted and filtered in section 3, in total, there were 1.3M normalization pairs used during training. The test sentences are normalized using four different setups. The first setup leaves the input sentence unchanged, which we call **No Norm**. The second uses the phrase-based model to normalize the input sentence, which we will denote **Norm+phrase**. The third uses the character-based model to output lattices, and then decodes with the phrase based model, which we will denote **Norm+phrase+char**. Finally, we test the same model after adding the training data extracted using monolingual documents, which we will refer as **Norm+phrase+char+mono**.

To test the normalizations themselves, we used Google Translate to translate the Mandarin side of the 1291 test sentence pairs back to English and use the original English tweet. While, this is by itself does not guarantee that the normalizations are correct, since the normalizations could be syntactically and semantically incorrect, it will allow us to check whether the normalizations are closer to those produced by systems trained on news data. This experiment will be called **Norm**.

As an application and extrinsic evaluation for our normalizer, we test if we can obtain gains on the MT task on microblog data by using our normalizer prior to translation. We build two MT systems using Moses. Firstly, we build a out-of-domain model using the full 2012 NIST Chinese-English dataset (approximately 8M sentence pairs), which is dataset from the news domain, and we will denote this system as **Inhouse+News**. Secondly, we build a in-domain model using the 800K sentence pairs from μtopia corpora (Ling et al., 2013). We also add the NIST dataset to improve coverage. We call this system **Inhouse+News+Weibo**. To train these systems, we use the Moses phrase-based MT system with standard features (Koehn et al., 2003). For reordering, we use the MSD reordering model (Axelrod et al., 2005). As the language model, we train a 5-gram model with Kneser-ney smoothing using a 10M tweets from twitter. Finally, the weights were tuned using MERT (Och, 2003). As for online systems, we consider the systems used to generate the paraphrase corpora in section 3, which we will denote as **Online A**, **Online B** and **Online C**[10]

The normalization and MT results are evaluated with BLEU-4 (Papineni et al., 2002) comparing the produced translations or normalizations with the appropriate reference.

6.2 Results

Results are shown in Table 6. In terms of the normalizations, we observe a much better match between

[10]The names of the systems are hidden to not violate the privacy issues in the terms and conditions of these online systems.

Table 6: Normalization and MT Results. Rows denote different normalizations, and columns different translation systems, except the first column (Norm), which denotes the normalization experiment. Cells display the BLEU score of that experiment.

Condition	Norm	Moses (News)	Moses (News+Weibo)	Online A	Online B	Online C
baseline	19.90	15.10	24.37	20.09	17.89	18.79
norm+phrase	21.96	15.69	24.29	20.50	18.13	18.93
norm+phrase+char	22.39	15.87	24.40	20.61	18.22	19.08
norm+phrase+char+mono	**22.91**	**15.94**	**24.46**	**20.78**	**18.37**	**19.21**

the normalized text with the reference, than the original tweets. In most cases, adding character-based models improves the quality of the normalizations.

We observe that better normalizations tend to lead to better translations. The relative improvements are most significant, when moving from No Norm to **norm+phrase** normalization. This is because, we are normalizing words that are not seen in general MT system's training data, but occur frequently in microblog data, such as *wanna* to *want to*, *u* to *you* and *im* to *i'm*. The only exception is in the **Inhouse+News+Weibo** system, where the normalization deteriorates the results. This is to be expected, since this system is trained on the same microblog data used to learn the normalizations. However, we can observe on **norm+phrase+char** that if we add the character-based model, we can observe improvements for this system as well as for all other ones. This is because the model is actually learning normalizations that are unseen in the data. Some examples of these normalization include, normalizing *lookin* to *looking*, *nutz* to *nuts* and *maimi* to *miami* but also separating *peaceof* to *peace of*. The fact that these improvements are obtained for all systems is strong evidence that we are actually producing good normalizations, and not overfitting to one of the systems that we used to generate our data. The gains are much smaller from **norm+phrase** to **norm+phrase+char**, since the improvements we obtain come from normalizing less frequent words. Finally, we can obtain another small improvement by adding monolingual data to the character-based model in **norm+phrase+char+mono**.

7 Related Work

Most of the work in microblog normalization is focused on finding the standard forms of lexical variants (Yang and Eisenstein, 2013; Han et al., 2013; Han et al., 2012; Kaufmann, 2010; Han and Baldwin, 2011; Gouws et al., 2011; Aw et al., 2006). A lexical variant is a variation of a standard word in a different lexical form. This ranges from minor or major spelling errors, such as *jst*, *juxt* and *jus* that are lexical variants of *just*, to abbreviations, such as *tmi* and *wanna*, which stand for *too much information* and *want to*, respectively. Jargon can also be treated as variants, for instance *cday* is a slang word for *birthday*, in some groups.

There are many rules that govern the process lexical variants are generated. Some variants are generated from orthographic errors, caused by some mistake from the user when writing. For instance, the variants *representin*, *representting*, or *reprecenting* can be generated by a spurious letter swap, insertion or substitution by the user. One way to normalize these types of errors is to attempt to insert, remove and swap words in a lexical variant until a word in a dictionary of standard words is found (Kaufmann, 2010). Contextual features are another way to find lexical variants, since variants generally occur in the same context as their standard form. This includes orthographic errors, abbreviations and slang. However, this is generally not enough to detect lexical variants, as many words share similar contexts, such as *already*, *recently* and *normally*. Consequently, contextual features are generally used to generate a confusion set of possible normalizations of a lexical variant, and then more features are used to find the correct normalization (Han et al., 2012). One simple approach is to compute the Levenshtein distance to find lexical similarities between words, which would effectively capture the mappings between *representting*, *reprecenting* and *representin* to *representing*. However, a pronunciation model (Tang et al., 2012)

would be needed to find the mapping between *g8*, *2day* and *4ever* to *great*, *today* and *forever*, respectively. Moreover, visual character similarity features would be required to find the mapping between *g00d* and ι to *good* and *i*.

Clearly, learning this process is a challenging task, and addressing each different case individually would require vast amounts of resources. Furthermore, once we change the language to normalize to another language, the types of rules that generate lexical variants would radically change and a new set of features would have to be engineered. We believe that to be successful in normalizing microblogs, the process to learn new lexical variants should be learned from data, making as few assumptions as possible. We learn our models without using any type of predefined features, such as phonetic features or lexical features. In fact, we will not assume that most words and characters map to themselves, as it is assumed in methods using the Levenshtein distance (Kaufmann, 2010; Han et al., 2012; Wang and Ng, 2013). All these mappings are learned from our data. Furthermore, in the work above, the dictionaries built using these methods assume that lexical variants are mapped to standard forms in a word-to-word mapping. Thus, variants such as *wanna*, *gonna* and *imma* are not normalizable, since they are normalized to multiple words *want to*, *going to* and *I am gonna*. Moreover, there are segmentation errors that occur from missing spaces, such as *sortof* and *goingfor*, which also map to more than one word to *sort of* and *going for*. These cases shall also be addressed in our work.

Wang and Ng (2013) argue that microblog normalization is not simply to map lexical variants into standard forms, but that other tasks, such as punctuation correction and missing word recovery should be performed. Consider the example tweet *you free?*, while there are no lexical variants in this message, the authors consider that it is the normalizer should recover the missing article *are* and normalize this tweet to *are you free?*. To do this, the authors train a series of models to detect and correct specific errors. While effective for narrow domains, training models to address each specific type of normalization is not scalable over all types of normalizations that need to be performed within the language, and the fact that a set of new models must be implemented for another language limits the applicability of this work.

Another strong point of the work above is that a decoder is presented, while the work on building dictionaries only normalize out of vocabulary (OOV) words. The work on (Han et al., 2012) trains a classifier to decide whether to normalize a word or not, but is still preconditioned on the fact that the word in question is OOV. Thus, lexical variants, such as, *4* and *u*, with the standard forms *for* and *you*, are left untreated, since they occur in other contexts, such as *u* in *u s a*. Inspired by the work above, we also propose a decoder based on the existing off-the-self decoder Moses (Koehn et al., 2007).

Finally, the work in (Xu et al., 2013) obtains paraphrases from Twitter, by finding tweets that contain common entities, such as *Obama*, that occur during the same period by matching temporal expressions. The resulting paraphrase corpora can also be used to train a normalizer.

8 Conclusion

We introduced a data-driven approach to microblog normalization based on paraphrasing. We build a corpora of tweets and their normalizations using parallel corpora from microblogs using MT techniques. Then, we build two models that learn generalizations of the normalization process, one the phrase level and on the character level. Then, we build a decoder that combines both models during decoding. Improvements on multiple MT systems support the validity of our method.

In future work, we shall attempt to build normalizations for other languages. We shall also attempt to learn an unsupervised normalization model with only monolingual data, similar to the work for MT in (Ravi and Knight, 2011).

Acknowledgements

The PhD thesis of Wang Ling is supported by FCT – Fundação para a Ciência e a Tecnologia, under project SFRH/BD/51157/2010. This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013.

The authors also wish to express their gratitude to the anonymous reviewers for their comments and insight.

References

[Aw et al.2006] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the ACL*, COLING-ACL '06, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Axelrod et al.2005] Amittai Axelrod, Ra Birch Mayne, Chris Callison-burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *In Proc. International Workshop on Spoken Language Translation (IWSLT*.

[Bannard and Callison-Burch2005] Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Bansal et al.2011] Mohit Bansal, Chris Quirk, and Robert C. Moore. 2011. Gappy phrasal alignment by agreement. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1308–1317, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Brown et al.1992] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Brown et al.1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19:263–311, June.

[Denkowski and Lavie2011] Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July. Association for Computational Linguistics.

[Dyer et al.2008] Chris Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of HLT-ACL*.

[Dyer et al.2013] Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT*, pages 644–648.

[Eisenstein et al.2011] Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1365–1374, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Eisenstein2013] Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of NAACL-HLT*, pages 359–369.

[Gouws et al.2011] Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11, pages 82–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Han and Baldwin2011] Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 368–378, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Han et al.2012] Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 421–432, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Han et al.2013] Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5.

[Hawn2009] Carleen Hawn. 2009. Take two aspirin and tweet me in the morning: how twitter, facebook, and other social media are reshaping health care. *Health affairs*, 28(2):361–368.

[Kaufmann2010] M. Kaufmann. 2010. Syntactic Normalization of Twitter Messages. *studies*, 2.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

[Koehn et al.2005] Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 nist mt evaluation. In *Proceedings of Machine Translation Evaluation Workshop 2005*.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-burch, Richard Zens, Rwth

Aachen, Alexandra Constantin, Marcello Federico, Nicola Bertoldi, Chris Dyer, Brooke Cowan, Wade Shen, Christine Moran, and Ondrej Bojar. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

[Laviosa1998] Sara Laviosa. 1998. Core patterns of lexical use in a comparable corpus of English lexical prose. *Meta*, 43(4):557–570.

[Ling et al.2010] Wang Ling, Tiago Luís, João Graça, Luísa Coheur, and Isabel Trancoso. 2010. Towards a general and extensible phrase-extraction algorithm. In *IWSLT '10: International Workshop on Spoken Language Translation*, pages 313–320, Paris, France.

[Ling et al.2013] Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, ACL '13. Association for Computational Linguistics.

[Och and Ney2003] Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

[Och and Ney2004] Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, December.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ravi and Knight2011] Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *ACL*, pages 12–21.

[Resnik and Smith2003] Philip Resnik and Noah A Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

[Tang et al.2012] Hao Tang, Joseph Keshet, and Karen Livescu. 2012. Discriminative pronunciation modeling: A large-margin, feature-rich approach. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 194–203. Association for Computational Linguistics.

[Vogel et al.1996] S. Vogel, H. Ney, and C. Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

[Volansky et al.2013] Vered Volansky, Noam Ordan, and Shuly Wintner. 2013. On the features of translationese. *Literary and Linguistic Computing*.

[Wang and Ng2013] Pidong Wang and Hwee Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of NAACL-HLT 2013*, NAACL '13. Association for Computational Linguistics.

[Xu et al.2013] Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 121–128, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Yang and Eisenstein2013] Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proc. of EMNLP*.

Question Difficulty Estimation in Community Question Answering Services*

Jing Liu[†] **Quan Wang**[‡] **Chin-Yew Lin**[♯] **Hsiao-Wuen Hon**[♯]
[†]Harbin Institute of Technology, Harbin 150001, P.R.China
[‡]Peking University, Beijing 100871, P.R.China
[♯]Microsoft Research Asia, Beijing 100080, P.R.China
jliu@ir.hit.edu.cn quanwang1012@gmail.com {cyl,hon}@microsoft.com

Abstract

In this paper, we address the problem of estimating question difficulty in community question answering services. We propose a competition-based model for estimating question difficulty by leveraging pairwise comparisons between questions and users. Our experimental results show that our model significantly outperforms a PageRank-based approach. Most importantly, our analysis shows that the text of question descriptions reflects the question difficulty. This implies the possibility of predicting question difficulty from the text of question descriptions.

1 Introduction

In recent years, community question answering (CQA) services such as Stackoverflow[1] and Yahoo! Answers[2] have seen rapid growth. A great deal of research effort has been conducted on CQA, including: (1) question search (Xue et al., 2008; Duan et al., 2008; Suryanto et al., 2009; Zhou et al., 2011; Cao et al., 2010; Zhang et al., 2012; Ji et al., 2012); (2) answer quality estimation (Jeon et al., 2006; Agichtein et al., 2008; Bian et al., 2009; Liu et al., 2008); (3) user expertise estimation (Jurczyk and Agichtein, 2007; Zhang et al., 2007; Bouguessa et al., 2008; Pal and Konstan, 2010; Liu et al., 2011); and (4) question routing (Zhou et al., 2009; Li and King, 2010; Li et al., 2011).

However, less attention has been paid to ***question difficulty estimation*** in CQA. Question difficulty estimation can benefit many applications: (1) Experts are usually under time constraints. We do not want to bore experts by routing every question (including both easy and hard ones) to them. Assigning questions to experts by matching question difficulty with expertise level, not just question topic, will make better use of the experts' time and expertise (Ackerman and McDonald, 1996). (2) Nam et al. (2009) found that winning the point awards offered by the reputation system is a driving factor in user participation in CQA. Question difficulty estimation would be helpful in designing a better incentive mechanism by assigning higher point awards to more difficult questions. (3) Question difficulty estimation can help analyze user behavior in CQA, since users may make strategic choices when encountering questions of different difficulty levels.

To the best of our knowledge, not much research has been conducted on the problem of estimating question difficulty in CQA. The most relevant work is a PageRank-based approach proposed by Yang et al. (2008) to estimate task difficulty in crowdsourcing contest services. Their key idea is to construct a graph of tasks: creating an edge from a task t_1 to a task t_2 when a user u wins task t_1 but loses task t_2, implying that task t_2 is likely to be more difficult than task t_1. Then the standard PageRank algorithm is employed on the task graph to estimate PageRank score (i.e., difficulty score) of each task. This approach implicitly assumes that task difficulty is the only factor affecting the outcomes of competitions (i.e. the best answer). However, the outcomes of competitions depend on both the difficulty levels of tasks and the expertise levels of competitors (i.e.

*This work was done when Jing Liu and Quan Wang were visiting students at Microsoft Research Asia. Quan Wang is currently affiliated with Institute of Information Engineering, Chinese Academy of Sciences.

[1]http://stackoverflow.com

[2]http://answers.yahoo.com

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 85–90,
Seattle, Washington, USA, 18-21 October 2013.

other answerers).

Inspired by Liu et al. (2011), we propose a competition-based approach which jointly models question difficulty and user expertise level. Our approach is based on two intuitive assumptions: (1) given a question answering thread, the difficulty score of the question is higher than the expertise score of the asker, but lower than that of the best answerer; (2) the expertise score of the best answerer is higher than that of the asker as well as all other answerers. Given the two assumptions, we can determine the question difficulty score and user expertise score through pairwise comparisons between (1) a question and an asker, (2) a question and a best answerer, (3) a best answerer and an asker, and (4) a best answerer and all other non-best answerers.

The main contributions of this paper are:

• We propose a competition-based approach to estimate question difficulty (Sec. 2). Our model significantly outperforms the PageRank-based approach (Yang et al., 2008) for estimating question difficulty on the data of Stack Overflow (Sec. 3.2).

• Additionally, we calibrate question difficulty scores across two CQA services to verify the effectiveness of our model (Sec. 3.3).

• Most importantly, we demonstrate that different words or tags in the question descriptions indicate question difficulty levels. This implies the possibility of predicting question difficulty purely from the text of question descriptions (Sec. 3.4).

2 Competition based Question Difficulty Estimation

CQA is a virtual community where people can ask questions and seek opinions from others. Formally, when an asker u_a posts a question q, there will be several answerers to answer her question. One answer among the received ones will be selected as the best answer by the asker u_a or voted by the community. The user who provides the best answer is called the best answerer u_b, and we denote the set of all non-best answerers as $S = \{u_{o_1}, \cdots, u_{o_M}\}$. Assuming that question difficulty scores and user expertise scores are expressed on the same scale, we make the following two assumptions:

• The difficulty score of question q is higher than the expertise score of asker u_a, but lower than that of the best answerer u_b. This is intuitive since the best answer u_b correctly responds to question q that asker u_a does not know.

• The expertise score of the best answerer u_b is higher than that of asker u_a and all answerers in S. This is straightforward since the best answerer u_b solves question q better than asker u_a and all non-best answerers in S.

Let's view question q as a pseudo user u_q. Taking a competitive viewpoint, each pairwise comparison can be viewed as a two-player competition with one winner and one loser, including (1) one competition between pseudo user u_q and asker u_a, (2) one competition between pseudo user u_q and the best answerer u_b, (3) one competition between the best answerer u_b and asker u_a, and (4) $|S|$ competitions between the best answerer u_b and all non-best answers in S. Additionally, pseudo user u_q wins the first competition and the best answerer u_b wins all remaining $(|S| + 2)$ competitions.

Hence, the problem of estimating the question difficulty score (and the user expertise score) is cast as a problem of learning the relative skills of players from the win-loss results of the generated two-player competitions. Formally, let $\mathcal{Q}$ denote the set of all questions in one category (or topic), and $\mathcal{R}_q$ denote the set of all two-player competitions generated from question $q \in \mathcal{Q}$, i.e., $\mathcal{R}_q = \{(u_a \prec u_q), (u_q \prec u_b), (u_a \prec u_b), (u_{o_1} \prec u_b), \cdots, (u_{o_{|S|}} \prec u_b)\}$, where $j \prec i$ means that user i beats user j in the competition. Define

$$\mathcal{R} = \bigcup_{q \in \mathcal{Q}} \mathcal{R}_q \tag{1}$$

as the set of all two-player competitions. Our problem is then to learn the relative skills of players from $\mathcal{R}$. The learned skills of the pseudo question users are question difficulty scores, and the learned skills of all other users are their expertise scores.

TrueSkill In this paper, we follow (Liu et al., 2011) and apply TrueSkill to learn the relative skills of players from the set of generated competitions $\mathcal{R}$ (Equ. 1). TrueSkill (Herbrich et al., 2007) is a Bayesian skill rating model that is developed for estimating the relative skill levels of players in games. In this paper, we present a two-player version of TrueSkill with no-draw.

TrueSkill assumes that the practical performance of each player in a game follows a normal distribu-

tion $\mathcal{N}(\mu, \sigma^2)$, where μ means the skill level of the player and σ means the uncertainty of the estimated skill level. Basically, TrueSkill learns the skill levels of players by leveraging Bayes' theorem. Given the current estimated skill levels of two players (priori probability) and the outcome of a new game between them (likelihood), TrueSkill model updates its estimation of player skill levels (posterior probability). TrueSkill updates the skill level μ and the uncertainty σ intuitively: (a) if the outcome of a new competition is expected, i.e. the player with higher skill level wins the game, it will cause small updates in skill level μ and uncertainty σ; (b) if the outcome of a new competition is unexpected, i.e. the player with lower skill level wins the game, it will cause large updates in skill level μ and uncertainty σ. According to these intuitions, the equations to update the skill level μ and uncertainty σ are as follows:

$$\mu_{winner} = \mu_{winner} + \frac{\sigma^2_{winner}}{c} \cdot v\left(\frac{t}{c}, \frac{\varepsilon}{c}\right), \quad (2)$$

$$\mu_{loser} = \mu_{loser} - \frac{\sigma^2_{loser}}{c} \cdot v\left(\frac{t}{c}, \frac{\varepsilon}{c}\right), \quad (3)$$

$$\sigma^2_{winner} = \sigma^2_{winner} \cdot \left[1 - \frac{\sigma^2_{winner}}{c^2} \cdot w\left(\frac{t}{c}, \frac{\varepsilon}{c}\right)\right], \quad (4)$$

$$\sigma^2_{loser} = \sigma^2_{loser} \cdot \left[1 - \frac{\sigma^2_{loser}}{c^2} \cdot w\left(\frac{t}{c}, \frac{\varepsilon}{c}\right)\right], \quad (5)$$

where $t = \mu_{winner} - \mu_{loser}$ and $c^2 = 2\beta^2 + \sigma^2_{winner} + \sigma^2_{loser}$. Here, ε is a parameter representing the probability of a draw in one game, and $v(t, \varepsilon)$ and $w(t, \varepsilon)$ are weighting factors for skill level μ and standard deviation σ respectively. Please refer to (Herbrich et al., 2007) for more details. In this paper, we set the initial values of the skill level μ and the standard deviation σ of each player the same as the default values used in (Herbrich et al., 2007).

3 Experiments

3.1 Data Set

In this paper, we use Stack Overflow (SO) for our experiments. We obtained a publicly available data set[3] of SO between July 31, 2008 and August 1, 2012. SO contains questions with various topics, such as programming, mathematics, and English. In this paper, we use SO C++ programming (SO/CPP) and mathematics[4] (SO/Math) questions for our main experiments. Additionally, we use the data of Math Overflow[5] (MO) for calibrating question difficulty scores across communities (Sec. 3.3). The statistics of these data sets are shown in Table 1.

	SO/CPP	**SO/Math**	**MO**
# of questions	122,012	51,174	27,333
# of answers	357,632	94,488	65,966
# of users	67,819	16,961	12,064

Table 1: The statistics of the data sets.

To evaluate the effectiveness of our proposed model for estimating question difficulty scores, we randomly sampled 300 question pairs from both SO/CPP and SO/Math, and we asked experts to compare the difficulty of every pair. We had two graduate students majoring in computer science annotate the SO/CPP question pairs, and two graduate students majoring in mathematics annotate the SO/Math question pairs. When annotating each question pair, only the titles, descriptions, and tags of the questions were shown, and other information (e.g. users, answers, etc.) was excluded. Given each pair of questions (q_1 and q_2), the annotators were asked to give one of four labels: (1) $q_1 \succ q_2$, which means that the difficulty of q_1 was higher than q_2; (2) $q_1 \prec q_2$, which means that the difficulty of q_1 was lower than q_2; (3) $q_1 = q_2$, which means that the difficulty of q_1 was equal to q_2; (4) Unknown, which means that the annotator could not make a decision. The agreements between annotators on both SO/CPP (kappa value $= 0.741$) and SO/Math (kappa value $= 0.873$) were substantial. When evaluating models, we only kept the pairs that annotators had given the same labels. There were 260 SO/CPP question pairs and 280 SO/Math question pairs remaining.

3.2 Accuracy of Question Difficulty Estimation

We employ a standard evaluation metric for information retrieval: accuracy (Acc), defined as follows:

$$Acc = \frac{\text{the number of correct pairwise comparisons}}{\text{the total number of pairwise comparisons}}.$$

We use the *PageRank*-based approach proposed by Yang et al. (2008) as a baseline. As described in

[3] http://blog.stackoverflow.com/category/cc-wiki-dump/

[4] http://math.stackexchange.com

[5] http://mathoverflow.net

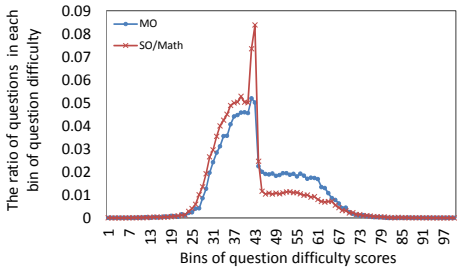


Figure 1: The distributions of calibrated question difficulty scores of MO and SO/Math.

Sec. 1, this is the most relevant method for our problem. Table 2 gives the accuracy of the baseline and our *Competition*-based approach on SO/CPP and SO/Math. From the results, we can see that (1) the proposed *Competition*-based approach significantly outperformed the *PageRank*-based approach on both data sets; (2) *PageRank*-based approach only achieved a similar performance as randomly guessing. This is because the *PageRank*-based approach only models the outcomes of competitions affected by question difficulty. However, the outcomes of competitions depend on both the question difficulty levels and the expertise levels of competitors. Our *Competition*-based approach considers both these factors for modeling the competitions. The experimental results demonstrate the advantage of our approach.

	Acc@**SO/CPP**	*Acc*@**SO/Math**
PageRank	50.38%	48.93%
Competition	66.54%	71.79%

Table 2: Accuracy on SO/CPP and SO/Math.

3.3 Calibrating Question Difficulty across CQA Services

Both MO and SO/Math are CQA services for asking mathematics questions. However, these two services are designed for different audiences, and they have different types of questions. MO's primary goal is asking and answering research level mathematics questions[6]. In contrast, SO/Math is for people studying mathematics at any level in related fields[7]. Usually, the community members in MO are not interested in basic mathematics questions. If a posted question is too elementary, someone will suggest moving it to SO/Math. Similarly, if a posted question is advanced, the community members in SO/Math will recommend moving it to MO. Hence, it is expected that the ratio of difficult questions in MO is higher than SO/Math. In this section, we examine whether our competition-based model can identify such differences.

We first calibrate the estimated question difficulty scores across these two services on a same scale. The key idea is to link the users who participate in both services. In both MO and SO/Math, users can specify their home pages. We assume that if a user u_1 on MO and a user u_2 on SO/Math have the same home page URL, they should be linked as one natural person in the real world. We successfully linked 633 users. They provided $18,196$ answers in SO/Math among which $10,993$ (60.41%) were selected as the best answers. In contrast, they provided $8,044$ answers in MO among which $3,215$ (39.97%) were selected as the best answers. This shows that these users reflect more competitive contests in MO. After the common users are linked, we have a joint data set of MO and SO/Math. Then, we can calibrate the estimated question difficulty scores across the two services by performing the competition-based model on the joint data set. Figure 1 shows the distributions of the calibrated question difficulty scores of MO and SO/Math on the same scale. As expected, we observed that the ratio of difficult questions in MO was higher than SO/Math. Additionally, these two distributions were significantly different (Kolmogorov-Smirnov Test, p-value < 0.05). This demonstrates that our competition-based model successfully identified the difference between questions on two CQA services.

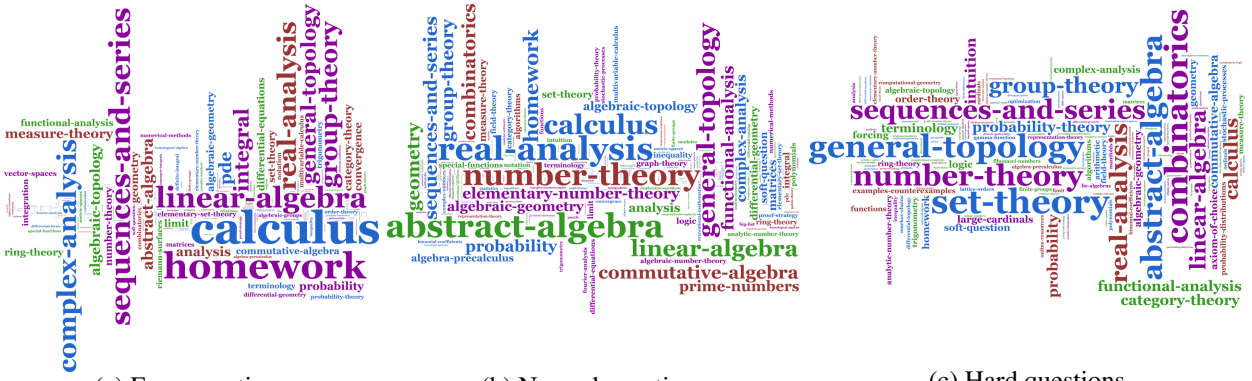
3.4 Analysis on the Question Descriptions

In this section, we analyze the text of question descriptions on the scale of question difficulty scores estimated by the competition model.

Micro Level We first examine the frequency distributions of individual words over the question difficulty scores. Figure 3 shows the examples of four words in SO/CPP. We observe that the words *'list'* and *'array'* have the lowest mean of difficulty scores, compared to the words *'virtual'* and *'gcc'*. This is reasonable, since *'list'* and *'array'* are related

[6] http://mathoverflow.net/faq

[7] http://area51.stackexchange.com/proposals/3355/mathematics



(a) Easy questions (b) Normal questions (c) Hard questions

Figure 2: Tag clouds on SO/Math questions with different difficulty levels

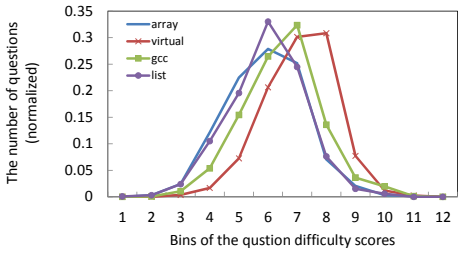


Figure 3: The frequency distributions of words on the scale of question difficulty scores (SO/CPP).

to basic concepts in programming language, while *'virtual'* and *'gcc'* are related to more advanced topics. It can be observed that the order of the means of the difficulty scores of these words are well aligned to our learning process.

Macro Level We evenly split the range of question difficulty scores into n buckets, and we grouped the questions into the n buckets according to which bucket their difficulty scores were in. Then, we had n question buckets and each bucket corresponded to a word distribution of questions. Let variable X denote the distance between the difficulty scores in two question buckets (which is the difference between the average difficulty scores of questions in the two buckets), and variable Y denote the Jensen-Shannon distance between word distributions in two question buckets. We examined the correlation between variable X and variable Y. The experimental results showed that the correlation between these two variables were strongly positive. Specifically, the correlation coefficient on SO/CPP was 0.8129 and on SO/Math was 0.7412. In other words, when the distance between the difficulty scores of two buckets become larger, the two word distributions in the two buckets become less similar, and vice versa.

We further visualized the word distribution in each question bucket. We set n as 3, and we had three question buckets: (1) easy questions; (2) normal questions; and (3) hard questions. Figure 3.4 plots the tag clouds of SO/Math questions in the three buckets. The size of tags is proportional to the frequency of tags in each bucket. We observed that (1) the tag *'homework'* and *'calculus'* become smaller from easy questions to hard questions; (2) the tag *'set-theory'* becomes larger. These observations also reflect our learning process.

The above experimental results show that different words or tags of question descriptions reflect the question difficulty levels. This implies the possibility of predicting question difficulty purely from the text of question descriptions.

4 Conclusion and Future Work

In this paper, we address the problem of estimating question difficulty in CQA services. Our proposed competition-based model for estimating question difficulty significantly outperforms the PageRank-based approach. Most importantly, our analysis shows that the text of question descriptions reflects the question difficulty. In the future, we would like to explore predicting question difficulty from the text of question descriptions. We also will investigate non-technical areas, where there might be no strongly distinct notion of experts and non-experts.

Acknowledgments

We would like to thank Yunbo Cao and Jie Cai for their valuable suggestions for this paper, and the anonymous reviewers for their helpful comments.

References

M.S. Ackerman and D.W. McDonald. 1996. Answer garden 2: merging organizational memory with collaborative help. In *Proceedings of CSCW*.

E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. 2008. Finding high-quality content in social media. In *Proceedings of WSDM*.

J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. 2009. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of WWW*.

M. Bouguessa, B. Dumoulin, and S. Wang. 2008. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *Proceeding of SIGKDD*.

Xin Cao, Gao Cong, Bin Cui, and Christian S Jensen. 2010. A generalized framework of exploring category information for question retrieval in community question answer archives. In *Proceedings of WWW*.

H. Duan, Y. Cao, C.Y. Lin, and Y. Yu. 2008. Searching questions by identifying question topic and question focus. In *Proceedings of ACL*.

R. Herbrich, T. Minka, and T. Graepel. 2007. Trueskill: A bayesian skill rating system. In *Proceedings of NIPS*.

J. Jeon, W.B. Croft, J.H. Lee, and S. Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of SIGIR*.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of CIKM*.

P. Jurczyk and E. Agichtein. 2007. Discovering authorities in question answer communities by using link analysis. In *Proceedings of CIKM*.

B. Li and I. King. 2010. Routing questions to appropriate answerers in community question answering services. In *Proceedings of CIKM*.

B. Li, I. King, and M.R. Lyu. 2011. Question routing in community question answering: putting category in its place. In *Proceedings of CIKM*.

Y. Liu, J. Bian, and E. Agichtein. 2008. Predicting information seeker satisfaction in community question answering. In *Proceedings of SIGIR*.

J. Liu, Y.I. Song, and C.Y. Lin. 2011. Competition-based user expertise score estimation. In *Proceedings of SIGIR*.

K.K. Nam, M.S. Ackerman, and L.A. Adamic. 2009. Questions in, knowledge in?: a study of naver's question answering community. In *Proceedings of CHI*.

A. Pal and J.A. Konstan. 2010. Expert identification in community question answering: exploring question selection bias. In *Proceedings of CIKM*.

M.A. Suryanto, E.P. Lim, A. Sun, and R.H.L. Chiang. 2009. Quality-aware collaborative question answering: methods and evaluation. In *Proceedings of WSDM*.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of SIGIR*.

Jiang Yang, Lada Adamic, and Mark Ackerman. 2008. Competing to share expertise: the taskcn knowledge sharing community. In *Proceedings of ICWSM*.

J. Zhang, M.S. Ackerman, and L. Adamic. 2007. Expertise networks in online communities: structure and algorithms. In *Proceedings of WWW*.

Weinan Zhang, Zhaoyan Ming, Yu Zhang, Liqiang Nie, Ting Liu, and Tat-Seng Chua. 2012. The use of dependency relation graph to enhance the term weighting in question retrieval. In *Proceedings of COLING*.

Y. Zhou, G. Cong, B. Cui, C.S. Jensen, and J. Yao. 2009. Routing questions to the right users in online communities. In *Proceedings of ICDE*.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of ACL*.

Measuring Ideological Proportions in Political Speeches

Yanchuan Sim* **Brice D. L. Acree†** **Justin H. Gross†** **Noah A. Smith***
*Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{ysim,nasmith}@cs.cmu.edu

†Department of Political Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599, USA
{brice.acree,jhgross}@unc.edu

Abstract

We seek to measure political candidates' ideological positioning from their speeches. To accomplish this, we infer ideological cues from a corpus of political writings annotated with known ideologies. We then represent the speeches of U.S. Presidential candidates as sequences of cues and lags (filler distinguished only by its length in words). We apply a domain-informed Bayesian HMM to infer the proportions of ideologies each candidate uses in each campaign. The results are validated against a set of preregistered, domain expert-authored hypotheses.

1 Introduction

The artful use of language is central to politics, and the language of politicians has attracted considerable interest among scholars of political communication and rhetoric (Charteris-Black, 2005; Hart, 2009; Deirmeier et al., 2012; Hart et al., 2013) and computational linguistics (Thomas et al., 2006; Fader et al., 2007; Gerrish and Blei, 2011, *inter alia*). In American politics, candidates for office give speeches and write books and manifestos expounding their ideas. Every political season, however, there are accusations of candidates "flip-flopping" on issues, with opinion shows, late-night comedies, and talk radio hosts replaying clips of candidates contradicting earlier statements. Presidential candidate Mitt Romney's own aide infamously proclaimed in 2012: "I think you hit a reset button for the fall campaign [i.e., the general election]. Everything changes. It's almost like an Etch-a-Sketch. You can kind of shake it up and we start all over again."

A more general observation, often stated but not yet, to our knowledge, tested empirically, is that successful primary candidates "move to the center" before a general election. The expectation follows directly from long-standing and widely influential theories of political competition that are collectively referred to in their simplest form as the "median voter theorem" (Hotelling, 1929; Black, 1948; Downs, 1957). Thus it is to be expected that when a set of voters that are more ideologically concentrated are replaced by a set who are more widely dispersed across the ideological spectrum, as occurs in the transition between the United States primary and general elections, that candidates will present themselves as more moderate in an effort to capture enough votes to win.

Do political candidates in fact stray ideologically at opportune moments? More specifically, can we measure candidates' ideological positions from their prose at different times? Following much work on *classifying* the political ideology expressed by a piece of text (Laver et al., 2003; Monroe and Maeda, 2004; Hillard et al., 2008), we start from the assumption that a candidate's choice of words and phrases reflects a deliberate attempt to signal common cause with a target audience, and as a broader strategy, to respond to political competitors. Our central hypothesis is that, despite candidates' intentional vagueness, differences in position—among candidates or over time—can be automatically detected and described as *proportions* of ideologies expressed in a speech.

In this work, we operationalize ideologies in a novel empirical way, exploiting political writings published in explicitly ideological books and magazines (§2).[1] The corpus then serves as evidence for

[1] We consider general positions in terms of broad ideological groups that are widely discussed in current political discourse (e.g., "Far Right," "Religious Right," "Libertarian,"'

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 91–101,
Seattle, Washington, USA, 18-21 October 2013.

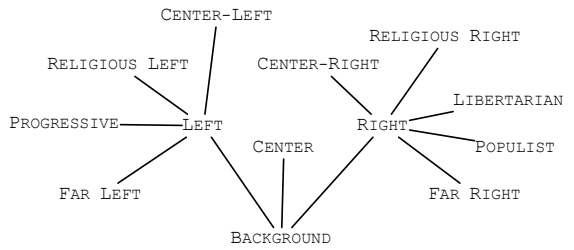


Figure 1: Ideology tree showing the labels for the ideological corpus in §2.1 (excluding BACKGROUND) and corresponding to states in the HMM (§3.3).

a probabilistic model that allows us to automatically infer compact, human-interpretable lexicons of cues strongly associated with each ideology.

These lexicons are used, in turn, to create a low-dimensional representation of political speeches: a speech is a sequence of cues interspersed with lags. Lags correspond to the lengths of sequences of non-cue words, which are treated as irrelevant to the inference problem at hand. In other words, a speech is represented as a series alternating between cues signaling ideological positions and uninteresting filler.

Our main contribution is a probabilistic technique for inferring proportions of ideologies expressed by a candidate (§3). The inputs to the model are the cue-lag representation of a speech and a domain-specific topology relating ideologies to each other. The topology tree (shown in Figure 1) encoding the closeness of different ideologies and, by extension, the odds of transitioning between them within a speech. Bayesian inference is used to manage uncertainty about the associations between cues and ideologies, probabilities of traversing each of the tree's edges, and other parameters.

We demonstrate the usefulness of the measurement model by showing that it accurately recovers pre-registered beliefs regarding narratives widely accepted—but not yet tested empirically—about the 2008 and 2012 U.S. Presidential elections (§4).

2 First Stage: Cue Extraction

We first present a data-driven technique for automatically constructing "cue lexicons" from texts labeled with ideologies by domain experts.

etc.). Analysis of positions on specific issues is left for future work.

Total tokens	32,835,190
Total types	138,235
Avg. tokens per book	77,628
Avg. tokens per mag. issue	31,713

Breakdown by ideology:	*Documents*	*Tokens*
LEFT	0	0
FAR LEFT	112	3,334,601
CENTER-LEFT	196	7,396,264
PROGRESSIVE LEFT	138	7,257,723
RELIGIOUS LEFT	7	487,844
CENTER	5	429,480
RIGHT	97	3,282,744
FAR RIGHT	211	7,392,163
LIBERTARIAN RIGHT	88	1,703,343
CENTER-RIGHT	9	702,444
POPULIST RIGHT	5	407,054
RELIGIOUS RIGHT	6	441,530

Table 1: Ideology corpus statistics. Note that some documents are not labeled with finer-grained ideologies.

2.1 Ideological Corpus

We start with a collection of contemporary political writings whose authors are perceived as representative of one particular ideology. Our corpus consists of two types of documents: books and magazines. Books are usually written by a single author, while each magazine consists of regularly published issues with collections of articles written by several authors. A political science domain expert who is a co-author of this work manually labeled each element in a collection of 112 books and 10 magazine titles[2] with one of three coarse ideologies: LEFT, RIGHT, or CENTER. Documents that were labeled LEFT and RIGHT were further broken down into more fine-grained ideologies, shown in Fig. 1.[3] Table 1 summarizes key details about the ideological corpus.

In addition to ideology labels, individual chapters within the books were manually tagged with topics that the chapter was about. For instance, in Barack Obama's book *The Audacity of Hope*, his chapter

[2]There are 765 magazine issues, which are published bi-weekly to quarterly, depending on the magazine. All of a magazine's issues are labeled with the same ideology.

[3]We cannot claim that these texts are "pure" examples of the ideologies they are labeled with (i.e., they may contain parts that do not match the label). By finding relatively few terms strongly associated with texts sharing a label, our model should be somewhat robust to impurities, focusing on those terms that are indicative of whatever drew the expert to identify them as (mostly) sharing an ideology.

titled "Faith" is labeled as RELIGIOUS. Not all chapters have clearly defined topics, and as such, these chapters are simply labeled MISC. Magazines are not labeled with topics because each issue of a magazine generally touches on multiple topics. There are a total of 61 topics; the full list can be found in the supplementary materials, along with a table summarizing key details about the corpus, which contains 32.8 million tokens.

2.2 Cue Discovery Model

We use the ideological corpus to infer ideological cues: terms that are strongly associated with an ideology. Because our ideologies are organized hierarchically, we required a technique that can account for multiple effects within a single text. We further require that the sets of cue terms be small, so that they can be inspected by domain experts. We therefore turn to the sparse additive generative (SAGE) models introduced by Eisenstein et al. (2011).

Like other probabilistic language models, SAGE assigns probability to a text as if it were a bag of terms. It differs from most language models in parameterizing the distribution using a generalized linear model, so that different effects on the log-odds of terms are additive. In our case, we define the probability of a term w conditioned on attributes of the text in which it occurs. These attributes include both the ideology and its coarsened version (e.g., a FAR RIGHT book also has the attribute RIGHT). For simplicity, let $\mathcal{A}(d)$ denote the set of attributes of document d and $\mathcal{A} = \bigcup_d \mathcal{A}(d)$. The parametric form of the distribution is given, for term w in document d, by:

$$p(w \mid \mathcal{A}(d); \boldsymbol{\eta}) = \frac{\exp\left(\eta^0_w + \sum_{a \in \mathcal{A}(d)} \eta^a_w\right)}{Z(\mathcal{A}(d), \boldsymbol{\eta})}$$

Each of the η weights can be a positive or negative value influencing the probability of the word, conditioned on various properties of the document. When we stack an attribute a's weights into a vector across all words, we get an $\boldsymbol{\eta}^a$ vector, understood as an effect on the term distribution. (We use $\boldsymbol{\eta}$ to refer to the collection of all of these vectors.) The effects in our model, described in terms of attributes, are:

- $\boldsymbol{\eta}^0$, the background (log) frequencies of words, fixed to the empirical frequencies in the corpus. Hence the other effects can be understood as *deviations* from this background distribution.
- $\boldsymbol{\eta}^{i_c}$, the coarse ideology effect, which takes different values for LEFT, RIGHT, and CENTER.
- $\boldsymbol{\eta}^{i_f}$, the fine ideology effect, which takes different values for the fine-grained ideologies corresponding to the leaves in Fig. 1.
- $\boldsymbol{\eta}^t$, the topic effect, taking different values for each of the 61 manually assigned topics. We further include one effect for each magazine series (of which there are 10) to account for each magazine's idiosyncrasies (topical or otherwise).
- $\boldsymbol{\eta}^d$, a document-specific effect, which captures idiosyncratic usage within a single document.

Note that the effects above are not mutually exclusive, although some effects never appear together due to constraints imposed by their semantics (e.g., no book is labeled both LEFT and RIGHT).

When estimating the parameters of the model (the $\boldsymbol{\eta}$ vectors), we impose a sparsity-inducing ℓ_1 prior that forces many weights to zero. The objective is:

$$\max_{\boldsymbol{\eta}} \sum_d \sum_{w \in d} \log p(w \mid \mathcal{A}(d); \boldsymbol{\eta}) - \sum_{a \in \mathcal{A}} \lambda_a \|\boldsymbol{\eta}^a\|_1$$

This objective function is convex but requires special treatment due to non-differentiability when any elements are zero; we use the OWL-QN algorithm to solve it (Andrew and Gao, 2007). To reduce the complexity of the hyperparameter space (the possible values of all λ_a) and to encourage similar levels of sparsity across the different effect vectors, we let, for each ideology attribute a,

$$\lambda_a = \lambda \cdot |\mathcal{V}(a)| \,/ \max_{a' \in \mathcal{A}} |\mathcal{V}(a')|$$

where $\mathcal{V}(a)$ is the set of term types appearing in the data with attribute a (i.e., its vocabulary) , and λ is a hyperparameter we can adjust to control the amount of sparsity in the SAGE vectors. For the non-ideology effects, we fix $\lambda_a = 10$ (not tuned).

2.3 Bigram and Trigram Lexicons

After estimating parameters, we are left with sparse $\boldsymbol{\eta}^a$ for each attribute. We are only interested, however, in the ideological attributes $\mathcal{I} \subset \mathcal{A}$. For an ideological attribute $i \in \mathcal{I}$, we take the terms with positive elements of this vector to be the cues for ideology i; call this set $\mathcal{L}(i)$ and let $\mathcal{L} = \bigcup_{i \in \mathcal{I}} \mathcal{L}(i)$.

Because political texts use a fair amount of multiword jargon, we initially represented each document as a bag of unigrams, bigrams, and trigrams, ignoring the fact that these "overlap" with each other.[4] While this would be inappropriate in language modeling and is inconsistent with our model's independence assumptions among words, it is sensible since our goal is to identify cues that are statistically associated with attributes like ideologies.

Preliminary trials revealed that unigrams tend to dominate in such a model, since their frequency counts are so much higher. Further, domain experts found them harder to interpret out of context compared to bigrams and trigrams. We therefore included only bigrams and trigrams as terms in our cue discovery model.

2.4 Validation

The term selection method we have described can be understood as a form of feature selection that reasons globally about the data and tries to control for some effects that are not of interest (topic or document idiosyncrasies). We compared the approach to two classic, simple methods for feature selection: ranking based on pointwise mutual information (PMI) and weighted average PMI (WAPMI) (Schneider, 2005; Cover and Thomas, 2012). Selected features were used to classify the ideologies of held-out documents from our corpus.[5] We evaluated these feature selection methods within naïve Bayes classification in a 5-fold cross-validation setup. We vary λ for the SAGE model and compare the results to equal-sized sets of terms selected by PMI and WAPMI. We consider SAGE with and without topic effects.

Figure 2 visualizes accuracy against the number of features for each method. Bigrams and trigrams consistently outperform unigrams (McNemar's, $p < 0.05$). Otherwise, there are no significant differences in performance except WAPMI

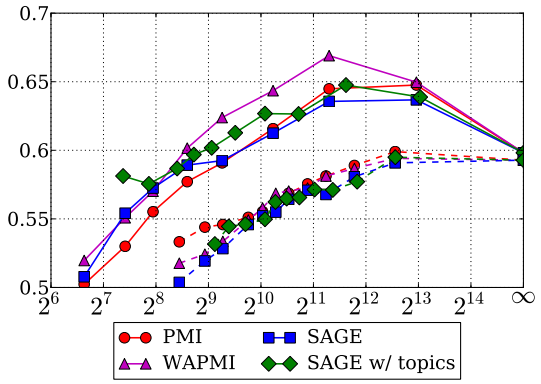


Figure 2: Plot of average classification accuracy for 5-fold cross validation against the number of features. Dashed lines refer to using only unigram features, while solid lines refer to using bigram and trigram features.

with bigrams/trigrams at its highest point. SAGE with topics is slightly (but not significantly) better than without. We conclude that SAGE is a competitive choice for cue discovery, noting that a principled way of controlling for topical and document effects—offered by SAGE but not the other methods—may be even more relevant to our task than classification accuracy.

2.5 Cue Lexicon

We ran SAGE on the the full ideological book corpus, including topic effects, and setting $\lambda = 30$, obtained a set of $|\mathcal{L}| = 8,483$ cue terms. The supplementary materials include top cue terms associated with various ideologies and a heatmap of similarities among SAGE vectors.

We conducted a small, relatively informal study in which seven subjects (including four scholars of American politics) were asked to match brief descriptions of the classes, including prominent prototypical individuals exemplifying each, to cue terms. About 70% of ideologies were correctly matched by experts, with relatively few confusions between LEFT and RIGHT. More details are given in supplementary materials.

3 Second Stage: Cue-Lag Ideological Proportions

The main contribution of this paper is a technique for measuring ideology proportions in the prose of political candidates. We adopt a Bayesian approach that manages our uncertainty about the cue lexi-

[4]Generative models that produce the same evidence more than once are sometimes called "deficient," but model deficiency does not necessarily imply that the model is ineffective. Some of the IBM models for statistical machine translation provide a classic example (Brown et al., 1993).

[5]The text was tokenized and stopwords removed. Punctuation, numbers, and web addresses were normalized. Tokens appearing less than 20 times in training data, or in fewer than 5 documents were removed.

con $\mathcal{L}$, the tendencies of political speakers to "flip-flop" among ideological types, and the relative "distances" among different ideologies. The representation of a candidate's ideology as a mixture among discrete, hierarchically related categories can be distinguished from continuous representations ("scaling" or "spatial" models) often used in political science, especially to infer positions from Congressional roll-call voting patterns (Poole and Rosenthal, 1985; Poole and Rosenthal, 2000; Clinton et al., 2004). Moreover, the ability to draw inferences about individual policy-makers' ideologies from their votes on proposed legislation is severely limited by institutional constraints on the types of legislation that is actually subject to recorded votes.

3.1 Political Speeches Corpus

We gathered transcribed speeches given by candidates of the two main parties (Democrats and Republicans) during the 2008 and 2012 Presidential election seasons. Each election season is comprised of two stages: (i) the primary elections, where candidates seek the support of their respective parties to be nominated as the party's Presidential candidate, and (ii) the general elections where the parties' chosen candidates travel across the states to garner support from all citizens. Each candidate's speeches are partitioned into *epochs* for each election; e.g., those that occur before the candidate has secured enough pledged delegates to win the party nomination are "from the primary." Table 2 presents a breakdown of the candidates and speeches in our corpus.

3.2 Cue-Lag Representation

Our measurement model only considers ideological cues; other terms are treated as filler. We therefore transform each speech into a **cue-lag** representation.

The representation is a sequence of alternating cues (elements from the ideological lexicon $\mathcal{L}$) and integer "lags" (counts of non-cue terms falling between two cues). This will allow us to capture the intuition that a candidate may use longer lags between evocations of different ideologies, while nearby cues are likely to be from similar ideologies.

To map a speech into the cue-lag representation, we simply match all elements of $\mathcal{L}$ in the speech and replace sequences of other words by their lengths. When a trigram cue strictly includes a bigram cue, we take only the trigram. When two cues partially overlap, we treat them as consecutive cue terms and set the lag to 0. Figure 3 shows an example of our cue-lag representation.

Party	Pri'08	Gen'08	Pri'12	Gen'12
Democrats*	167	-	-	-
Republicans†	50	-	49	-
Obama (D)	78	81	-	99
McCain (R)	9	159	-	-
Romney (R)	8	‡(13)	19	19

*Democrats in our corpus are: Joe Biden, Hillary Clinton, John Edwards, and Bill Richardson in 2008 and Barack Obama in both 2008 and 2012.

†Republicans in our corpus are: Rudy Giuliani, Mike Huckabee, John McCain, and Fred Thompson in 2008, Michelle Bachmann, Herman Cain, Newt Gingrich, Jon Huntsman, Rick Perry, and Rick Santorum in 2012, and Ron Paul and Mitt Romney in both 2008 and 2012.

‡For Romney, we have 13 speeches which he gave in the period 2008-2011 (between his withdrawal from the 2008 elections and before the commencement of the 2012 elections). While these speeches are not technically part of the regular Presidential election campaign, they can be seen as his preparation towards the 2012 elections, which is particularly interesting as Romney has been accused of having inconsistent viewpoints.

Table 2: Breakdown of number of speeches in our political speech corpus by epoch. On average, 2,998 tokens, and 95 cue terms are found in each speech document.

3.3 CLIP: An Ideology HMM

The model we use to infer ideologies, **cue-lag ideological proportions** (CLIP), is a hidden Markov model. Each state corresponds to an ideology (Fig. 1) or BACKGROUND. The emission from a state consists of (i) a cue from $\mathcal{L}$ and (ii) a lag value. The high-level generative story for a single speech with T cue-lag pairs is as follows:

1. Parameters are drawn from conjugate priors (details in §3.3.3).
2. Let the initial state be the BACKGROUND state.
3. For $t \in \{1, 2, \ldots, T\}$:[6]
   (a) Transition to state S_t based on the transition distribution, discussed in §3.3.1. This transition is conditioned on the previous state S_{t-1} and the lag at timestep $t-1$, denoted by L_{t-1}.

[6]The length of the sequence is assumed to be exogenous, so that no stop state needs to be defined.

Original sentence	Just compare this President's record with **Ronald Reagan's** first term. **President Reagan** also faced an **economic crisis**. In fact, in 1982, the **unemployment rate** peaked at nearly 11 percent. But in the two years that followed, he delivered a true recovery **economic growth** and **job creation** were three times higher than in the Obama Economy.
Cue-lag representation	$\ldots \xrightarrow{6}$ ronald_reagan $\xrightarrow{2}$ presid_reagan $\xrightarrow{3}$ econom_crisi $\xrightarrow{5}$ unemploy_rate $\xrightarrow{17}$ econom_growth $\xrightarrow{1}$ job_creation $\xrightarrow{9} \ldots$

Figure 3: Example of the cue-lag representation.

(b) Emit cue term W_t from the lexicon $\mathcal{L}$ and lag L_t based on the emission distribution, discussed in §3.3.2.

We turn next to the transitions and emissions.

3.3.1 Ideology Topology and Transition Parameterization

CLIP assumes that each cue term uttered by a politician is generated from a hidden state corresponding to an ideology. The ideologies are organized into a tree based on their hierarchical relationships; see Fig. 1. In this study, the tree is fixed according to our domain knowledge of current American politics; in future work it might be enriched with greater detail or its structure learned automatically.

The ideology tree is used in defining the transition distribution in the HMM, but not to directly define the topology of the HMM. Importantly, each state may transition to any other state, but the transition *distribution* is defined using the graph, so that ideologies that are closer to each other will tend to be more likely to transition to each other. To transition between two states s_i and s_j, a walk must be taken in the tree from vertex s_i to vertex s_j. We emphasize that the walk corresponds to a *single* transition—the speaker does not emit anything from the states passed through along the path.

A simplified version of our transition distribution, for exposition, is given as follows:

$$p_{tree}(s_j \mid s_i; \boldsymbol{\zeta}, \boldsymbol{\theta}) = \left(\prod_{\langle u,v\rangle \in Path(s_i,s_j)} (1-\zeta_u)\theta_{u,v}\right) \zeta_{s_j}$$

$Path(s_i, s_j)$ refers to the sequence of edges in the tree along the unique path from s_i to s_j. Each of these edges $\langle u, v\rangle$ must be traversed, and the probability of doing so, conditioned on having already reached u, is $(1-\zeta_u)$—i.e., not stopping in u—times $\theta_{u,v}$—i.e., selecting vertex v from among those that share an edge with u. Eventually, s_j is reached, and the walk ends, incurring probability ζ_{s_j}.

In order to capture the intuition that a longer lag after a cue term should increase the entropy over the next ideology state, we introduce a **restart** probability, which is conditioned on the length of the most recent lag, ℓ. The probability of restarting the walk from the BACKGROUND state is a noisy-OR model with parameter ρ. This gives the transition distribution:

$$p(s_j \mid s_i, \ell; \boldsymbol{\zeta}, \boldsymbol{\theta}, \rho) = (1-\rho)^{\ell+1} p_{tree}(s_j \mid s_i; \boldsymbol{\zeta}, \boldsymbol{\theta}) + (1-(1-\rho)^{\ell+1}) p_{tree}(s_j \mid s_{\text{BACKGROUND}}; \boldsymbol{\zeta}, \boldsymbol{\theta})$$

Note that, if $\rho = 1$, there is no Markovian dependency between states (i.e., there is always a restart), so CLIP reverts to a mixture model.

This approach allows us to parameterize the full set of $|\mathcal{I}|^2$ transitions with $O(|\mathcal{I}|)$ parameters.[7] Since the graph is a tree and the walks are not allowed to backtrack, the only ambiguity in the transition is due to the restart probability; this distinguishes CLIP from other algorithms based on random walks (Brin and Page, 1998; Mihalcea, 2005; Toutanova et al., 2004; Collins-Thompson and Callan, 2005).

3.3.2 Emission Parameterization

Recall that, at time step t, CLIP emits a cue from the lexicon $\mathcal{L}$ and an integer-valued lag. For each state s, we let the probability of emitting cue w be denoted by $\psi_{s,w}$; $\boldsymbol{\psi}_s$ is a multinomial distribution over the entire lexicon $\mathcal{L}$. This allows our approach to handle ambiguous cues that can associate with more than one ideology, and also to associate a cue with a different ideology than our cue discovery method proposed, if the signal from the data is sufficiently strong. We assume each lag to be generated by a Poisson distribution with global parameter ν.

[7] More precisely, there are $|\mathcal{I}|$ edges (since there are $|\mathcal{I}|+1$ vertices including BACKGROUND), each with a $\boldsymbol{\theta}$-parameter in each direction. For a vertex with degree d, however, there are only $d-1$ degrees of freedom, so that there are $2|\mathcal{I}|-(|\mathcal{I}|+1) = |\mathcal{I}|-1$ degrees of freedom for $\boldsymbol{\theta}$. There are $|\mathcal{I}|$ $\boldsymbol{\zeta}$-parameters and a single ρ, for a total of $2|\mathcal{I}|$ degrees of freedom.

3.3.3 Inference and Learning

Above we described CLIP's transitions and emissions. Because our interest is in measuring proportions—and, as we will see, in *comparing* those proportions across speakers and campaign periods—we require a way to allow variation in parameters across different conditions. Specifically, we seek to measure differences in time spent in each ideology state. This can be captured by allowing each speaker to have a different $\boldsymbol{\theta}$ and $\boldsymbol{\zeta}$ in each stage of the campaign. On the other hand, we expect that a speaker draws from his ideological lexicon similarly across different epochs—there is a single $\boldsymbol{\psi}$ shared between different epochs.

In order to manage uncertainty about the parameters of CLIP, to incorporate prior beliefs based on our ideology-specific cue lexicons $\{\mathcal{L}(i)\}_i$, and to allow sharing of statistical strength across conditions, we adopt a Bayesian approach to inference. This will allow principled exploration of the posterior distribution over the proportions of interest.

We place a symmetric Dirichlet prior on the tree walk probabilities $\boldsymbol{\theta}$; its parameter is α. For the cue emission distribution associated with ideology i, $\boldsymbol{\psi}_{s_i}$, we use an *informed* Dirichlet prior with two different values, β_{cue} for cues in $\mathcal{L}(i)$, and a smaller β_{def} for those in $\mathcal{L} \setminus \mathcal{L}(i)$.[8]

Learning proceeds by collapsed Gibbs sampling for the hidden states and slice sampling (with vague priors) for the hyperparameters (α, $\boldsymbol{\beta}$, ρ, and $\boldsymbol{\zeta}$). Details of the sampler are given in the supplementary materials. At each Gibbs step, we resample the ideology state and restart indicator variable for every cue term in every speech.

We ran our Gibbs sampler for 75,000 iterations, discarding the first 25,000 iterations for burn-in, and collected samples at every 10 iterations. Further, we perform the slice sampling step at every 5,000 iterations. For each candidate, we collected 5,000 posterior samples which we use to infer his/her ideological proportions.

In order to determine the amount of time a candidate spends in each ideology, we denote the unit of time in terms of half the lag before and after each cue term, i.e., when a candidate draws a cue term from ideology i during timestep t, we say that he spends $\frac{1}{2}(L_{t-1} + L_t)$ amount of time in ideology i. Averaging over all the samples returned by our sampler and normalizing it by the length of the documents in each epoch, we obtain a candidate's expected ideological proportions within the epoch.

[8] This implies that a term can, in the posterior distribution, be associated with an ideology i of whose $\mathcal{L}(i)$ it was not a member. In fact, this occurred frequently in our runs of the model.

4 Pre-registered Hypotheses

The traditional way to evaluate a text analysis model in NLP is, of course, to evaluate its output against gold-standard judgements by humans. In the case of recent political speeches, however, we are doubtful that such judgments can be made objectively at a fine-grained level. While we are confident about gross categorization of books and magazines in our ideological corpus (§2.1), many of which are *overtly* marked by their ideological assocations, we believe that human estimates of ideological proportions, or even association of particular tokens with ideologies they may evoke, may be overly clouded by the variation in annotator ideology and domain expertise.

We therefore adopt a different method for evaluation. Before running our model, we identified a set of hypotheses, which we **pre-registered** as expectations. These are categorized into groups based on their strength and relevance to judging the validity of the model. *Strong* hypotheses are those that constitute the lowest bar for face validity; if violated, they suggest a flaw in the model. *Moderate* hypotheses are those that match the intuition of domain experts conducting the research, or extant theory. Violations suggest more examination is required, and may raise the possibility that further testing might be pursued to demonstrate the hypothesis is false. Our 13 principal hypotheses are enumerated in Table 3.

5 Evaluation

We compare the posterior proportions inferred by CLIP with several baselines:

- HMM: rather than §3.3.1, a fully connected, traditional transition matrix is used.
- MIX: a mixture model; at each timestep, we *always* restart ($\rho = 1$). This eliminates Markovian dependencies between ideologies at nearby timesteps, but still uses the ideology tree in defining the probabilities of each state through $\boldsymbol{\theta}$.

Hypotheses	CLIP	HMM	MIX	NORES
Sanity checks (strong):				
S1. Republican primary candidates should tend to draw more from RIGHT than from LEFT.	*12/12	10/13	13/13	12/13
S2. Democratic primary candidates should tend to draw more from LEFT than from RIGHT.	4/5	5/5	5/5	5/5
S3. In general elections, Democrats should draw more from the LEFT than the Republicans and vice versa for the RIGHT.	4/4	4/4	3/4	0/4
S total	20/21	19/22	21/22	17/22
Primary hypotheses (strong):				
P1. Romney, McCain and other Republicans should almost never draw from FAR LEFT, and extremely rarely from PROGRESSIVE.	29/32	*21/31	27/32	29/32
P2. Romney should draw more heavily from the RIGHT than Obama in both stages of the 2012 campaign.	2/2	2/2	1/2	1/2
Primary hypotheses (moderate):				
P3. Romney should draw more heavily on words from the LIBERTARIAN, POPULIST, RELIGIOUS RIGHT, and FAR RIGHT in the primary compared to the general election. In the general election, Romney should draw more heavily on CENTER, CENTER–RIGHT and LEFT vocabularies.	2/2	2/2	0/2	2/2
P4. Obama should draw more heavily on words from the PROGRESSIVE in the 2008 primary than in the 2008 general election.	0/1	0/1	0/1	1/1
P5. In the 2008 general election, Obama should draw more heavily on the CENTER, CENTER–LEFT, and RIGHT vocabularies than in the 2008 primary.	1/1	1/1	1/1	1/1
P6. In the 2012 general election, Obama should sample more from the LEFT than from the RIGHT, and should sample more from the LEFT vocabularies than Romney.	2/2	2/2	0/2	0/2
P7. McCain should draw more heavily from the FAR RIGHT, POPULIST, and LIBERTARIAN in the 2008 primary than in the 2008 general election.	0/1	1/1	1/1	1/1
P8. In the general 2008, McCain should draw more heavily from the CENTER, CENTER–RIGHT, and LEFT vocabularies than in the 2008 primary.	1/1	1/1	1/1	1/1
P9. McCain should draw more heavily from the RIGHT than Obama in both stages of the campaign.	2/2	2/2	2/2	1/2
P10.Obama and other Democrats should very rarely draw from FAR RIGHT.	6/7	5/7	7/7	4/7
P total	45/51	37/50	40/51	41/51

Table 3: Pre-registered hypotheses used to validate the measurement model; number of statements evaluated correctly by different models. *Some differences were not significant at $p = 0.05$ and are not included in the results.

- NORES, where we *never* restart ($\rho = 0$). This strengthens the Markovian dependencies.

In MIX, there are no temporal effects between cue terms, although the structure of our ideology tree encourages the speaker to draw from coarse-grained ideologies over fine-grained ideologies. On the other hand, the strong Markovian dependency between states in NORES would encourage the model to stay local within the ideology tree. In our experiments, we will see how that the ideology tree and the random treatment of restarting both contribute to our model's inferences.

Table 3 presents a summary of which hypotheses the models' inferences are in accordance with. CLIP is not consistently outperformed by any of the competing baselines.

Sanity checks (S1–3) CLIP correctly identifies sixteen LEFT/RIGHT alignments of primary candidates (S1, S2), but is unable to determine one candidate's orientation; it finds Jon Huntsman to spend roughly equal proportions of speech-time drawing on LEFT and RIGHT cue terms. Interestingly, Huntsman, who had served as U.S. Ambassador to China under Obama, was considered the one moderate in the 2012 Republican field. MIX correctly identifies all thirteen Republicans, while NORES places McCain from the 2008 primaries as mostly LEFT-leaning and HMM misses three of thirteen, including Perry and Gingrich, who might be deeply

disturbed to find that they are misclassified as LEFT-leaning. As for the Democratic primary candidates (S2), CLIP's one questionable finding is that John Edwards spoke slightly more from the RIGHT than the LEFT. For the general elections (S3), CLIP and HMM correctly identify the relative amount of time spent in LEFT/RIGHT between Obama and his Republican competitors. NORES had the most trouble, missing all four. CLIP finds Obama spending slightly more time on the RIGHT than on the LEFT in the 2008 general elections but nevertheless, Obama is still found to spend more time engaging in LEFT-speak than McCain.

Name interference When we looked at the cue terms actually used in the speeches, we found one systematic issue: the inclusion of candidates' names as cue terms. Terms mentioning John McCain are associated with the RIGHT, so that Obama's mentions of his opponent are taken as evidence for rightward positioning; in total, mentions of McCain contributed 4% absolute to Obama's RIGHT ideological proportion. Similarly, *barack_obama* and *presid_obama* are LEFT cues (though *senat_obama* is a RIGHT cue). In future work, we believe filtering candidate names in the first stage will be beneficial.

Strong hypotheses P1 and P2 CLIP and the variants making use of the ideology tree were in agreement on most of the strong primary hypotheses. Most of these involved our expectation that the Republican candidates would rarely draw on FAR LEFT and PROGRESSIVE LEFT. Our qualitative hypotheses were not specific about how to quantify "rare" or "almost never." We chose to find a result inconsistent with a P1 hypothesis any time a Republican had proportions greater than 5% for either ideology. The notable deviations for CLIP were Fred Thompson (13% from the PROGRESSIVE LEFT during the 2008 primary) and Mitt Romney (12% from the PROGRESSIVE LEFT between the 2008 and 2012 elections, 13% from the FAR LEFT during the 2012 general election). This model did no worse than other variants here and much better than one: HMM had 10 inconsistencies out of 32 opportunities, suggesting the importance of the ideology tree.

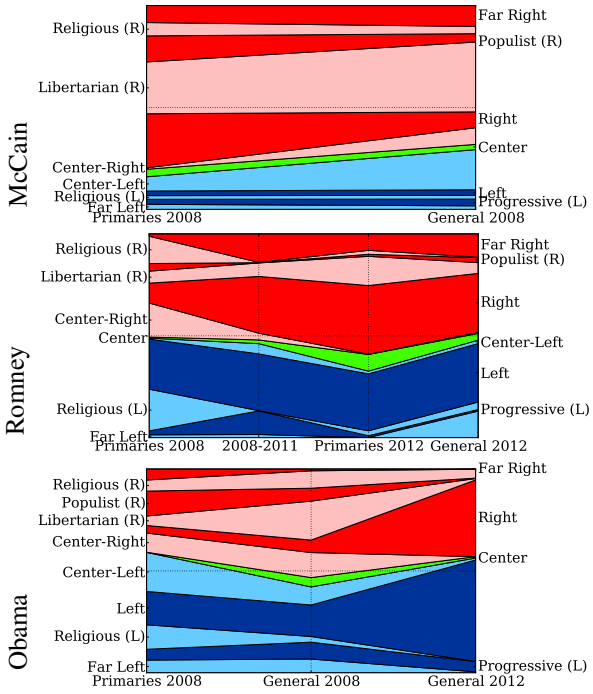


Figure 4: Proportion of time spent in each ideology by McCain, Romney, and Obama during the 2008 and 2012 Presidential election seasons.

"Etch-a-Sketch" hypotheses Hypotheses P3, P4, P5, P7, and P8 are all concerned with differences between the primary and general elections: successful primary candidates are expected to "move to the center." A visualization of CLIP's proportions for McCain, Romney, and Obama is shown in Figure 4, with their speeches grouped together by different epochs. The model is in agreement with most of these hypotheses. It did not confirm P4—Obama appears to CLIP to be more PROGRESSIVE in the 2008 general election than in the primary, though the difference is small (3%) and may be within the margin of error. Likewise, in P7, the difference between McCain drawing from FAR RIGHT, POPULIST and LIBERTARIAN between the 2008 primary and general elections is only 2% and highly uncertain, with a 95% credible interval of 44–50% during the primary (vs. 47–50% in the general election).

Fine-grained ideologies Fine-grained ideologies are expected to account for smaller proportions, so that making predictions about them is quite difficult. This is especially true for primary elections, where a broader palette of ideologies is expected to be drawn from, but we have fewer speeches from each candi-

date. CLIP's inconsistency with P10, for example, comes from assigning 5.4% of Obama's 2008 primary cues to FAR RIGHT.

CLIP's inferences on the corpus of political speeches can be browsed at http://www.ark.cs.cmu.edu/CLIP. We emphasize that CLIP and its variants are intended to quantify the ideological content candidates express in *speeches*, not necessarily their *beliefs* (which may not be perfectly reflected in their words), or even how they are described by pundits and analysts (who draw on far more information than is expressed in speeches). CLIP's deviations from the hypotheses are suggestive of potential improvements to cue extraction (§2), but also of incorrect hypotheses. We expect future research to explore a richer set of linguistic cues and attributes beyond ideology (e.g., topics and framing on various issues). We plan to use CLIP as a text analysis method to support substantive inquiry in political science, such as following trends in expressed ideology over time.

6 Related Work

As early as the 1960s, there has been research on modeling ideological beliefs using automated systems (Abelson and Carroll, 1965; Carbonell, 1978; Sack, 1994). These early works model ideology at a sophisticated level, involving the actors, actions and goals; they require manually constructed knowledge bases. Poole and Rosenthal (1985) used congressional roll call data to demonstrate the ideological divide in Congress, and provided a methodology for measuring ideological positions. Gerrish and Blei (2011; 2012) augmented the methodology with text from congressional bills using probabilistic models to uncover lawmakers' positions on specific political issues, putting them on a left-right spectrum, while Thomas et al. (2006) made use of floor debate speeches to predict votes. Likewise, taking advantage of the proliferation of text today, numerous techniques have been developed to identify topics and perspectives in the media (Gentzkow and Shapiro, 2005; Lin et al., 2008; Fortuna et al., 2009; Gentzkow and Shapiro, 2010); determine the political leanings of a document or author (Laver et al., 2003; Efron, 2004; Mullen and Malouf, 2006; Fader et al., 2007); or recognize stances in debates (Somasundaran and Wiebe, 2009; Anand et al., 2011). Going beyong lexical indicators, Greene and Resnik (2009) investigated syntactic features to identify perspectives or implicit sentiment.

7 Conclusions

We introduced CLIP, a domain-informed, Bayesian model of ideological proportions in political language. We showed how ideological cues could be discovered from a lightly labeled corpus of ideological writings, then incorporated into CLIP. The resulting inferences are largely consistent with a set of preregistered hypotheses about candidates in the 2008 and 2012 Presidential elections.

Acknowledgments

For thoughtful feedback on this research, the authors thank: several anonymous reviewers, Amber Boydstun, Philip Resnik, members of the ARK group at CMU, and participants in Princeton University's Political Methodology Colloquium and PolMeth XXX hosted by The University of Virginia. This work was supported in part by an A*STAR fellowship to Y. Sim, NSF grants IIS-1211201 and IIS-1211277, and Google's support of the Reading is Believing project at CMU.

References

Robert P. Abelson and J. Douglas Carroll. 1965. Computer simulation of individual belief systems. *American Behavioral Scientist*, 8(9):24–30.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the Second Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l_1-regularized log-linear models. In *Proceedings of ICML*.

Duncan Black. 1948. On the rationale of group decision-making. *The Journal of Political Economy*, 56(1):23–34.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117.

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

Jaime G. Carbonell. 1978. Politics: Automated ideological reasoning. *Cognitive Science*, 2(1):27–51.

Jonathan Charteris-Black. 2005. *Politicians and Rhetoric: The Persuasive Power of Metaphor*. Palgrave-MacMillan.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. The statistical analysis of roll call data. *American Political Science Review*, 98(2):355–370.

Kevyn Collins-Thompson and Jamie Callan. 2005. Query expansion using random walk models. In *Proceedings of CIKM*.

Thomas M. Cover and Joy A. Thomas. 2012. *Elements of Information Theory*. Wiley-Interscience.

Daniel Deirmeier, Jean-Francois Godbout, Bei Yu, and Stefan Kaufmann. 2012. Language and ideology in congress. *British Journal of Political Science*, 42(1):31–55.

Anthony Downs. 1957. *An Economic Theory of Democracy*. Harper, New York.

Miles Efron. 2004. Cultural orientation: Classifying subjective documents by cociation analysis. In *AAAI Fall Symposium on Style and Meaning in Language, Art, and Music*.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text. In *Proceedings of ICML*.

Anthony Fader, Dragomir R. Radev, Michael H. Crespin, Burt L. Monroe, Kevin M. Quinn, and Michael Colaresi. 2007. MavenRank: Identifying influential members of the US senate using lexical centrality. In *Proceedings of EMNLP-CoNLL*.

Blaz Fortuna, Carolina Galleguillos, and Nello Cristianini. 2009. Detecting the bias in media with statistical learning methods. In Ashok N. Srivastava and Mehran Sahami, editors, *Text Mining: Classification, Clustering, and Applications*, chapter 2, pages 27–50. Chapman & Hall/CRC.

Matthew Gentzkow and Jesse Shapiro. 2005. Media bias and reputation. Technical report, National Bureau of Economic Research.

Matthew Gentzkow and Jesse M. Shapiro. 2010. What drives media slant? evidence from u.s. daily newspapers. *Econometrica*, 78(1):35–71.

Sean M. Gerrish and David M. Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of ICML*.

Sean M. Gerrish and David M. Blei. 2012. How they vote: Issue-adjusted models of legislative behavior. In *Advances in NIPS 25*.

Stephan Greene and Philip Resnik. 2009. More than words: syntactic packaging and implicit sentiment. In *Proceedings of NAACL*.

Roderick P. Hart, Jay P. Childers, and Colene J. Lind. 2013. *Political Tone: How Leaders Talk and Why*. University of Chicago Press.

Roderick P. Hart. 2009. *Campaign talk: Why elections are good for us*. Princeton University Press.

Dustin Hillard, Stephen Purpura, and John Wilkerson. 2008. Computer-assisted topic classification for mixed-methods social science research. *Journal of Information Technology & Politics*, 4(4):31–46.

Harold Hotelling. 1929. Stability in competition. *The Economic Journal*, 39(153):41–57.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *The American Political Science Review*, 97(2):311–331.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Proceedings of ECML-PKDD*.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of EMNLP*.

Burt L. Monroe and Ko Maeda. 2004. Talk's cheap: Text-based estimation of rhetorical ideal-points. Presented at the Annual Meeting of the Society for Political Methodology.

Tony Mullen and Robert Malouf. 2006. A preliminary investigation into sentiment analysis of informal political discourse. In *AAAI Symposium on Computational Approaches to Analysing Weblogs*.

Keith T. Poole and Howard Rosenthal. 1985. A spatial model for legislative roll call analysis. *American Journal of Political Science*, 29(2):357–384.

Keith T. Poole and Howard Rosenthal. 2000. *Congress: A Political-Economic History of Roll Call Voting*. Oxford University Press.

Warren Sack. 1994. Actor-role analysis: ideology, point of view, and the news. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Karl-Michael Schneider. 2005. Weighted average pointwise mutual information for feature selection in text categorization. In *Proceedings of PKDD*.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: determining support or opposition from congressional floor-debate transcripts. In *Proceedings of EMNLP*.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *Proceedings of ICML*.

Learning to Freestyle: Hip Hop Challenge-Response Induction via Transduction Rule Segmentation

Dekai Wu Karteek Addanki Markus Saers Meriem Beloucif
Human Language Technology Center
Department of Computer Science
HKUST, Clear Water Bay, Hong Kong
{dekai|vskaddanki|masaers|mbeloucif}@cs.ust.hk

Abstract

We present a novel model, Freestyle, that learns to improvise rhyming and fluent responses upon being challenged with a line of hip hop lyrics, by combining both bottom-up token based rule induction and top-down rule segmentation strategies to learn a stochastic transduction grammar that simultaneously learns both phrasing and rhyming associations. In this attack on the woefully under-explored natural language genre of music lyrics, we exploit a strictly unsupervised transduction grammar induction approach. Our task is particularly ambitious in that no use of any *a priori* linguistic or phonetic information is allowed, even though the domain of hip hop lyrics is particularly noisy and unstructured. We evaluate the performance of the learned model against a model learned only using the more conventional bottom-up token based rule induction, and demonstrate the superiority of our combined token based and rule segmentation induction method toward generating higher quality improvised responses, measured on fluency and rhyming criteria as judged by human evaluators. To highlight some of the inherent challenges in adapting other algorithms to this novel task, we also compare the quality of the responses generated by our model to those generated by an out-of-the-box phrase based SMT system. We tackle the challenge of selecting appropriate training data for our task via a dedicated rhyme scheme detection module, which is also acquired via unsupervised learning and report improved quality of the generated responses. Finally, we report results with Maghrebi French hip hop lyrics indicating that our model performs surprisingly well with no special adaptation to other languages.

1 Introduction

The genre of lyrics in music has been severely understudied from the perspective of computational linguistics despite being a form of language that has perhaps had the most impact across almost all human cultures. With the motivation of spurring further research in this genre, we apply stochastic transduction grammar induction algorithms to address some of the modeling issues in song lyrics. An ideal starting point for this investigation is hip hop, a genre that emphasizes rapping, spoken or chanted rhyming lyrics against strong beats or simple melodies. Hip hop lyrics, in contrast to poetry and other genres of music, present a significant number of challenges for learning as it lacks well-defined structure in terms of rhyme scheme, meter, or overall meaning making it an interesting genre to bring to light some of the less studied modeling issues.

The domain of hip hop lyrics is particularly unstructured when compared to classical poetry, a domain on which statistical methods have been applied in the past. Hip hop lyrics are unstructured in the sense that a very high degree of variation is permitted in the meter of the lyrics, and large amounts of colloquial vocabulary and slang from the subculture are employed. The variance in the permitted meter makes it hard to make any assumptions about the stress patterns of verses in order to identify the rhyming words used when generating output. The broad range of unorthodox vocabulary used in hip hop make it difficult to use off-the-shelf NLP tools for doing phonological and/or morphological analysis. These problems are further exacerbated by differences in intonation of the same word and lack of robust transcription (Liberman, 2010).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 102–112,
Seattle, Washington, USA, 18-21 October 2013.

We argue that stochastic transduction grammars,[1] given their success in the area of machine translation and efficient unsupervised learning algorithms, are ideal for capturing the structural relationship between lyrics. Hence, our FREESTYLE system models the problem of improvising a rhyming response given any hip hop lyric challenge as transducing a challenge line into a rhyming response. We use a stochastic transduction grammar induced in a completely unsupervised fashion using a combination of token based rule induction and segmenting (Saers *et al.*, 2013) as the underlying model to fully-automatically learn a challenge-response system and compare its performance against a simpler token based transduction grammar model. Both our models are completely unsupervised and use no prior phonetic or linguistic knowledge whatsoever despite the highly unstructured and noisy domain.

We believe that the challenge-response system based on an interpolated combination of token based rule induction and rule segmenting transduction grammars will generate more fluent and rhyming responses compared to one based on token based transduction grammars models. This is based on the observation that token based transduction grammars suffer from a lack of fluency; a consequence of the degree of expressivity they permit. Therefore, as a principal part of our investigation we compare the quality of responses generated using a combination of token based rule induction and top-down rule segmenting transduction grammars to those generated by pure token based transduction grammars.

We also hypothesize that in order to generate fluent and rhyming responses, it is not sufficient to train the transduction grammars on all adjacent lines of a hip hop verse. Therefore, we propose a data selection scheme using a rhyme scheme detector acquired through unsupervised learning to generate the training data for the challenge-response systems. The rhyme scheme detector segments each verse of a hip hop song into stanzas and identifies the lines in each stanza that rhyme with each other which are then added as training instances. We demonstrate the superiority of our training data selection method by comparing the quality of the responses generated by the models trained on data selected with and without using the rhyme scheme detector.

Unlike conventional spoken and written language, disfluencies and backing vocals[2] occur very frequently in the domain of hip hop lyrics which affect the performance of NLP models designed for processing well-formed sentences. We propose two strategies to mitigate the effect of disfluencies on our model performance and compare their efficacy using human evaluations. Finally, in order to illustrate the challenges faced by other NLP algorithms, we contrast the performance of our model against a conventional, widely used phrase-based SMT model.

A brief terminological note: "stanza" and "verse" are frequently confused and sometimes conflated. Worse yet, their usage for song lyrics is often contradictory to that for poetry. To avoid ambiguity we consistently follow these technical definitions for segments in decreasing size of granularity:

verse a large unit of a song's lyrics. A song typically contains several verses interspersed with choruses. In the present work, we do not differentiate choruses from verses. In song lyrics, a verse is most commonly represented as a separate paragraph.

stanza a segment within a verse which has a meter and rhyme scheme. Stanzas often consist of 2, 3, or 4 lines, but stanzas of more lines are also common. Particularly in hip hop, a single verse often contains many stanzas with different rhyme schemes and meters.

line a segment within a stanza consisting of a single line. In poetry, strictly speaking this would be called a "verse", which however conflicts with the conventional use of "verse" in song lyrics.

In Section 2, we discuss some of the previous work that applies statistical NLP methods to less conventional domains and problems. We describe our experimental conditions in Section 3. We compare the performance of token and segment based transduction grammar models in Section 4. We compare our data selection schemes and disfluency handling strategies in Sections 5 and 6. Finally, in

[1] Also known in SMT as "synchronous grammars".

[2] Particularly the repetitive chants, exclamations, and interjections in hip hop "hype man" style backing vocals.

Section 7 we describe some preliminary results obtained using our approach on improvising hip hop responses in French and conclude in Section 8.

2 Related work

Although a few attempts have been made to apply statistical NLP learning methods to unconventional domains, Freestyle is among the first to tackle the genre of hip hop lyrics (Addanki and Wu, 2013; Wu *et al.*, 2013a,b). Our preliminary work suggested the need for further research to identify models that capture the correct generalizations to be able to generate fluent and rhyming responses. As a step towards this direction, we contrast the performance of interpolated bottom-up token based rule induction and top-down segmenting transduction grammar models and token based transduction grammar models. We briefly describe some of the past work in statistical NLP on unconventional domains below.

Most of the past work either uses some form of prior linguistic knowledge or enforces harsher constraints such as set number of words in a line, or a set meter which are warranted by more structured domains such as poetry. However, in hip hop lyrics it is hard to make any linguistic or structural assumptions. For example, words such as sho, flo, holla which frequently appear in the lyrics are not part of any standard lexicon and hip hop does not require a set number of syllables in a line, unlike poems. Also, surprising and unlikely rhymes in hip hop are frequently achieved via intonation and assonance, making it hard to apply prior phonological constraints.

A phrase based SMT system was trained to "translate" the first line of a Chinese couplet or *duilian* into the second by Jiang and Zhou (2008). The most suitable next line was selected by applying linguistic constraints to the n best output of the SMT system. However in contrast to Chinese couplets, which adhere to strict rules requiring, for example, an identical number of characters in each line and one-to-one correspondence in their metrical length, the domain of hip hop lyrics is far more unstructured and there exists no clear constraint that would ensure fluent and rhyming responses to hip hop challenge lyrics. Barbieri *et al.* (2012) use controlled Markov processes to semi-automatically generate lyrics that satisfy the structural constraints of rhyme and meter. Tamil lyrics were automatically generated given a melody using conditional random fields by A. *et al.* (2009). The lyrics were represented as a sequence of labels using the *KNM* system where *K*, *N* and *M* represented the long vowels, short vowels and consonants respectively.

Genzel *et al.* (2010) used SMT in conjunction with stress patterns and rhymes found in a pronunciation dictionary to produce translations of poems. Although many constraints were applied in translating full verses of poems, it was challenging to satisfy all the constraints. Stress patterns were assigned to words given the meter of a line in Shakespeare's sonnets by Greene *et al.* (2010), which were then combined with a language model to generate poems. Sonderegger (2011) attempted to infer the pronunciation of words in old English by identifying the rhyming patterns using graph theory. However, their heuristic of clustering words with similar IPA endings resulted in large clusters of false positives such as bloom and numb. A language-independent generative model for stanzas in poetry was proposed by Reddy and Knight (2011) via which they could discover rhyme schemes in French and English poetry.

3 Experimental conditions

Before introducing our Freestyle models, we first detail our experimental assumptions and the evaluation scheme under which the responses generated by different models are compared against one another. We describe our training data as well as a phrase-based SMT (PBSMT) contrastive baseline. We also define the evaluation scheme used to compare the responses of different systems on criteria of fluency and rhyming.

3.1 Training data

We used freely available user generated hip hop lyrics on the Internet to provide training data for our experiments. We collected approximately 52,000 English hip hop song lyrics amounting to approximately 800Mb of raw HTML content. The data was cleaned by stripping HTML tags, metadata and normalized for special characters and case differences. The processed corpus contained 22 million tokens with 260,000 verses and 2.7 million lines of hip hop lyrics. As human evaluation using expert hip hop

listeners is expensive, a small subset of 85 lines was chosen as the test set to provide challenges for comparing the quality of responses generated by different systems.

3.2 Evaluation scheme

The performance of various FREESTYLE versions was evaluated on the task of generating a improvised fluent and rhyming response given a single line of a hip hop verse as a challenge. The output of all the systems on the test set was given to three independent frequent hip hop listeners for manual evaluation. They were asked to evaluate the system outputs according to fluency and the degree of rhyming. They were free to choose the tune to make the lyrics rhyme as the beats of the song were not used in the training data. Each evaluator was asked to score the response of each system on the criterion of fluency and rhyming as being *good*, *acceptable* or *bad*.

3.3 Phrase-based SMT baseline

In order to evaluate the performance of an out-of-the-box phrase-based SMT (PBSMT) system toward this novel task of generating rhyming and fluent responses, a standard Moses baseline (Koehn *et al.*, 2007) was also trained in order to compare its performance with our transduction grammar induction model. A 4-gram language model which was trained on the entire training corpus using SRILM (Stolcke, 2002) was used to generate responses in conjunction with the phrase-based translation model. As no automatic quality evaluation metrics exist for hip hop responses analogous to BLEU for SMT, the model weights cannot be tuned in conventional ways such as MERT (Och, 2003). Instead, a slightly higher than typical language model weight was empirically chosen using a small development set to produce fluent outputs.

4 Interpolated segmenting model vs. token based model

We compare the performance of transduction grammars induced via interpolated token based and rule segmenting (ISTG) versus token based transduction grammars (TG) on the task of generating a rhyming and fluent response to hip hop challenges. We use the framework of stochastic transduction grammars, specifically bracketing ITGs (inversion transduction grammars) (Wu, 1997), as our translation model for "transducing" any given challenge into a rhyming and fluent response. Our choice is motivated by the significant amount of empirical evidence for the representational capacity of transduction grammars across a spectrum of natural language tasks such as textual entailment (Wu, 2006), mining parallel sentences (Wu and Fung, 2005) and machine translation (Zens and Ney, 2003). Further, existence of efficient learning algorithms (Saers *et al.*, 2012; Saers and Wu, 2011) that make no language specific assumptions, make inversion transduction grammars a suitable framework for our modeling needs. Examples of lexical transduction rules can be seen in Tables 3 and 5. In addition, the grammar also includes structural transduction rules for the straight case $A \rightarrow [A\ A]$ and also the inverted case $A \rightarrow \langle A\ A \rangle$.

4.1 Token based vs. segmental ITGs

The degenerate case of ITGs are token based ITGs wherein each translation rule contains at most one token in input and output languages. Efficient induction algorithms with polynomial run time exist for token based ITGs and the expressivity they permit has been empirically determined to capture most of the word alignments that occur across natural languages. The parameters of the token based ITGs can be estimated using expectation maximization through an efficient dynamic programming algorithm in conjunction with beam pruning (Saers and Wu, 2011).

In contrast to token based ITGs, each rule in a segmental ITG grammar can contain more than one token in both input and output languages. In machine translation applications, segmental models produce translations that are more fluent as they can capture lexical knowledge at a phrasal level. However, only a handful of purely unsupervised algorithms exist for learning segmental ITGs under matched training and testing assumptions. Most other approaches in SMT use a variety of ad hoc heuristics for extracting segments from token alignments, justified purely by short term improvements in automatic MT evaluation metrics such as BLEU (Papineni *et al.*, 2002) which cannot be transferred to our current task. Instead, we use a completely unsupervised learning algorithm for segmental ITGs that stays strictly within the transduction grammar optimization framework for both training and testing as proposed in Saers

et al. (2013).

Saers *et al.* (2013) induce a phrasal inversion transduction grammar via interpolating the bottom-up rule chunking approach proposed in Saers *et al.* (2012) with a top-down rule segmenting approach driven by a minimum description length objective function (Solomonoff, 1959; Rissanen, 1983) that trades off the maximum likelihood against model size. Saers *et al.* (2013) report improvements in BLEU score (Papineni *et al.*, 2002) on their translation task. In our current approach instead of using a bottom-up rule chunking approach we use a simpler token based grammar instead. Given two grammars (G_a and G_b) and an interpolation parameter α the probability function of the interpolated grammar is given by:

$$p_{a+b}(r) = \alpha p_a(r) + (1-\alpha) p_b(r)$$

for all rules r in the union of the two rule sets, and where p_{a+b} is the rule probability function of the combined grammar and p_a and p_b are the rule probability functions of G_a and G_b respectively. The pseudocode for the top-down rule segmenting algorithm is shown in 1. The algorithm uses the methods `collect_biaffixes`, `eval_dl`, `sort_by_delta` and `make_segmentations`. These methods collect all the biaffixes in an ITG, evaluate the difference in description length, sort candidates by these differences, and commit to a given set of candidates, respectively. The suitable interpolation parameter is chosen empirically based on the responses generated on a small development set.

We compare the performance of inducing a token based ITG versus inducing a segmental ITG using interpolated bottom-up token based rule induction and top-down rule segmentation. To highlight some of the inherent challenges in adapting other algorithms to this novel task, we also compare the quality of the responses generated by our model to those generated by an off-the-shelf phrase based SMT system.

Algorithm 1 Iterative rule segmenting learning driven by minimum description length.

```
Φ                                          ▷ The ITG being induced
repeat
    δ_sum ← 0
    bs ← collect_biaffixes(Φ)
    bδ ← []
    for all b ∈ bs do
        δ ← eval_dl(b, Φ)
        if δ < 0 then
            bδ ← [bδ, ⟨b, δ⟩]
    sort_by_delta(bδ)
    for all ⟨b, δ⟩ ∈ bδ do
        δ′ ← eval_dl(b, Φ)
        if δ′ < 0 then
            Φ ← make_segmentations(b, Φ)
            δ_sum ← δ_sum + δ′
until δ_sum ≥ 0
return Φ
```

4.2 Decoding heuristics

We use our in-house ITG decoder implemented according to the algorithm mentioned in Wu (1996) for the generating responses to challenges by decoding with the trained transduction grammars. The decoder uses a CKY-style parsing algorithm (Cocke, 1969) with cube pruning (Chiang, 2007). The decoder builds an efficient hypergraph structure which is then scored using the induced grammar. The trained transduction grammar model was decoded using the 4-gram language model and the model weights determined as described in 3.3.

In our decoding algorithm, we restrict the reordering to only be monotonic as we want to produce output that follows the same rhyming order of the challenge. Interleaved rhyming order is harder to evaluate without the larger context of the song and we do not address that problem in our current model. We also penalize singleton rules to produce responses of similar length as successive lines in a stanza are typically of similar length. Finally, we add a penalty to *reflexive* translation rules that map the same surface form to itself such as $A \rightarrow$ yo/yo. We obtain these rules with a high probability due to the presence of sentence pairs where both the input and output are identical strings as many stanzas in our data contain repeated chorus lines.

4.3 Results: Rule segmentation improves responses

Results in Table 1 indicate that the ISTG outperforms the TG model towards the task of generating fluent and rhyming responses. On the criterion of fluency,

Table 1: Percentage of ≥*good* and ≥*acceptable* (i.e., either good or acceptable) responses on fluency and rhyming criteria. PBSMT, TG and ISTG models trained using corpus generated from all adjacent lines in a verse. PBSMT+RS, TG+RS, ISTG+RS are models trained on rhyme scheme based corpus selection strategy. Disfluency correction strategy was used in all cases.

model	*fluency (≥good)*	*fluency (≥acceptable)*	*rhyming (≥good)*	*rhyming (≥acceptable)*
PBSMT	3.14%	4.70%	1.57%	4.31%
TG	21.18%	54.51%	23.53%	39.21%
ISTG	**26.27%**	**57.64%**	**27.45%**	**48.23%**
PBSMT+RS	30.59%	43.53%	1.96%	9.02%
TG+RS	**34.12%**	60.39%	20.00%	42.74%
ISTG+RS	30.98%	**61.18%**	**30.98%**	**53.72%**

Table 2: Transduction rules learned by ISTG model.

transduction grammar rule	*log prob.*
$A \rightarrow$ long/wrong	-11.6747
$A \rightarrow$ rhyme/time	-11.6604
$A \rightarrow$ felt bad/couldn't see what i really had	-11.3196
$A \rightarrow$ matter what you say/leaving anyway	-11.8792
$A \rightarrow$ arhythamatic/this rhythm is sick	-12.3492

the ISTG model produces a significantly higher fraction of sentences rated *good* (26.27% vs. 21.18%) and ≥*acceptable* (57.64% vs. 54.51%). Higher fraction of responses generated by the ISTG model are rated as *good* (27.45% vs. 23.53%) and ≥*acceptable* (57.64% vs. 54.51%) compared to the TG model. Both TG and ISTG model perform significantly better than the PBSMT baseline. Upon inspecting the learned rules, we noticed that the ISTG models capture rhyming correspondences both at the token and segmental levels. Table 2 shows some examples of the transduction rules learned by ISTG grammar trained using rhyme scheme detection.

5 Data selection via rhyme scheme detection vs. adjacent lines

We now compare two data selection approaches for generating the training data for transduction grammar induction via a rhyme scheme detection module and choosing all adjacent lines in a verse. We also briefly describe the training of the rhyme scheme detection module and determine the efficacy of our data selection scheme by training the ISTG model, TG model and the PBSMT baseline on training data generated with and without employing the rhyme scheme detection module. As the rule segmenting approach was intended to improve the fluency as opposed to the rhyming nature of the responses, we only train the rule segmenting model on the randomly chosen subset of all adjacent lines in the verse. Further, adding adjacent lines as the training data to the segmenting model maintains the context of the responses generated thereby producing higher quality responses. The segmental transduction grammar model was combined with the token based transduction grammar model trained on data selected with and without using rhyme scheme detection model.

5.1 Rhyme scheme detection

Although our approach adapts a transduction grammar induction model toward the problem of generating fluent and rhyming hip hop responses, it would be undesirable to train the model directly on all the successive lines of the verses—as done by Jiang and Zhou (2008)—due to variance in hip hop rhyming patterns. For example, adding successive lines of a stanza which follows **ABAB** rhyme scheme as training instances to the transduction grammar causes incorrect rhyme correspondences to be learned. The fact that a verse (which is usually represented as a separate paragraph) may contain multiple stanzas of varying length and rhyme schemes worsens this problem. Adding all possible pairs of lines in a verse as training examples not only introduces a lot of noise but also explodes the size of the training data due to the large size of the verse.

We employ a rhyme scheme detection model (Addanki and Wu, 2013) in order to select training instances that are likely to rhyme. Lines belonging to

the same stanza and marked as rhyming according to the rhyme scheme detection model are added to the training corpus. We believe that this data selection scheme will improve the rhyming associations learned during the transduction grammar induction thereby biasing the model towards producing fluent and rhyming output.

The rhyme scheme detection model proposes a HMM based generative model for a verse of hip hop lyrics similar to Reddy and Knight (2011). However, owing to the lack of well-defined verse structure in hip hop, a number of hidden states corresponding to stanzas of varying length are used to automatically obtain a *soft-segmentation* of the verse. Each state in the HMM corresponds to a stanza with a particular rhyme scheme such as **AA**, **ABAB**, **AAAA** while the emissions correspond to the final words in the stanza. We restrict the maximum length of a stanza to be four to maintain a tractable number of states and further only use states to represent stanzas whose rhyme schemes could not be partitioned into smaller schemes without losing a rhyme correspondence.

The parameters of the HMM are estimated using the EM algorithm (Devijer, 1985) using the corpus generated by taking the final word of each line in the hip hop lyrics. The lines from each stanza that rhyme with each other according to the Viterbi parse using the trained model are added as training instances for transduction grammar induction. As the source and target languages are identical, each selected pair generates two training instances: a challenge-response and a response-challenge pair.

The training data for the rhyme scheme detector was obtained by extracting the end-of-line tokens from each verse. However, upon data inspection we noticed that shorter lines in hip hop stanzas are typically joined with a comma and represented as a single line of text and hence all the tokens before the commas were also added to the training corpus. We obtained a corpus containing 4.2 million tokens corresponding to potential rhyming candidates comprising of around 153,000 unique token types.

We evaluated the performance of our rhyme scheme detector on the task of correctly labeling a given verse with rhyme schemes. As our model is completely unsupervised, we chose a random sample of 75 verses from our training data as our test set. Two native English speakers who were frequent hip hop listeners were asked to partition the verse into stanzas and assign them with a gold standard rhyme scheme. Precision and recall were aggregated for the Viterbi parse of each verse against this gold standard and f-score was calculated. The rhyme scheme detection module employed in our data selection obtained a precision of 35.81% and a recall of 57.25%, giving an f-score of 44.06%.

5.2 Training data selection via rhyme scheme detection

We obtained around 600,000 training instances upon extracting a training corpus using rhyme scheme detection module as described in Section 5.1. We added those lines that were adjacent *and* labeled as rhyming by the rhyme scheme detector as training instances resulting in a training corpus of size 200,000.

5.3 Training data selection via adjacent lines

Considering all adjacent lines in a verse resulted in a corpus with over 5 million training instances. In order to ensure fair comparison of models trained on data selected using rhyme scheme detection, we randomly chose 200,000 training instances from the generated corpus. The training corpus thus generated shared around 15% of training instances with the corpus generated through our proposed data selection scheme.

5.4 Results: Rhyme scheme detection helps

Results in Table 1 indicate that using the rhyme scheme detector for training data selection helps produce significantly more fluent responses compared to using adjacent lines. A possible explanation for this could be that adding all adjacent lines as training instances introduces a lot of noise into the model which hurts the fluency of the responses generated. Also, the cumulative fraction of sentences that were labeled *good* or ≥*acceptable* on the criterion of rhyming is larger when rhyme scheme detection was used to generate the training data (although the TG model trained on the corpus generated using adjacent lines produces a higher percentage of rhyming responses that were rated *good*). Given the significantly higher rate of response fluency when using rhyme scheme detection, we argue that using rhyme scheme detector for data selection is beneficial.

Table 3: English hip hop challenge-response examples.

challenge	man i **die** to **see** em all thun i just don't **care**
TG+RS	in the **sky** and **me** the in polla and the you **there**
TG	and the ride the me the and white the i the **air**
PBSMT+RS	man i live to see em all i just don't care
challenge	did a twelve year **bid** in the streets and held it **down**
TG+RS	to the girls here **kid** the and to the thought the **now**
TG	to the p's here **did** the a the i was the the
PBSMT+RS	did a year in the streets and it down
challenge	oh i believe in **yesterday**
TG+RS	can you see the **day**
TG	now you see the **way**
PBSMT+RS	oh i believe in tomorrow
challenge	what would i **do**
TG+RS	just me and **you**
TG	and you and **you**
PBSMT+RS	what would you do
challenge	cause you ain't going **home** till the early **morn**
TG+RS	and the you this **alone** i i gotta **on**
TG	and i you my on the a home we
PBSMT+RS	cause you and your friends aint nothing but

It is also interesting to note from Table 1 that ISTG+RS performs better than TG+RS indicating that transduction grammar induced via interpolating token based grammar and rule segmenting produces better responses than token based transduction grammar on both data selection schemes. Although the average fraction of responses rated *good* on fluency are slightly lower for ISTG+RS compared to TG+RS (34.12% vs. 30.98%), the fraction of responses rated ≥*acceptable* are higher (61.18% vs. 57.64%). It is important to note that the fraction of sentences rated *good* and ≥*acceptable* on rhyming are much larger for ISTG+RS model. Although the fluency of the responses generated by PBSMT+RS drastically improves compared to PBSMT it still lags behind the TG+RS and ISTG+RS models on both fluency and rhyming. The results in Table 1 confirm our hypothesis that off-the-shelf SMT systems are not guaranteed to be effective on our novel task.

5.5 Challenge-response examples

Table 3 shows some of the challenges and the corresponding responses of PBSMT+RS, TG+RS and TG model. While PBSMT+RS and TG+RS models generate responses reflecting a high degree of fluency, the output of the TG contains a lot of articles. It is interesting to note that TG+RS produces responses comparable to PBSMT+RS despite being a token based transduction grammar. However, PBSMT tends to produce responses that are too similar to the challenge. Moreover, TG models produce responses that indeed rhyme better (shown in boldface). In fact, TG tries to rhyme words not only at the end but also in middle of the lines, as our transduction grammar model captures structural associations more effectively than the phrase-based model.

6 Disfluency handling via disfluency correction and filtering

In this section, we compare the effect of two disfluency mitigating strategies on the quality of the responses generated by the PBSMT baseline and token based transduction grammar model with and without using rhyme scheme detection.

6.1 Correction vs. filtering

Error analysis of our initial runs showed a disturbingly high proportion of responses generated by our system that contained disfluencies with successive repetitions of words such as the and I. Upon inspection of data we noticed that the training lyrics actually did contain such disfluencies and backing vocal lines, amounting to 10% of our training data. We therefore compared two alternative strategies to tackle this problem. The first strategy involved filtering out all lines from our training corpus which contained such disfluencies. In the second strategy, we implemented a disfluency detection and correction algorithm (for example, the the the, which frequently occurred in the training corpus, was corrected to simply the). The PBSMT baseline and the TG model were trained on both the filtered and corrected versions of the training corpus and the quality of the responses were compared.

6.2 Results: Disfluency correction helps

The results in Table 4 indicate that the disfluency correction strategy outperforms the filtering strategy for both TG and TG+RS models. For the model TG+RS, disfluency correction generated 34.12% *good* responses in terms of fluency, while the filtering strategy produced only 28.63% *good* responses. Similarly for the model TG, disfluency correction produced 21.8% of responses with *good* fluency and the filtering strategy produced only 17.25%. Disfluency correction strategy produces higher fraction of responses with ≥*acceptable* fluency compared to the filtering strategy for both TG and TG+RS models. This result is not surprising, as harshly pruning

Table 4: Effect of the disfluency correction strategies on fluency of the responses generated for the TG induction models vs PBSMT baselines using both rhyme scheme detection and adjacent lines as the corpus selection method.

model+disfluency strat.	fluency (good)	fluency (≥acceptable)	rhyming (good)	rhyming (≥acceptable)
PBSMT+filtering	**4.3%**	**13.72%**	3.53%	7.06%
PBSMT+correction	3.14%	4.70%	1.57%	4.31%
PBSMT+RS+filtering	**31.76%**	**43.91%**	12.15%	21.17%
PBSMT+RS+correction	30.59%	43.53%	1.96%	9.02%
TG+filtering	17.25%	46.27%	18.04%	33.33%
TG+correction	**21.18%**	**54.51%**	23.53%	39.21%
TG+RS+filtering	28.63%	56.86%	14.90%	34.51%
TG+RS+correction	**34.12%**	**60.39%**	20.00%	42.74%

the training corpus causes useful word association information necessary for rhyming to be lost. Surprisingly, for both PBSMT and PBSMT+RS models, the disfluency correction has a negative effect on the fluency level of the response but still falls behind TG and TG+RS models. As disfluency correction yields more fluent responses for TG and TG+RS models, the results for ISTG and ISTG+RS models in Table 1 were obtained using disfluency correction strategy.

7 Maghrebi French hip hop

We have begun to apply Freestyle to rap in languages other than English, taking advantage of the language independence and linguistics-light approach of our unsupervised transduction grammar induction methods. With no special adaption our transduction grammar based model performs surprisingly well, even with significantly smaller training data size and noisier data. These results across different languages are encouraging as they can be used to discover truly language independence assumptions. We briefly describe our initial experiments on Maghrebi French hip hop lyrics below.

7.1 Dataset

We collected freely available French hip hop lyrics of approximately 1300 songs. About 85% of the songs were by Maghrebi French artists of Algerian, Moroccan, or Tunisian cultural backgrounds, while the remaining were by artists from the rest of Francophonie. As the large majority of songs are in Maghrebi French, the lyrics are sometimes interspersed with romanized Arabic such as "De la traversée du désert au bon couscous de **Yéma**" (**Yéma** means *My mother*). Some songs also contain Berber phrases, for instance "**a yemmi ino, a thizizwith**" (which means *my son, a bee*). Furthermore, some songs also contained English phrases in the style of gangster rap such as "T'es **game over**, **game over**... Le son de **Chicken wings**". As mentioned earlier, it is complexity like this which dissuaded us from making language specific assumptions in our model.

We extracted the end-of-line words and obtained a corpus containing 120,000 tokens corresponding to potential rhyming candidates with around 29,000 unique token types which was used as the training data for the rhyme scheme detector module. For the transduction grammar induction, the training data contained about 47,000 sentence pairs selected using rhyme scheme detection.

7.2 Results

After human evaluation by native French speakers and frequent hip hop listeners, our transduction grammar based model generates about 9.2% and 14.5% of the responses that are rated *good* by the human evaluators on the criterion of fluency and rhyming respectively. About 30.2% and 38% of the responses are rated as ≥*acceptable*. These numbers are encouraging given the noisy lyrics and much smaller amount of training data. Some examples of the challenge-response pairs and learned transduction rules in French are shown in Tables 5 and 6.

From Table 5, we can see that responses generated by the system rhyme with the challenges. The first response is fluent and not only perfectly rhymes with the challenge but also semantically valid. In the second example, the model realizes a less common **AABA** rhyme scheme through the response. The re-

Table 5: French hip hop challenge-response examples.

challenge	Si je me **trompe**
response	faut que je **raconte**
challenge	Un jour je suis un **livre**
response	et ce que je de **vivre**
challenge	Pacha mama ils ne voient pas ta **souffrance**
response	Combat ni leur de voulait de la **décadence**
challenge	la palestine n'etait pas une **terre** sans peuple.
response	le **darfour d'autre** de la **guerre** on est
challenge	Une **banlieue** qui **meut**
response	les **yeux** et

Table 6: Transduction rules for Maghrebi French hip hop.

transduction grammar rule	*log prob.*
$A \rightarrow$ terre/la guerre	-9.4837
$A \rightarrow$ haine/peine	-9.77056
$A \rightarrow$ mal/pays natal	-10.6877
$A \rightarrow$ je frissonne/mi corazon	-11.0931
$A \rightarrow$ gratteurs/rappeurs	-11.7306

sponse in the third example, exhibits strong rhyming with the challenge and both the challenge and the response contain words like souffrance, combat and décadence which are related. Similarly in the fourth example, the challenge and response also contain semantically related tokens which also rhyme. These examples illustrate that our transduction grammar formalism coupled with our rhyme scheme detection module does capture the necessary correspondences between lines of hip hop lyrics without assuming any language specific resources.

8 Conclusion

We presented a new machine learning approach for improvising hip hop responses to challenge lyrics by inducing stochastic transduction grammars, and demonstrated that inducing the transduction rules by interpolating bottom-up token based rule induction and rule segmentation strategies outperforms a token based baseline. We compared the performance of our Freestyle model against a widely used off-the-shelf phrase-based SMT model, showing that PB-SMT falls short in tackling the noisy and highly unstructured domain of hip hop lyrics. We showed that the quality of responses improves when the training data for the transduction grammar induction is selected using a rhyme scheme detector. Several domain related oddities such as disfluencies and backing vocals have been identified and some strategies for alleviating their effects have been compared. We also reported results on Maghrebi French hip hop lyrics which indicate that our model works surprisingly well with no special adaptation for languages other than English. In the future, we plan to investigate alternative training data selection techniques, disfluency handling strategies, search heuristics, and novel transduction grammar induction models.

Acknowledgements

This material is based upon work supported in part by the Hong Kong Research Grants Council (RGC) research grants GRF620811, GRF621008, GRF612806; by the Defense Advanced Research Projects Agency (DARPA) under BOLT contract no. HR0011-12-C-0016, and GALE contract nos. HR0011-06-C-0022 and HR0011-06-C-0023; and by the European Union under the FP7 grant agreement no. 287658. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the RGC, EU, or DARPA.

References

Ananth Ramakrishnan A., Sankar Kuppan, and Lalitha Devi Sobha. "Automatic generation of Tamil lyrics for melodies." *Workshop on Computational Approaches to Linguistic Creativity (CALC-09)*. 2009.

Karteek Addanki and Dekai Wu. "Unsupervised rhyme scheme identification in hip hop lyrics using hidden Markov models." *1st International Conference on Statistical Language and Speech Processing (SLSP 2013)*. 2013.

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. "Markov constraints for generating lyrics with style." *20th European Conference on Artificial Intelligence, (ECAI 2012)*. 2012.

David Chiang. "Hierarchical phrase-based translation." *Computational Linguistics*, 33(2), 2007.

John Cocke. *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences, New York University, 1969.

P.A. Devijer. "Baum's forward-backward algorithm revisited." *Pattern Recognition Letters*, 3(6), 1985.

D. Genzel, J. Uszkoreit, and F. Och. "Poetic statistical machine translation: rhyme and meter." *2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*. Association for Computational Linguistics, 2010.

E. Greene, T. Bodrumlu, and K. Knight. "Automatic analysis of rhythmic poetry with applications

to generation and translation." *2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*. Association for Computational Linguistics, 2010.

Long Jiang and Ming Zhou. "Generating Chinese couplets using a statistical MT approach." *22nd International Conference on Computational Linguistics (COLING 2008)*. 2008.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. "Moses: Open source toolkit for statistical machine translation." *Interactive Poster and Demonstration Sessions of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*. June 2007.

Mark Liberman. "Rap scholarship, rap meter, and the anthology of mondegreens." `http://languagelog.ldc.upenn.edu/nll/?p=2824`, December 2010. Accessed: 2013-06-30.

Franz Josef Och. "Minimum error rate training in statistical machine translation." *41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*. July 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation." *40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*. July 2002.

S. Reddy and K. Knight. "Unsupervised discovery of rhyme schemes." *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT 2011)*, vol. 2. Association for Computational Linguistics, 2011.

Jorma Rissanen. "A universal prior for integers and estimation by minimum description length." *The Annals of Statistics*, 11(2), June 1983.

Markus Saers, Karteek Addanki, and Dekai Wu. "From finite-state to inversion transductions: Toward unsupervised bilingual grammar induction." *24th International Conference on Computational Linguistics (COLING 2012)*. December 2012.

Markus Saers, Karteek Addanki, and Dekai Wu. "Combining top-down and bottom-up search for unsupervised induction of transduction grammars." *Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-7)*. June 2013.

Markus Saers and Dekai Wu. "Reestimation of reified rules in semiring parsing and biparsing." *Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-5)*. Association for Computational Linguistics, June 2011.

Ray J. Solomonoff. "A new method for discovering the grammars of phrase structure languages." *International Federation for Information Processing Congress (IFIP)*. 1959.

M. Sonderegger. "Applications of graph theory to an English rhyming corpus." *Computer Speech & Language*, 25(3), 2011.

Andreas Stolcke. "SRILM – an extensible language modeling toolkit." *7th International Conference on Spoken Language Processing (ICSLP2002 - INTERSPEECH 2002)*. September 2002.

Dekai Wu. "A polynomial-time algorithm for statistical machine translation." *34th Annual Meeting of the Association for Computational Linguistics (ACL96)*. 1996.

Dekai Wu. "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora." *Computational Linguistics*, 23(3), 1997.

Dekai Wu. "Textual entailment recognition using inversion transduction grammars." Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc (eds.), *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop (MLCW 2005)*, vol. 3944 of *Lecture Notes in Computer Science*. Springer, 2006.

Dekai Wu, Karteek Addanki, and Markus Saers. "FREESTYLE: A challenge-response system for hip hop lyrics via unsupervised induction of stochastic transduction grammars." *14th Annual Conference of the International Speech Communication Association (Interspeech 2013)*. 2013a.

Dekai Wu, Karteek Addanki, and Markus Saers. "Modeling hip hop challenge-response lyrics as machine translation." *14th Machine Translation Summit (MT Summit XIV)*. 2013b.

Dekai Wu and Pascale Fung. "Inversion transduction grammar constraints for mining parallel sentences from quasi-comparable corpora." *Second International Joint Conference on Natural Language Processing (IJCNLP 2005)*. Springer, 2005.

Richard Zens and Hermann Ney. "A comparative study on reordering constraints in statistical machine translation." *41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*. Association for Computational Linguistics, 2003.

Modeling Scientific Impact with Topical Influence Regression

James Foulds **Padhraic Smyth**
Department of Computer Science
University of California, Irvine
{jfoulds, smyth}@ics.uci.edu

Abstract

When reviewing scientific literature, it would be useful to have automatic tools that identify the most influential scientific articles as well as how ideas propagate between articles. In this context, this paper introduces *topical influence*, a quantitative measure of the extent to which an article tends to spread its topics to the articles that cite it. Given the text of the articles and their citation graph, we show how to learn a probabilistic model to recover both the degree of topical influence of each article and the influence relationships between articles. Experimental results on corpora from two well-known computer science conferences are used to illustrate and validate the proposed approach.

1 Introduction

Scientific articles are not created equal. Some articles generate entire disciplines or sub-disciplines of research, or revolutionize how we think about a problem, while others contribute relatively little. When we are first introduced to a new area of scientific study, it would be useful to automatically find the most important articles, and the relationships of influence between articles. Understanding the impact of scientific work is also crucial for hiring decisions, allocation of funding, university rankings and other tasks that involve the assessment of scientific merit. If scientific works stand on the shoulders of giants, we would like to be able to find the giants.

The importance of a scientific work has previously been measured chiefly through metrics derived from citation counts, such as impact factors. However, citation counts are not the whole story. Many citations are made in passing, are relevant to only one section of an article, or make no impact on a work but are referenced out of "politeness, policy or piety" (Ziman, 1968). In reality, scientific impact has many dimensions. Some articles are important because they describe scientific discoveries that alter our understanding of the world, while some develop essential tools and techniques which facilitate future research. Other articles are influential because they introduce the seeds of new ideas, which in turn inspire many other articles.

In this work we introduce *topical influence*, a quantitative metric for measuring the latter type of scientific influence, defined in the context of an unsupervised generative model for scientific corpora. The model posits that articles "coerce" the articles that cite them into having similar topical content to them. Thus, articles with higher topical influence have a larger effect on the topics of the articles that cite them. We model this influence mechanism via a regression on the parameters of the Dirichlet prior over topics in an LDA-style topic model. We show how the models can be used to recover meaningful influence scores, both for articles and for specific citations. By looking not just at the citation graph but also taking into account the content of the articles, topical influence can provide a better picture of scientific impact than simple citation counts.

2 Background

Bibliometrics, the quantitative study of scientific literature, has a long history. One example of a widely-used bibliometric measure of interest is the *impact factor* of a publication venue for a given year, defined to be the average number of times articles from

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 113–123,
Seattle, Washington, USA, 18-21 October 2013.

that venue, published in the previous two years, were cited in that year. However, the quality of articles in a given publication venue can vary wildly, and it is difficult to compare impact factors between different disciplines of study. The number of citations an article receives is an indication of importance, but this is confounded by the unknown function of each citation. Measures of importance such as PageRank (Brin and Page, 1998) can be derived recursively from the citation graph. Such graph-based measures do not in general make use of the textual content of the articles, although it is possible to apply them to graphs where the edges between articles are determined based on the similarity of their content instead of the citation graph (Lin, 2008).

A variety of methods have previously been proposed for analyzing text and citation links together, such as modeling connections between words and citations Cohn and Hofmann (2001), classifying citation function (Teufel et al., 2006), and jointly modeling citation links and document content (Chang and Blei, 2009). However, these methods do not directly measure article importance or influence relationships between articles given their citations.

More closely related to the present work, Dietz et al. (2007) proposed the citation influence model (CIM). Building on the latent Dirichlet allocation (LDA) framework, CIM assumes that each word is drawn by first selecting either (a) the distribution over topics of a cited article (with probability proportional to the influence weight of that article on the present article) or (b) a novel topic distribution, and drawing a topic from the selected distribution, then finally drawing the word from the chosen topic.[1] In their approach, every word is assigned an extra latent variable, namely the cited article whose topic distribution the topic was drawn from. For the model proposed in this paper, we do not need to introduce these additional latent variables, which leads to a simpler latent representation and fewer variables to sample during inference. Dietz et al. (2007) also assume that the citation graph is bipartite, consisting of one set of citing articles and one set of cited articles—in contrast, our proposed models can handle arbitrary citation graphs in the form of directed acyclic graphs (DAGs). While both the CIM and our approach can identify the influence of specific citations between articles, our model can also infer how influential each article is overall, and provides a flexible modeling framework which can handle different assumptions about influence.

Another related method is due to Shaparenko and Joachims (2009), who propose a mixture modeling approach for the detection of novel text content. Nallapati et al. (2011) introduced TopicFlow, a PLSA-based model for the flow of topics in a document network. In their model, citing articles "vote" on each cited article's topic distribution in retrospect, via a network flow model. Since this voting occurs in time-reversed order, it does not describe an influence mechanism and is not a generative model that can simulate or predict new documents.

Finally, the document influence model of Gerrish and Blei (2010) can be viewed as orthogonal to this work, in that it models the impact of documents on *topics* over time (specifically, how topics change over time) rather than how articles influence the specific *articles* that cite them.

3 Topical Influence Regression

Scientific research is seldom performed in a vacuum. New research builds on the research that came before it. Although there are many aspects by which the importance of a scientific article can be judged, in this work we are interested in the extent to which a given article has or will have subsequent articles that build upon it or are otherwise inspired by its ideas. We begin by defining *topical influence*, a quantitative measure for this type of influence.

3.1 Topical Influence

It is not immediately obvious how one might quantify such a notion of "idea-based" influence. However, the mechanism used in the scientific community for giving credit to prior work is citation. The presence of a citation from article b to article a therefore indicates that article b may have been influenced by the ideas in article a, to some unknown extent. We hypothesize that the extent of this influence manifests itself in the language of b. Using latent Dirichlet allocation (LDA) topics as a concrete proxy for

[1] A somewhat similar model was also proposed by He et al. (2009)

the vague notion of "ideas", we define the *topical influence* of a to be the extent to which article a coerces the documents which cite it to have similar topic distributions to it. Topical influence will be made precise in the context of a generative model for scientific corpora, conditioned on the citation graph, called *topical influence regression* (TIR).

The proposed model extends the LDA framework of Blei et al. (2003). In LDA, each word $w_i^{(d)}$ of each document d is assigned to one of K latent topics, $z_i^{(d)}$. Each topic $\Phi^{(k)}$ is a discrete distribution over words. Document d has a distribution over topics $\theta^{(d)}$, which can be viewed as a "location in topic space" summarizing its thematic content. The $\theta^{(d)}$'s have a Dirichlet prior distribution with parameters $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_K]^\intercal$. Although the α_k's are often set to be equal, representing a relatively uninformative prior over the θ's, a unique $\alpha^{(d)}$ for each document can also be used to encode prior information such as the effect of other variables on the topics of that document (Mimno and McCallum, 2008). In our case, we want to model the influence that a document has on the topic distributions of the documents that cite it. A natural way to encode such influence, then, is to allow documents to affect the value of $\alpha^{(d)}$ for each document d that cites them.

Accordingly, we model each article d as having a latent, non-negative "topical influence" value $l^{(d)}$. Let $n^{(d)}$ be number of words in article d, $n_k^{(d)}$ be the number of words assigned to topic k, and let $C^{(d)}$ be the set of articles that d cites. We model $\alpha^{(d)}$ as

$$\alpha^{(d)} = \sum_{c \in C^{(d)}} l^{(c)} \bar{z}^{(c)} + \alpha \,, \qquad (1)$$

where $\bar{z}^{(c)} = \frac{1}{n^{(c)}}[n_1^{(c)}, \ldots, n_K^{(c)}]^\intercal$ is the normalized histogram of topic counts for document c, and α is a constant for smoothing. Since the $\bar{z}^{(c)}$'s sum to one, the topical influence $l^{(c)}$ of article c can be interpreted as the number of words of precision that it adds to the prior of the topic distributions of each document that cites it. As we increase $l^{(c)}$, the articles that cite c become more likely to have similar topic proportions to it. Thus, $l^{(c)}$ encodes the degree to which article c influences the topics of each of the articles that cite it.

From another perspective, marginalizing out $\theta^{(d)}$, we can view the topic counts (in the standard LDA model) for document d as being drawn from a Polya urn scheme with $\alpha_k^{(d)}$ (possibly fractional) balls of each color $k \in \{1, \ldots, K\}$ initially in the urn. For each word, a ball is drawn randomly from the urn and the topic assignment is determined according to its color k. The ball is replaced in the urn, along with a new ball of color k. In our model, for each article c cited by article d we place $l^{(c)}$ balls, with colors distributed according to $\bar{z}^{(c)}$, into article d's urn initially. Thus, article d's topic assignments are more likely to be similar to those of the more influential articles that it cites. The total number of balls that d added to other articles' urns,

$$T^{(d)} \triangleq \sum_{b: d \in C^{(b)}} l^{(d)} = l^{(d)} \left| \{b : d \in C^{(b)}\} \right| \qquad (2)$$

measures the total impact (in a topical sense) of the article. We refer to this as *total topical influence*.

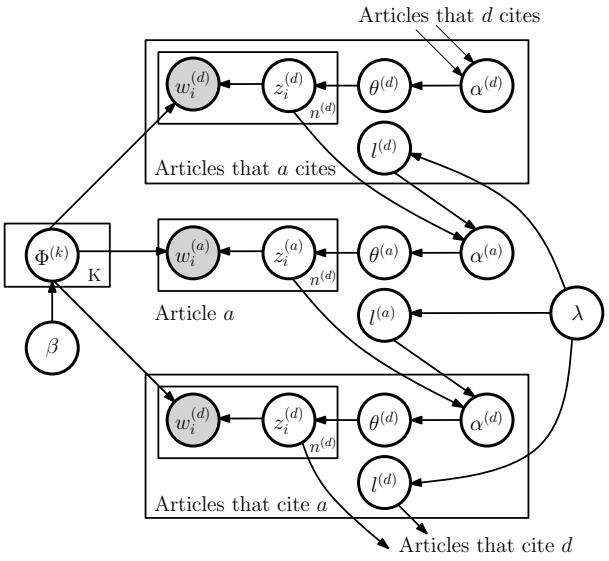


Figure 1: The graphical model for the portion of the TIR model connected to article a (the links from the z's and l's to the $\alpha^{(d)}$'s are deterministic).

3.2 Generative Model for Topical Influence Regression

The full assumed generative process for articles in this model begins with a directed acyclic citation graph $G = \{V, E\}$. Intuitively, citation graphs are typically DAGs because articles can normally only cite articles that precede them in time. We assume that G is a DAG so that influence relationships are

consistent with some temporal ordering of the articles, and so that the resulting model is a Bayesian network. Here, each vertex v_i corresponds to an article d_i, edge $e = (v_1, v_2) \in E$ IFF d_1 is cited by d_2, and vertices (articles) are numbered in a topological ordering with respect to G. Such an ordering exists because G is a DAG. We model each article d's word vector $w^{(d)}$ as being generated in topological sequence, similarly to LDA but with its prior over topic distribution being Dirichlet($\alpha^{(d)}$), as given by Equation 1. Note that each $\alpha^{(d)}$ is a function of the topics of the documents that it cites, parameterized by their topical influence values. We therefore call this model *topical influence regression* (TIR).

The TIR model provides us with topical influence scores for each article, but it does not tell us about topical influence relationships between specific pairs of cited and citing articles. To model such relationships, we can consider a hierarchical extension to TIR, with edge-wise topical influences $l^{(c,d)}$ for each edge (c, d) of the citation graph, $l^{(c,d)} \sim$ TruncGaussian($l^{(c)}, \sigma, l^{(c,d)} \geq 0$). In this case,

$$\alpha^{(d)} = \sum_{c \in C^{(d)}} l^{(c,d)} \bar{z}^{(c)} + \alpha \,. \tag{3}$$

This hierarchical setup allows us to continue to infer article-level topical influences, and provides a mechanism for sharing statistical strength between influences associated with one cited article. We shall refer to the model with influences on just the nodes (articles) as TIR, and the hierarchical extension with influences on the edges as TIRE. The graphical model for TIR is given in Figure 1, and the generative process is detailed in the following pseudocode:

- For each topic k
  - Sample the topic $\Phi^{(k)} \sim$ Dirichlet(β)
- For each document d, in topological order
  - Sample an influence weight, $l^{(d)} \sim$ Exponential(λ)
  - If using the TIRE model
    - For each cited document $c \in C^{(d)}$
      - Draw edge influence weight, $l^{(c,d)} \sim$ TruncGauss($l^{(c)}, \sigma, l^{(c,d)} \geq 0$)
  - Assign a prior over topics via $\alpha^{(d)} = \sum_{c \in C^{(d)}} l^{(c)} \bar{z}^{(c)} + \alpha$ (TIR), or $\alpha^{(d)} = \sum_{c \in C^{(d)}} l^{(c,d)} \bar{z}^{(c)} + \alpha$ (TIRE)
  - Sample a distribution over topics, $\theta^{(d)} \sim$ Dirichlet($\alpha^{(d)}$)
  - For each word i in document d
    - Sample a topic $z_i^{(d)} \sim$ Discrete($\theta^{(d)}$)
    - Sample a word $w_i^{(d)} \sim$ Discrete($\Phi^{(z_i^{(d)})}$)

3.3 Relationship to Dirichlet-Multinomial Regression

The TIR model can be viewed as an adaption of the Dirichlet-multinomial regression (DMR) framework of Mimno and McCallum (2008) to model topical influence. DMR also endows each document with its own unique $\alpha^{(d)}$, but with $\alpha_k^{(d)} = \exp(x^{(d)\intercal} \lambda_k)$ being a function of the observed feature vector $x^{(d)}$ parameterized by regression coefficients λ. The DMR model can also be applied to text corpora with citation information, by setting the feature vectors to be binary indicators of the presence of a citation to each article. TIR differs in that the functional form of the regression is parameterized in a way that directly models influence, and also differs in that the regression takes advantage of the content of the cited articles via their topic assignments.

Because an article's prior over topic distributions depends on the topic assignments of the articles that it cites, TIR induces a network of dependencies between the topic assignments of the documents. Specifically, if we collapse out Θ, the dependencies between the z's of each document form a Bayesian network whose graph is the citation graph. In contrast, DMR treats the documents as conditionally independent given their citations, and does not exploit their content in the regression.

To illustrate this, Figure 2 shows an example citation graph and the resulting Bayesian network. In the figure, an edge in (a) from c to d corresponds to a citation of c by d. Conditioned on the topics, the dependence relationships between z nodes in (b) follow the same structure as the citation graph.

4 Inference

We perform inference using a Markov chain Monte Carlo technique. We use a collapsed Gibbs sampling approach analogous to Griffiths and Steyvers (2004), integrating out Θ and

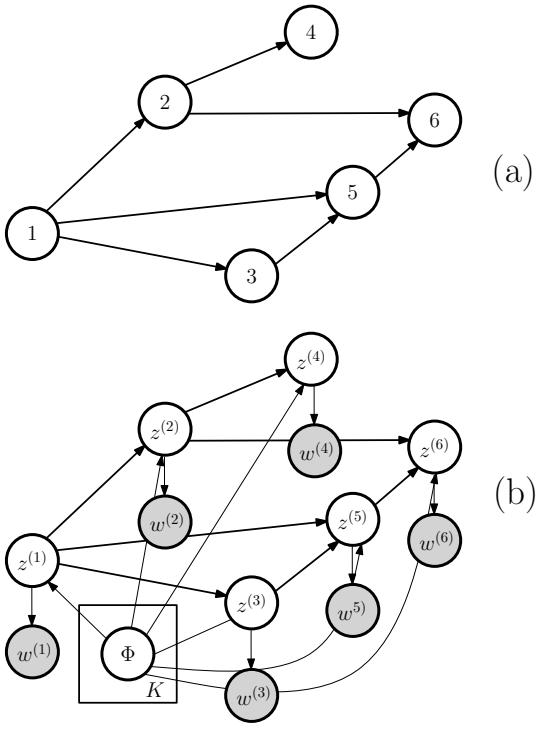


Figure 2: (a) An example citation network. (b) Graphical model for TIR on the example network, collapsing out Θ but retaining topics Φ. Influence variables and hyperparameters not shown for simplicity.

Φ. The update equation for the topic assignments is

$$Pr(z_i^{(d)} = k | z^{-(d,i)}, \ldots) \propto (n_k^{(d)-(d,i)} + \alpha_k^{(d)}) \frac{n_k^{(w_i^{(d)})-(d,i)} + \beta_{w_i^{(d)}}}{n_k^{-(d,i)} + \sum_w \beta_w} \times \prod_{d':d \in C(d')} \text{Polya}(z^{(d')} | \alpha^{(d')} : z_i^{(d)} = k, z^{-(d,i)}, l) \quad (4)$$

where the n_k's are the counts of the occurrences of topic k over all of the entries determined by the superscript. The $-(d,i)$ superscript indicates excluding the current assignment for $z_i^{(d)}$. The update equation is similar to the update equations of Griffiths and Steyvers, but with a different α for each document d, and with multiplicative weights for each document that cites it. These weights $\text{Polya}(z^{(d)}|\alpha^{(d)})$ are the likelihood for a multivariate Polya (a.k.a. Dirichlet-multinomial) distribution,

$$\text{Polya}(z^{(d)}|\alpha^{(d)}) = \frac{\Gamma(\sum_k \alpha_k^{(d)})}{\Gamma(n^{(d)} + \sum_k \alpha_k^{(d)})} \prod_k \frac{\Gamma(n_k^{(d)} + \alpha_k^{(d)})}{\Gamma(\alpha_k^{(d)})}.$$

In the case of TIR, in the collapsed model the full conditional posterior for the topical influence values l is $Pr(l|z,\lambda) \propto Pr(z|l)Pr(l|\lambda)$. Here, $Pr(z|l) = \prod_{d=1}^{D} \text{Polya}(z^{(d)}|l^{C^{(d)}}, z^{C^{(d)}})$. The topical influence values l can be sampled using Metropolis-Hastings updates, or slice sampling. An alternative is to perform stochastic EM, optimizing the likelihood or the posterior probability of l, interleaved within the Gibbs sampler, as in Mimno and McCallum (2008) and Wallach (2006). In experiments on synthetic data we found that maximum likelihood updates on l, obtained via gradient ascent, resulted in the lowest L1 error from the true l, so we use this strategy for the experimental results in this paper. The derivative of the log-likelihood with respect to the topical influence $l^{(a)}$ of article a is

$$\frac{\mathrm{d}Pr(z|l)}{\mathrm{d}l^{(a)}} = \sum_{d:a \in C(d)} \Big(\Psi(\sum_k \sum_{c \in C^{(d)}} l^{(c)} \bar{z}_k^{(c)} + K\alpha) - \Psi(\sum_k \sum_{c \in C^{(d)}} l^{(c)} \bar{z}_k^{(c)} + K\alpha + n^{(d)}) \Big) + \sum_{d:a \in C(d)} \sum_{k=1}^{K} \bar{z}_k^{(a)} \Big(\Psi(\sum_{c \in C^{(d)}} l^{(c)} \bar{z}_k^{(c)} + \alpha + n_k^{(d)}) - \Psi(\sum_{c \in C^{(d)}} l^{(c)} \bar{z}_k^{(c)} + \alpha) \Big),$$

where $\Psi(.)$ is the digamma function. For TIRE, the likelihood decomposes across documents and we can optimize the incoming edge weights for each document separately. We have

$$\frac{\mathrm{d}Pr(z^{(d)}|l)}{\mathrm{d}l^{(a,d)}} = \Psi(\sum_k \sum_{c \in C^{(d)}} l^{(c,d)} \bar{z}_k^{(c)} + K\alpha) - \Psi(\sum_k \sum_{c \in C^{(d)}} l^{(c,d)} \bar{z}_k^{(c)} + K\alpha + n^{(d)}) + \sum_{k=1}^{K} \bar{z}_k^{(a)} \Big(\Psi(\sum_{c \in C^{(d)}} l^{(c,d)} \bar{z}_k^{(c)} + \alpha + n_k^{(d)}) - \Psi(\sum_{c \in C^{(d)}} l^{(c,d)} \bar{z}_k^{(c)} + \alpha) \Big).$$

We optimize the node-level l's in TIRE via the least squares estimate (LSE), $\hat{l}^{(a)} = \frac{1}{|\{d:a \in C^{(d)}\}|} \sum_{d:a \in C^{(d)}} l^{(a,d)}$. Although the LSE for the mean of a truncated Gaussian is biased, it is widely used as it is more robust than the MLE (A'Hearn, 2004).

5 Experimental Analysis

In this section we experimentally investigate the properties of TIR and TIRE. We consider two scientific corpora: a collection of 3286 of articles from the Association for Computational Linguistics (ACL) conference[2] (Radev et al., 2009) published between 1987 and 2011, and a corpus of articles from the Neural Information Processing Systems (NIPS) conference[3] containing 1740 articles from 1987 to 1999. The corpora both contained a small number (53, and 14, respectively) of citation graph loops due to insider knowledge of simultaneous publications. Some loops were removed by manual deletion of "insider knowledge" edges, and others were removed by deleting edges in the loop uniformly at random. For computational efficiency, we performed approximate Gibbs updates where we drop the multiplicative Polya likelihood terms in Equation 4. This corresponds to only transmitting influence information downward in the citation DAG, but not transmitting "reverse influence" information upwards. Preliminary experiments on synthetic data indicated that this did not significantly impact the ability of the model to recover the topical influence weights. As one might expect, LDA is already capable of inferring topic distributions which are good enough to perform the regression on, without fully exploiting the additional feedback from the regression. This algorithm has a similar running time to the standard collapsed Gibbs sampler for LDA, as the regression step is not a bottleneck.

In all experiments, we set the hyper-parameters to $\alpha = 0.1, \beta = 0.1$ and the σ parameter for the truncated Gaussian in TIRE to be 1. We interleaved regression steps every 10 Gibbs iterations. For exploratory data analysis experiments the models were trained for 500 burn-in iterations, and the samples from the final iterations were used for the analysis.

[2] http://clair.eecs.umich.edu/aan/

[3] http://www.arbylon.net/resources.html, published by Gregor Heinrich and based on an earlier collection due to Sam Roweis.

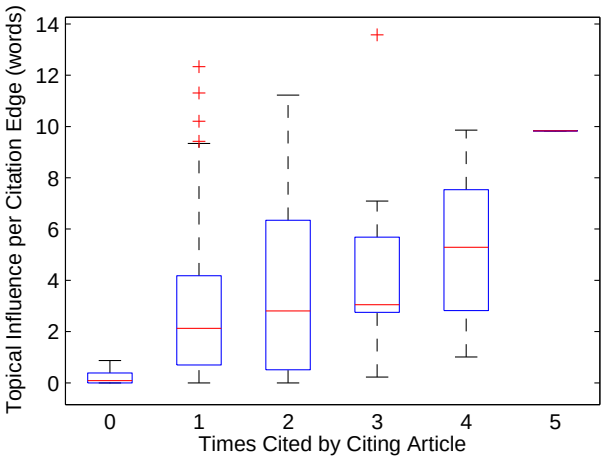


Figure 3: Topical influence per edge versus number of times cited by the citing article (NIPS). Several articles had zero in-text citations due to author or dataset errors.

5.1 Model Validation using Metadata

It is not immediately obvious how to best validate an unsupervised model of citation influence. Ground truth is not well-defined and human evaluation requires extensive knowledge of the individual papers in the corpora. With this in mind, we explore how topical influence scores relate to document metadata, which serves as a proxy for ground truth.

In many cases, if article c is repeatedly cited in the text of article d it may indicate that d builds heavily on c. We would therefore expect to see an association between repeated citations and edge-wise topical influence $l^{(c,d)}$. For each of the 106 papers in the NIPS corpus with at least three distinct references, we counted the number of repeated citations for the most influential and least influential references according to the TIRE model (Figure 3). Overall, the "most influential" references were cited 171 times in the text of their citing articles, while the "least influential" references were cited 128 times. Of the 45 articles where the counts were not tied, the most influential references had the higher citation counts 33 times. A sign test rejects the null hypothesis that the median difference in citation counts between least and most influential references is zero at $\alpha = 0.05$, with p-value $\approx 5 \times 10^{-4}$.

Self-citations, where at least one author is in common between cited and citing articles, are also informative (Figure 4). Authors often build upon their own work, so we would expect self-citations to have higher edge-wise topical influence on average. For ACL the mean topical influence for a self citation edge is 2.80 and for a non-self citation is 1.40. For NIPS the means are 5.05 (self) and 3.15 (non-self). A two-sample t-test finds these differences are both significant at $\alpha = 0.05$.

5.2 Prediction Experiments

We also used a document prediction task to explore whether the posited latent structure is predictively useful. We selected roughly 10% of the articles in each corpus (170 and 330 documents for NIPS and ACL, respectively) for testing, chosen among the articles that made at least one citation. We held out a randomly selected set of 50% of their words and evaluated the log probability of the held out partial documents under each model. This is equivalent to evaluating on a set of new documents with the same set of references as the held out set. Evaluation was performed using annealed importance sampling (Neal, 2001), as in Wallach et al. (2009) except we used multiple samples per likelihood computation.

The TIR models were compared to LDA and an "additive" version of DMR with link function $\alpha_k^{(d)} = x^{(d)\intercal}\lambda_k + \alpha$, where the λs were constrained to be positive and given an exponential prior with mean one. For DMR, binary feature vectors encoded the presence or absence of each possible citation. For each algorithm, we burned in for 250 iterations, then executed 1000 iterations, optimizing topical influence weights/DMR parameters every 10th iteration. Held-out log probability scores were computed by performing AIS with every 100th sample, and averaging the results to estimate the posterior predictive probability $Pr(\text{held out article}|\text{training set, citations, model})$.

It was found that all of the regression methods had superior predictive performance to LDA on these corpora, demonstrating that topical influence has predictive value (Table 1). Although DMR performed slightly better than TIR predictively, TIR was competitive despite the fact that it has a factor of K less regression parameters. Note that DMR does not provide an interpretable notion of influence.

5.3 Exploring Topical Influence

In this section we explore the inferred topical influence scores $l^{(d)}$, total topical influence scores $T^{(d)}$ and edgewise topical influence scores $l^{(c,d)}$ (recall their definitions in Equations 1, 2 and 3, respectively). Table 2 shows the most influential articles in the ACL corpus, according to citation counts, topical influence and total topical influence (the latter two inferred with the TIR model). The most frequently cited paper within the ACL corpus, written by Papineni et al., introduces BLEU, a technique for evaluating machine translation (MT) systems.[4] This paper is of great importance to the computational linguistics community because the method that it introduces is widely used to validate MT systems. However, the BLEU article has a relatively low *topical* influence value of 0.58, consistent with the fact that most of the papers that cite it use the technique as part of their *methodology* but do not *build upon its ideas*. We emphasize that topical influence measures a specific dimension of scientific importance, namely the tendency of an article to influence the ideas (as mediated by the topics) of citing articles; papers with low topical influence such as the BLEU article may be important for other reasons.

Ranking papers by their influence weights $l^{(d)}$ (Table 2, middle) has the opposite difficulty to ranking by citation counts — the papers with the highest topical influence were typically cited only once, by the same authors. This makes sense, given what the model is designed to do. The lone citing papers were certainly topically influenced by these articles.

A more useful metric, however, is the total topical influence $T^{(d)}$ (the bottom sub-table in Table 2). This is the total number of words of prior concentration, summed over all of its citers, that the article has contributed, and is a measure of the total corpus-wide topical influence of the paper. This metric ranks the BLEU paper at 5th place, down from 1st place by citation count. The ACL paper with the highest total topical influence, by David Chiang, won the ACL best paper award in 2005.

The behavior of the different metrics is echoed in the NIPS corpus (Table 3). The most cited paper, "Handwritten Digit Recognition," by

[4] Citations within the corpora are of course only a small fraction of the total set of citations for many of these papers.

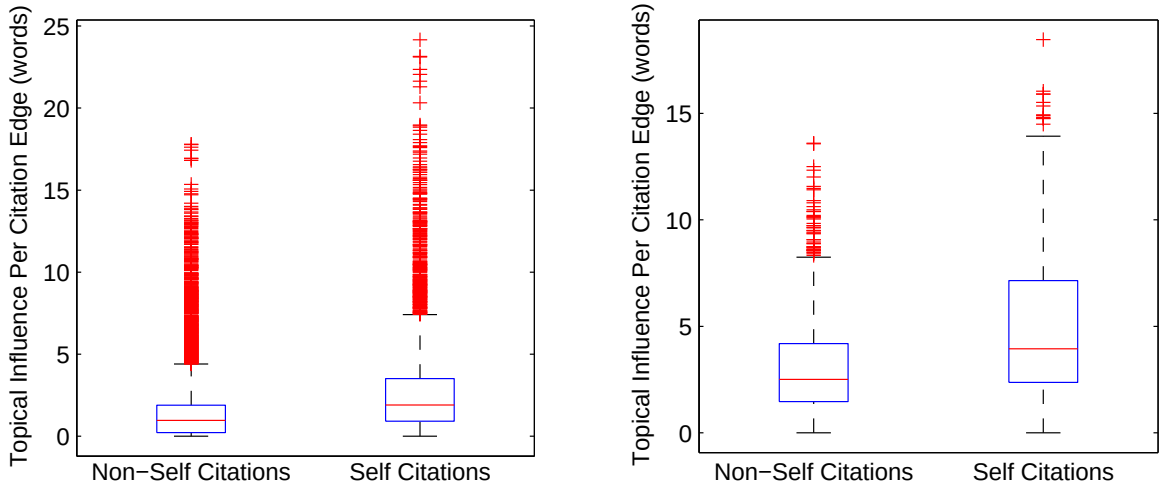


Figure 4: Topical influence for self and non-self citation edges. Left: ACL. Right: NIPS.

	ACL			NIPS		
	Wins	Losses	Average Improvement	Wins	Losses	Average Improvement
TIR	297	33	65.7	150	20	38.2
TIRE	276	54	63.0	148	22	38.7
DMR	302	28	79.1	157	13	48.4

Table 1: Wins, losses and average improvement for log probabilities of held-out articles, versus LDA. Each "Win" corresponds to the model assigning a higher log probability score for the test portion of a held-out document than LDA assigned to that document.

Le Cun et al. (1990), is an early successful application of neural networks. The paper does not introduce novel models or algorithms, but rather, in the authors' words, "show[s] that large back propagation (BP) networks can be applied to real image recognition problems." Thus, although it is has an important role as a landmark neural network success story, it does not score highly in terms of *topical* influence. This paper is ranked 13th according to total topical influence, with a score of 1.6. The top two-ranked papers according to total topical influence, on Gaussian Process Regression and POMDPs respectively, were both seminal papers that spawned large bodies of related work. An interesting case is the third-ranked paper in the NIPS corpus, by Wang et al., on the theory of early stopping. It is only referenced three times, but has a very high topical influence of 19.3 words. All three citing papers are also on the theory of early stopping, and one of the papers, by Wang and Venkatesh, directly extends a theoretical result of this paper. Although it is easy to see why this paper scores highly on topical influence, in this case the metric has perhaps overstated its importance. A limitation of topical influence is that it can potentially give more credit than is due when an article is cited by a small number of topically similar papers, due to overfitting. This is likely to be an issue for any topic-based approach for modeling scientific influence. However, topics help to absorb lexical ambiguity and author-specific idiosyncracies, mitigating the problem relative to word-based approaches.

Using the TIRE model, we can also look at influence relationships between pairs of articles. Tables 4 and 5 show the most and least topically influential references, and the most and least influenced citing papers, for three example articles from ACL and NIPS, respectively. The model correctly assigns higher influence scores along the edges to and from relevant documents. For the ACL papers, the BLEU algorithm's article is inferred to have zero topical influence on Chiang's paper, consistent with its role

Top 5 Articles by Citation Count	
140	BLEU: a Method for Automatic Evaluation of Machine Translation. K. Papineni, S. Roukos, T. Ward, W. Zhu.
105	Minimum Error Rate Training in Statistical Machine Translation. F. Och.
64	A Hierarchical Phrase-Based Model for Statistical Machine Translation. D. Chiang.
64	Accurate Unlexicalized Parsing. D. Klein, C. Manning.
59	Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. D. Yarowsky.
Top 5 articles by Topical Influence	
11.38	Refining Event Extraction through Cross-document Inference. H. Ji, R. Grishman.
11.37	Bayesian Learning of Non-compositional Phrases with Synchronous Parsing. H. Zhang, C. Quirk, R. Moore, D. Gildea.
10.48	A Plan Recognition Model for Clarification Subdialogues. D. Litman, J. Allen.
10.38	PCFGs with Syntactic and Prosodic Indicators of Speech Repairs. J. Hale et al.
10.30	Referring as Requesting, P. Cohen
Top 5 Articles by Total Topical Influence	
111.46 (1.74 $\times$ 64)	A Hierarchical Phrase-Based Model for Statistical Machine Translation. D. Chiang.
101.12 (6.74 $\times$ 15)	Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. D. Xiong, Q. Liu, S. Lin.
98.56 (5.80 $\times$ 17)	A Logical Semantics for Feature Structures. R. Kasper, W. Rounds.
85.15 (2.18 $\times$ 39)	Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. F. Och, H. Ney
81.82 (0.58 $\times$ 140)	BLEU: a Method for Automatic Evaluation of Machine Translation, K. Papineni, S. Roukos, T. Ward, and W. Zhu.

Table 2: Most influential articles in the ACL Conference corpus, according to citation counts (top), topical influence $l^{(d)}$ inferred by TIR (middle), and total topical influence $T^{(d)}$ inferred by TIR (bottom). For total topical influence, the breakdown of $T^{(d)} = l^{(d)}\times$ citation count is shown in parentheses.

Top 5 Articles by Citation Count	
26	Handwritten Digit Recognition with a Back-Propagation Network. Y. Le Cun, et al.
19	Optimal Brain Damage. Y. Le Cun, J. Denker, S. Solla.
17	A New Learning Algorithm for Blind Signal Separation. S. Amari, A. Cichocki, H. Yang.
17	Efficient Pattern Recognition Using a New Transformation Distance. P. Simard, Y. Le Cun, J. Denker.
14	The Cascade-Correlation Learning Architecture. S. Fahlman, C. Lebiere.
Top 5 articles by Topical Influence	
29.7	Synchronization and Grammatical Inference in an Oscillating Elman Net. B. Baird, T. Troyer, F. Eeckman.
26.3	Learning the Solution to the Aperture Problem for Pattern Motion with a Hebb Rule. M. Sereno.
25.9	ALVINN: An Autonomous Land Vehicle in a Neural Network. D. Pomerleau.
25.1	Some Estimates of Necessary Number of Connections and Hidden Units for Feed-Forward Networks. A. Kowalczyk.
24.7	Complex- Cell Responses Derived from Center-Surround Inputs: The Surprising Power of Intradendritic Computation. B. Mel, D. Ruderman, K. Archie.
Top 5 Articles by Total Topical Influence	
84.7 (10.6 $\times$ 8)	Gaussian Processes for Regression. C. Williams, C. Rasmussen.
63.9 (7.1 $\times$ 9)	Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. T. Jaakkola, S. Singh, M. Jordan.
57.9 (19.3 $\times$ 3)	Optimal Stopping and Effective Machine Complexity in Learning. C. Wang, S. Venkatesh, J. Judd.
54.7 (10.9 $\times$ 5)	Links Between Markov Models and Multilayer Perceptrons. H. Bourlard, C. Wellekens.
51.2 (3.7 $\times$ 14)	The Cascade-Correlation Learning Architecture. S. Fahlman, C. Lebiere.

Table 3: Most influential articles in the NIPS corpus, according to citation counts (top), topical influence $l^{(d)}$ inferred by TIR (middle), and total topical influence $T^{(d)}$ inferred by TIR (bottom).

A Hierarchical Phrase-Based Model for Statistical Machine Translation. D. Chiang.		
Most influential reference	1.48	Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. F. Och and H. Ney.
Least influential reference	0.00	BLEU: a Method for Automatic Evaluation of Machine Translation. K. Papineni, S. Roukos, T. Ward, W. Zhu.
Most influenced citer	2.54	Toward Smaller, Faster, and Better Hierarchical Phrase-based SMT. M. Yang, J. Zheng.
Least influenced citer	0.60	An Optimal-time Binarization Algorithm for Linear Context-Free Rewriting Systems with Fan-out Two. C. Gmez-Rodrguez, G. Satta.
Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. D. Yarowsky.		
Most influential reference	2.52	Subject-dependent Co-occurrence and Word Sense Disambiguation. J. Guthrie, L. Guthrie, Y. Wilks, H. Aidinejad.
Least influential reference	0.53	Word-sense Disambiguation using Statistical Methods. P. Brown, S. Della Pietra, V. Della Pietra, R. Mercer.
Most influenced citer	1.81	Discriminating Image Senses by Clustering with Multimodal Features. N. Loeff, C. Alm, D. Forsyth.
Least influenced citer	0.00	Semi-supervised Convex Training for Dependency Parsing. Q. Wang, D. Schuurmans, D. Lin.
Accurate Unlexicalized Parsing. D. Klein, C. Manning.		
Most influential reference	3.87	Parsing with Treebank Grammars: Empirical Bounds, Theoretical Models, and the Structure of the Penn Treebank. D. Klein and C. Manning.
Least influential reference	0.81	Efficient Parsing for Bilexical Context-Free Grammars and Head Automaton Grammars. J. Eisner, G. Satta.
Most influenced citer	1.67	Evaluating the Accuracy of an Unlexicalized Statistical Parser on the PARC DepBank. T. Briscoe, J. Carroll.
Least influenced citer	0.00	Finding Contradictions in Text. M. de Marneffe, A. Rafferty, C. Manning.

Table 4: Least and most influential references and citers, and the influence weights along these edges, inferred by the TIRE model for three example ACL articles.

Feudal Reinforcement Learning. P. Dayan, G. Hinton		
Most influential reference	5.47	Memory-based Reinforcement Learning: Efficient Computation with Prioritized Sweeping. A. Moore, C. Atkeson.
Least influential reference	0.00	A Delay-Line Based Motion Detection Chip. T. Horiuchi, J. Lazzaro, A. Moore, C. Koch.
Most influenced citer	3.36	The Parti-Game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-Spaces. A. Moore.
Least influenced citer	1.71	Multi-time Models for Temporally Abstract Planning. D. Precup, R. Sutton.
Optimal Brain Damage. Y. Le Cun, J. Denker , S. Solla.		
Most influential reference	2.82	Comparing Biases for Minimal Network Construction with Back-Propagation. S. Hanson, L. Pratt.
Least influential reference	0.15	Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. M. Mozer, P. Smolensky.
Most influenced citer	3.08	Structural Risk Minimization for Character Recognition. I. Guyon, V. Vapnik, B. Boser, L. Bottou, S. Solla.
Least influenced citer	0.64	Structural and Behavioral Evolution of Recurrent Networks. G. Saunders, P. Angeline, J. Pollack.
An Input Output HMM Architecture. Y. Bengio, P. Frasconi.		
Most influential reference	5.29	Credit Assignment through Time: Alternatives to Backpropagation. Y. Bengio, P. Frasconi.
Least influential reference	0.00	Induction of Multiscale Temporal Structure. M. Mozer
Most influenced citer	2.66	Learning Fine Motion by Markov Mixtures of Experts. M. Meila, M. Jordan.
Least influenced citer	1.47	Recursive Estimation of Dynamic Modular RBF Networks. V. Kadirkamanathan, M. Kadirkamanathan.

Table 5: Least and most influential references and citers, and the influence weights along these edges, inferred by the TIRE model for three example NIPS articles.

in the paper as an evaluation technique. The paper most topically influenced by Chiang's paper, written by Yang and Zheng, aims to improve upon the ideas in that paper. In the NIPS corpus, the article by Bengio and Frasconi, on recurrent neural network architectures, extends previous work by the same authors, which is correctly assigned the highest topical influence. A particularly interesting case is the paper by Dayan and Hinton, which is heavily influenced by a paper by Moore, and in turn strongly influences a later paper by Moore, thus illustrating the interplay of scientific influence between authors along the citation graph. These three papers were on reinforcement learning, while the lowest scoring reference and citer were on other subjects.

6 Conclusions / Discussion

This paper introduced the notion of topical influence, a quantitative measure of scientific impact which arises from a latent variable model called topical influence regression. The model builds upon the ideas of Dirichlet-multinomial regression to encode influence relationships between articles along the citation graph. By training TIR, we can recover topical influence scores that give us insight into the impact of scientific articles. The model was applied to two scientific corpora, demonstrating the utility of the method both quantitatively and qualitatively.

In future work, the proposed framework could readily be extended to model other aspects of scientific influence, such as the effects of authors and journals on topical influence, and to exploit the context in which citations occur. From an exploratory analysis perspective, it would be instructive to compare topical influence trajectories over time for different papers. This could be further facilitated by explicitly modeling the dynamics of each article's topical influence score. The TIR framework could potentially also be applicable to other application domains such as modeling how interpersonal influence affects the spread of memes via social media.

To complement TIR, it would be useful to also have systems for identifying articles which are important for alternative reasons, such as providing methodological tools and/or demonstrating important facts. Ultimately a suite of such tools could feed into a system such as Google Scholar or Citeseer. We envision that this line of work will also be useful for building visualization tools to help researchers explore scientific corpora.

Acknowledgments

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20155. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

[A'Hearn2004] B. A'Hearn. 2004. A restricted maximum likelihood estimator for truncated height samples. *Economics & Human Biology*, 2(1):5–19.

[Blei et al.2003] D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

[Brin and Page1998] S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

[Chang and Blei2009] J. Chang and D. Blei. 2009. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88.

[Cohn and Hofmann2001] D. Cohn and T. Hofmann. 2001. The missing link-a probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems*, pages 430–436.

[Dietz et al.2007] L. Dietz, S. Bickel, and T. Scheffer. 2007. Unsupervised prediction of citation influences. In *Proceedings of the 24th International Conference on Machine Learning*, pages 233–240.

[Gerrish and Blei2010] S. Gerrish and D.M. Blei. 2010. A language-based approach to measuring scholarly impact. In *Proceedings of the 26th International Conference on Machine Learning*, pages 375–382.

[Griffiths and Steyvers2004] T.L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228.

[He et al.2009] Q. He, B. Chen, J. Pei, B. Qiu, P. Mitra, and L. Giles. 2009. Detecting topic evolution in scientific literature: how can citations help? In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 957–966. ACM.

[Le Cun et al.1990] B.B. Le Cun, JS Denker, D. Henderson, RE Howard, W. Hubbard, and LD Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404.

[Lin2008] J. Lin. 2008. Pagerank without hyperlinks: Reranking with pubmed related article networks for biomedical text retrieval. *BMC bioinformatics*, 9(1):270.

[Mimno and McCallum2008] D. Mimno and A. McCallum. 2008. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Uncertainty in Artificial Intelligence*, pages 411–418.

[Nallapati et al.2011] R. Nallapati, D. McFarland, and C. Manning. 2011. Topicflow model: Unsupervised learning of topic-specific influences of hyperlinked documents. In *International Conference on Artificial Intelligence and Statistics*, pages 543–551.

[Neal2001] R.M. Neal. 2001. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

[Radev et al.2009] D. R. Radev, P. Muthukrishnan, and V. Qazvinian. 2009. The ACL anthology network corpus. In *Proceedings, ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, pages 54–61, Singapore.

[Shaparenko and Joachims2009] B. Shaparenko and T. Joachims. 2009. Identifying the original contribution of a document via language modeling. In *Machine Learning and Knowledge Discovery in Databases*, pages 350–365. Springer.

[Teufel et al.2006] S. Teufel, A. Siddharthan, and D. Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 103–110. Association for Computational Linguistics.

[Wallach et al.2009] H.M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM.

[Wallach2006] H.M. Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984. ACM.

[Ziman1968] J.M. Ziman. 1968. *Public knowledge: an essay concerning the social dimension of science*. Cambridge University Press.

Joint Parsing and Disfluency Detection in Linear Time

Mohammad Sadegh Rasooli*
Department of Computer Science
Columbia University, New York, NY
rasooli@cs.columbia.edu

Joel Tetreault
Nuance Communications, Inc.
Sunnyvale, CA
joel.tetreault@nuance.com

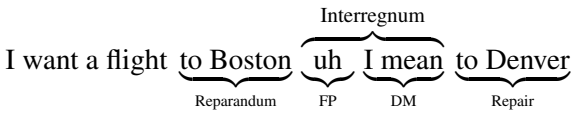
Abstract

We introduce a novel method to jointly parse and detect disfluencies in spoken utterances. Our model can use arbitrary features for parsing sentences and adapt itself with out-of-domain data. We show that our method, based on transition-based parsing, performs at a high level of accuracy for both the parsing and disfluency detection tasks. Additionally, our method is the fastest for the joint task, running in linear time.

1 Introduction

Detecting disfluencies in spontaneous speech has been widely studied by researchers in different communities including natural language processing (e.g. Qian and Liu (2013)), speech processing (e.g. Wang et al. (2013)) and psycholinguistics (e.g. Finlayson and Corley (2012)). While the percentage of spoken words which are disfluent is typically not more than ten percent (Bortfeld et al., 2001), this additional "noise" makes it much harder for spoken language systems to predict the correct structure of the sentence.

Disfluencies can be filled pauses (e.g. "uh", "um", "huh"), discourse markers (e.g. "you know", "I mean") or edited words which are repeated or corrected by the speaker. For example, in the following sentence, an edited phrase or *reparandum interval* ("to Boston") occurs with its repair ("to Denver"), a filled pause ("uh") and discourse marker ("I mean").[1]



Filled pauses and discourse markers are to some extent a fixed and closed set. The main challenge in finding disfluencies is the case where the edited phrase is neither a rough copy of its repair or has any repair phrase (i.e. discarded edited phrase). Hence, in previous work, researchers report their method performance on detecting edited phrases (reparandum) (Johnson and Charniak, 2004).

In contrast to most previous work which focuses solely on either detection or on parsing, we introduce a novel framework for jointly parsing sentences with disfluencies. To our knowledge, our work is the first model that is based on *joint dependency and disfluency detection*. We show that our model is robust enough to detect disfluencies with high accuracy, while still maintaining a high level of dependency parsing accuracy that approaches the upper bound. Additionally, our model outperforms prior work on joint parsing and disfluency detection on the disfluency detection task, and improves upon this prior work by running in linear time complexity.

The remainder of this paper is as follows. In §2, we overview some the previous work on disfluency detection. §3 describes our model. Experiments are described in §4 and Conclusions are made in §5.

* The first author worked on this project while he was a research intern in CoreNL research group, NLU lab, Nuance Communications, Sunnyvale, CA.

[1] In the literature, edited words are also known as "reparandum", and the fillers are known as "interregnum". Filled pauses are also called "Interjections".

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 124–129,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Disfluency detection approaches can be divided into two different groups: text-first and speech first (Nakatani and Hirschberg, 1993). In the first approach, all prosodic and acoustic cues are ignored while in the second approach both grammatical and acoustic features are considered. For this paper, we focus on developing a text-first approach but our model is easily flexible with speech-first features because there is no restriction on the number and types of features in our model.

Among text-first approaches, the work is split between developing systems which focus specifically on disfluency detection and those which couple disfluency detection with parsing. For the former, Charniak and Johnson (2001) employ a linear classifier to predict the edited phrases in Switchboard corpus (Godfrey et al., 1992). Johnson and Charniak (2004) use a TAG-based noisy channel model to detect disfluencies while parsing with getting n-best parses from each sentence and re-ranking with a language model. The original TAG parser is not used for parsing itself and it is used just to find rough copies in the sentence. Their method achieves promising results on detecting edited words but at the expense of speed (the parser has a complexity of O(N^5). Kahn et al. (2005) use the same TAG model and add semi-automatically extracted prosodic features. Zwarts and Johnson (2011) improve the performance of TAG model by adding external language modeling information from data sets such as Gigaword in addition to using minimal expected F-loss in n-best re-ranking.

Georgila (2009) uses integer linear programming combined with CRF for learning disfluencies. That work shows that ILP can learn local and global constraints to improve the performance significantly. Qian and Liu (2013) achieve the best performance on the Switchboard corpus (Godfrey et al., 1992) without any additional data. They use three steps for detecting disfluencies using weighted Max-Margin Markov (M^3) network: detecting fillers, detecting edited words, and refining errors in previous steps.

Some text-first approaches treat parsing and disfluency detection jointly, though the models differ in the type of parse formalism employed. Lease and Johnson (2006) use a PCFG-based parser to parse sentences along with finding edited phrases. Miller and Schuler (2008) use a right-corner transform of binary branching structures on bracketed sentences but their results are much worse than (Johnson and Charniak, 2004). To date, none of the prior joint approaches have used a dependency formalism.

3 Joint Parsing Model

We model the problem using a deterministic transition-based parser (Nivre, 2008). These parsers have the advantage of being very accurate while being able to parse a sentence in linear time. An additional advantage is that they can use as many non-local and local features as needed.

Arc-Eager Algorithm We use the arc-eager algorithm (Nivre, 2004) which is a bottom-up parsing strategy that is used in greedy and k-beam transition-based parsers. One advantage of this strategy is that the words can get a head from their left side, before getting right dependents. This is particularly beneficial for our task, since we know that reparanda are similar to their repairs. Hence, a reparandum may get its head but whenever the parser faces a repair, it removes the reparandum from the sentence and continues its actions.

The actions in an arc-eager parsing algorithm are:

- **Left-arc (LA)**: The first word in the buffer becomes the head of the top word in the stack. The top word is popped after this action.
- **Right-arc (RA)**: The top word in the stack becomes the head of the first word in the buffer.
- **Reduce (R)**: The top word in the stack is popped.
- **Shift (SH)**: The first word in the buffer goes to the top of the stack.

Joint Parsing and Disfluency Detection We first extend the arc-eager algorithm by augmenting the action space with three new actions:

- **Reparandum (Rp[i;j])**: treat a phrase (words *i* to *j*) outside the look-ahead buffer as a reparandum. Remove them from the sentence and clear their dependencies.
- **Discourse Marker (Prn[i])**: treat a phrase in the look-ahead buffer (first *i* words) as a discourse marker and remove them from the sentence.

Stack	Buffer	Act.
flight	to Boston uh I mean ...	RA
flight to	Boston uh I mean to ...	RA
flight to Boston	**uh** I mean to Denver	Intj[1]
flight to Boston	**I mean** to Denver	Prn[1]
flight **to Boston**	to Denver	RP[2:3]
flight	to Denver	RA
flight to	Denver	RA
flight to Denver		R
flight to		R
flight		R

Figure 1: A sample transition sequence for the sentence "flight to Boston uh I mean to Denver". In the third column, only the underlined parse actions are learned by the parser (second classifier). The first classifier uses all instances for training (learns fluent words with "regular" label).

- **Interjection (Intj[i])**: treat a phrase in the look-ahead buffer (first *i* words) as a filled pause and remove them from the sentence.[2]

Our model has two classifiers. The first classifier decides between four possible actions and possible candidates in the current configuration of the sentence. These actions are the three new ones from above and a new action **Regular (Reg)**: which means do one of the original arc-eager parser actions.

At each configuration, there might be several candidates for being a *prn*, *intj* or *reparandum*, and one *regular* candidate. The candidates for being a *reparandum* are a set of words outside the look-ahead buffer and the candidates for being an *intj* or *prn* are a set of words beginning from the head of the look-ahead buffer. If the parser decides *regular* as the correct action, the second classifier predicts the best parsing transition, based on arc-eager parsing (Nivre, 2004).

For example, in the 4th state in Figure 1, there are multiple candidates for the first classifier: regular, "I" as *prn*[1] or *intj*[1], "I mean" as *prn*[2] or *intj*[2], "I mean to" as *prn*[3] or *intj*[3], "I mean to Denver" as *prn*[4] or *intj*[4], "Boston" as *rp*[3:3], "to Boston" as *rp*[2:3], and "flight to Boston" as *rp*[1:3].

[2]In the bracketed version of Switchboard corpus, reparandum is tagged with *EDITED* and discourse markers and paused fillers are tagged as *PRN* and *INTJ* respectively.

Training A transition-based parser action (our second-level classifier) is sensitive to the words in the buffer and stack. The problem is that we do not have gold dependencies for edited words in our data. Therefore, we need a parser to remove reparandum words from the buffer and push them into the stack. Since our parser cannot be trained on disfluent sentences from scratch, the first step is to train it on clean treebank data.

In the second step, we adapt parser weights by training it on disfluent sentences. Our assumption is that we do not know the correct dependencies between disfluent words and other words in the sentence. At each configuration, the parser updates itself with new instances by traversing all configurations in the sentences. In this case, if at the head of the buffer there is an *intj* or *prn* tag, the parser allows them to be removed from the buffer. If a reparandum word is not completely outside the buffer (the first two states in Figure 1), the parser decides between the four regular arc-eager actions (i.e. *left-arc*, *right-arc*, *shift*, and *reduce*). If the last word pushed into the stack is a reparandum and the first word in the buffer is a regular word, the parser removes all reparanda at the same level (in the case of nested edited words), removes their dependencies to other words and push their dependents into the stack. Otherwise, the parser performs the oracle action and adds that action as its new instance.[3]

With an adapted parser which is our second-level classifier, we can train our first-level classifier. The same procedure repeats, except that instances for disfluency detection are used for updating parameter weights for the first classifier for deciding the actions. In Figure 1, only the oracle actions (underlined) are added to the instances for updating parser weights but all first-level actions are learned by the first level classifier.

4 Experiments and Evaluation

For our experiments, we use the Switchboard corpus (Godfrey et al., 1992) with the same train/dev/test split as Johnson and Charniak (2004). As in that

[3]The reason that we use a parser instead of expanding all possible transitions for an edited word is that, the number of *regular* actions will increase and the other actions become sparser than natural.

work, incomplete words and punctuations are removed from data (except that we do not remove incomplete words that are not disfluent[4]) and all words are turned into lower-case. The main difference with previous work is that we use Switchboard *mrg* files for training and testing our model (since they contain parse trees) instead of the more commonly used Swithboard *dps* text files. *Mrg* files are a subset of *dps* files with about more than half of their size. Unfortunately, the disfluencies marked in the *dps* files are not exactly the same as those marked in the corresponding *mrg* files. Hence, our result is not completely comparable to previous work except for (Kahn et al., 2005; Lease and Johnson, 2006; Miller and Schuler, 2008).

We use Tsurgeon (Levy and Andrew, 2006) for extracting sentences from *mrg* files and use the Penn2Malt tool[5] to convert them to dependencies. Afterwards, we provide dependency trees with disfluent words being the dependent of nothing.

Learning For the first classifier, we use averaged structured Perceptron (AP) (Collins, 2002) with a minor modification. Since the first classifier data is heavily biased towards the "regular label", we modify the weight updates in the original algorithm to 2 (original is 1) for the cases where a "reparandum" is wrongly recognized as another label. We call the modified version "weighted averaged Perceptron (WAP)". We see that this simple modification improves the model accuracy.[6] For the second classifier (parser), we use the original averaged structured Perceptron algorithm. We report results on both AP and WAP versions of the parser.

Features Since for every state in the parser configuration, there are many candidates for being disfluent; we use local features as well as global features for the first classifier. Global features are mostly useful for discriminating between the four actions and local features are mostly useful for choosing a phrase as a candidate for being a disfluent phrase. The features are described in Figure 2. For the second classifier, we use the same features as (Zhang and Nivre, 2011, Table 1) except that we train our parser in a similar manner as the MaltParser (Nivre et al., 2007) without k-beam training.

Global Features
First n words inside/outside buffer (n=1:4)
First n POS i/o buffer (n=1:6)
Are n words i/o buffer equal? (n=1:4)
Are n POS i/o buffer equal? (n=1:4)
n last FG transitions (n=1:5)
n last transitions (n=1:5)
n last FG transitions + first POS in the buffer (n=1:5)
n last transitions + first POS in the buffer (n=1:5)
(n+m)-gram of m/n POS i/o buffer (n,m=1:4)
Refined (n+m)-gram of m/n POS i/o buffer (n,m=1:4)
Are n first words of i/o buffer equal? (n=1:4)
Are n first POS of i/o buffer equal? (n=1:4)
Number of common words i/o buffer words (n=1:6)
Local Features
First n words of the candidate phrase (n=1:4)
First n POS of the candidate phrase (n=1:6)
Distance between the candidate and first word in the buffer

Figure 2: Features used for learning the first classifier. Refined n-gram is the n-gram without considering words that are recognized as disfluent. Fine-grained (FG) transitions are enriched with parse actions (e.g. "regular:left-arc").

Parser Evaluation We evaluate our parser with both unlabeled attachment accuracy of correct words and precision and recall of finding the dependencies of correct words.[7] The second classifier is trained with 3 iterations in the first step and 3 iterations in the second step. We use the attachment accuracy of the parse tree of the correct sentences (without disfluencies) as the upper-bound attachment score and parsed tree of the disfluent sentences (without disfluency detection) as our lower-bound attachment score. As we can see in Table 1, WAP does a slightly better job parsing sentences. The upper-bound parsing accuracy shows that we do not lose too much information while jointly detecting disfluencies. Our parser is not comparable to (Johnson and Charniak, 2004) and (Miller and Schuler, 2008), since we use dependency relations for evaluation instead of constituencies.

Disfluency Detection Evaluation We evaluate our model on detecting edited words in the sentences

[4] E.g. I want t- go to school.

[5] `http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html`

[6] This is similar to WM^{3}N in (Qian and Liu, 2013).

[7] The parser is actually trained to do labeled attachment and labeled accuracy is about 1-1.5% lower than UAS.

	UAS	LB	UB	Pr.	Rec.	F2
AP	**88.6**	70.7	90.2	86.8	88.0	87.4
WAP	88.1	70.7	90.2	87.2	88.0	**87.6**

Table 1: Parsing results. UB = upperbound (parsing clean sentences), LB = lowerbound (parsing disfluent sentences without disfluency correction). UAS is unlabeled attachment score (accuracy), Pr. is precision, Rec. is recall and F1 is f-score.

	Pr.	Rec.	F1
AP	**92.9**	71.6	80.9
WAP	85.1	**77.9**	**81.4**
KL (2005)	–	–	78.2
LJ (2006)	–	–	62.4
MS (2008)	–	–	30.6
QL (2013) – Default	–	–	81.7
QL (2013) – Optimized	–	–	**82.1**

Table 2: Disfluency results. Pr. is precision, Rec. is recall and F1 is f-score. KL = (Kahn et al., 2005), LJ = (Lease and Johnson, 2006), MS = (Miller and Schuler, 2008) and QL = (Qian and Liu, 2013).

(words with "EDITED" tag in *mrg* files). As we see in Table 2, WAP works better than the original method. As mentioned before, the numbers are not completely comparable to others except for (Kahn et al., 2005; Lease and Johnson, 2006; Miller and Schuler, 2008) which we outperform. For the sake of comparing to the state of the art, the best result for this task (Qian and Liu, 2013) is replicated from their available software[8] on the portion of *dps* files that have corresponding *mrg* files. For a fairer comparison, we also optimized the number of training iterations of (Qian and Liu, 2013) for the *mrg* set based on dev data (10 iterations instead of 30 iterations). As shown in the results, our model accuracy is slightly less than the state-of-the-art (which focuses solely on the disfluency detection task and does no parsing), but we believe that the performance can be improved through better features and by changing the model. Another characteristic of our model is that it operates at a very high precision, though at the expense of some recall.

[8]We use the second version of the code: `http://code.google.com/p/disfluency-detection/`. Results from the first version are 81.4 and 82.1 for the default and optimized settings.

5 Conclusion

In this paper, we have developed a fast, yet accurate, joint dependency parsing and disfluency detection model. Such a parser is useful for spoken dialogue systems which typically encounter disfluent speech and require accurate syntactic structures. The model is completely flexible with adding other features (either text or speech features).

There are still many ways of improving this framework such as using k-beam training and decoding, using prosodic and acoustic features, using out of domain data for improving the language and parsing models, and merging the two classifiers into one through better feature engineering. It is worth noting that we put the dummy *root* word in the first position of the sentence. Ballesteros and Nivre (2013) show that parser accuracy can improve by changing that position for English.

One of the main challenges in this problem is that most of the training instances are not disfluent and thus the sample space is very sparse. As seen in the experiments, we can get further improvements by modifying the weight updates in the Perceptron learner. In future work, we will explore different learning algorithms which can help us address the sparsity problem and improve the model accuracy. Another challenge is related to the parser speed, since the number of candidates and features are much greater than the number used in classical dependency parsers.

Acknowledgements We would like to thank anonymous reviewers for their helpful comments on the paper. Additionally, we were aided by researchers by their prompt responses to our many questions: Mark Core, Luciana Ferrer, Kallirroi Georgila, Mark Johnson, Jeremy Kahn, Yang Liu, Xian Qian, Kenji Sagae, and Wen Wang. Finally, this work was conducted during the first author's summer internship at the Nuance Sunnyvale Research Lab. We would like to thank the researchers in the group for the helpful discussions and assistance on different aspects of the problem. In particular, we would like to thank Chris Brew, Ron Kaplan, Deepak Ramachandran and Adwait Ratnaparkhi.

References

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13.

Heather Bortfeld, Silvia D. Leon, Jonathan E. Bloom, Michael F. Schober, and Susan E. Brennan. 2001. Disfluency rates in conversation: Effects of age, relationship, topic, role, and gender. *Language and Speech*, 44(2):123–147.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *NAACL-HLT*, pages 1–9.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *ACL*, pages 1–8.

Ian R. Finlayson and Martin Corley. 2012. Disfluency in dialogue: an intentional signal from the speaker? *Psychonomic bulletin & review*, 19(5):921–928.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *NAACL-HLT*, pages 109–112.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *ICASSP*, volume 1, pages 517–520.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *ACL*, pages 33–39.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *EMNLP*, pages 233–240.

Matthew Lease and Mark Johnson. 2006. Early deletion of fillers in processing conversational speech. In *NAACL-HLT*, pages 73–76.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *LREC*, pages 2231–2234.

Tim Miller and William Schuler. 2008. A unified syntactic model for parsing fluent and disfluent speech. In *ACL-HLT*, pages 105–108.

Christine Nakatani and Julia Hirschberg. 1993. A speech-first model for repair detection and correction. In *ACL*, pages 46–53.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *NAACL-HLT*, pages 820–825.

Wen Wang, Andreas Stolcke, Jiahong Yuan, and Mark Liberman. 2013. A cross-language study on automatic speech disfluency detection. In *NAACL-HLT*, pages 703–708.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *ACL (Short Papers)*, pages 188–193.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *ACL*, pages 703–711.

Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-0192, Japan
{masashi-t,kevinduh,shimbo,matsu}@is.naist.jp

Abstract

We present a novel vector space model for semantic co-compositionality. Inspired by Generative Lexicon Theory (Pustejovsky, 1995), our goal is a compositional model where both predicate and argument are allowed to modify each others' meaning representations while generating the overall semantics. This readily addresses some major challenges with current vector space models, notably the polysemy issue and the use of one representation per word type. We implement *co-compositionality* using *prototype projections* on predicates/arguments and show that this is effective in adapting their word representations. We further cast the model as a neural network and propose an unsupervised algorithm to jointly train word representations with co-compositionality. The model achieves the best result to date ($\rho = 0.47$) on the semantic similarity task of transitive verbs (Grefenstette and Sadrzadeh, 2011).

1 Introduction

Vector space models of words have been very successful in capturing the semantic and syntactic characteristics of individual lexical items (Turney and Pantel, 2010). Much research has addressed the question of how to construct individual word representations, for example distributional models (Mitchell and Lapata, 2010) and neural models (Collobert and Weston, 2008). These word representations are used in various natural language processing (NLP) tasks such as part-of-speech tagging, chunking, named entity recognition, and semantic role labeling (Turian et al., 2010; Collobert et al., 2011).

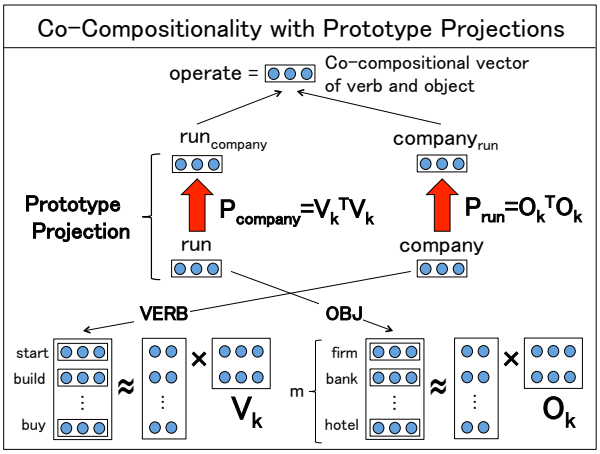


Figure 1: Here, we capture the semantics of *run* in *run company* by projecting the original word representation of *run* to the prototype space of *company* (and vice versa).

Recently, modeling of semantic compositionality (Frege, 1892) in vector space has emerged as another important line of research (Mitchell and Lapata, 2008; Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Socher et al., 2012; Grefenstette and Sadrzadeh, 2011; Van de Cruys et al., 2013). The goal is to formulate how individual word representations ought to be combined to achieve phrasal or sentential semantics.

The main questions for semantic compositionality that we are concerned with are: (1) how can polysemy be handled by a single vector representation per word type, learned by either a distributional or neural model, and (2) how does composition resolve

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 130–140,
Seattle, Washington, USA, 18-21 October 2013.

these ambiguities. To this end, we are inspired by the idea of *type coercion* and *co-compositionality* in Generative Lexicon Theory (Pustejovsky, 1995). Co-compositionality advocates that instead of a predicate-argument view of composition, both predicate and argument influence/coerce each other to generate the overall meaning. For example, consider a polysemous word like *run*:

- (a) He runs the company.
- (b) He runs the marathon.

Run may have several senses, but the prototypical verbs that select for *company* differ from those that select for *marathon*, and thus the ambiguity at the word level is resolved at the sentence level. The same is true for the other direction, where the predicate also coerces meaning to the argument to fit expectation.

We believe that models for semantic composition ought to incorporate elements of co-compositionality. We propose such a model here, using what we call *prototype projections*. For each predicate, we transform its vector representation by projecting it into a latent space that is prototypical of its argument. This projection is performed analogously for each argument as well, and the final meaning is computed by composition of these transformed vectors (Figure 1). In addition, the model is cast as a neural network where word representations could be re-trained or fine-tuned.[1]

Our contributions are two-fold:

1. We propose a novel model for semantic co-compositionality. This model, based on prototype projections, is easy to implement and achieves state-of-the-art performance in the sentence similarity dataset developed by Grefenstette and Sadrzadeh (2011).

2. Our results empirically confirm that existing word representations (eg., SDS and NLM in Section 2) are sufficiently effective at capturing polysemy, as long as we have the proper mechanism to tease out the proper sense during composition. We further propose an unsupervised neural network training algorithm that jointly fine-tunes the word representations within the co-composition model, resulting in even better performance on the sentence similarity task.

We would like to emphasize the second contribution especially. Semantics research is divided in two strands, one focusing on learning word representations without consideration for compositionality, and the other focusing on compositional semantics using the representations only as an input. But issues are actually related from the linguistics perspective, and even more so if we adopt a Generative Lexicon perspective. Our neural network model bridges these two strands of research by modeling co-compositionality and learning word representations simultaneously. We note that methods using context effects have been explored by Erk and Padó (2008; 2009) and Thater et al. (2010; 2011), but to the best of our knowledge, ours is the first model to perform co-compositionality and learning of word representations jointly.

In the following, we first provide background to the word representations employed here (Section 2). We describe the model for co-compositionality in Section 3 and the corresponding neural network in Section 4. Evaluation and experiments are presented in Sections 5 and 6. Finally, we end with related work (Section 7) and conclusions (Section 8).

2 Word Vector Representations

2.1 Simple Distributional Semantic space (SDS) word vectors

Word meaning is often represented in a high dimensional space, where each element corresponds to some contextual element in which the word is found. Mitchell and Lapata (2010) present a co-occurrence-based semantic space called Simple Distributional Semantic space (SDS). Their SDS model uses a context window of five words on either side of the target word and 2,000 vector components, representing the most frequent context words (excluding a list of stop words). These components $v_i(t)$ were set to the ratio of the probability of the context word given the

[1] While we are inspired by co-compositionality, it is important to note that our model does not implement qualia structure and other important components of Generative Lexicon Theory. We operate within the vector space model of distributional semantics, so these ideas are implemented with matrix algebra, which is a natural fit with neural networks.

target word to the probability of the context word overall:

$$v_i(t) = \frac{p(c_i|t)}{p(c_i)} = \frac{freq_{c_i,t} \times freq_{total}}{freq_t \times freq_{c_i}} \quad (1)$$

where $freq_{c_i,t}$, $freq_{total}$, $freq_t$ and $freq_{c_i}$ are the frequencies of the context word c_i with the target word t, the total count of all word tokens, the frequency of the target word t, and the frequency of the context word c_i, respectively.

2.2 Neural Language Model (NLM) word embeddings

Another popular way to learn word representations is based on the Neural Language Model (NLM) (Bengio et al., 2003). In comparison with SDS, NLM tend to be low-dimensional (e.g. 50 dimensions) but employ dense features. These dense feature vectors are usually called word embeddings, and it has been shown that such vectors can capture interesting linear relationships, such as $king - man + woman \approx queen$ (Mikolov et al., 2013). In this work, we adopt the model by Collobert and Weston (2008). The idea is to construct a neural network based on word sequences, where one outputs high scores for n-grams that occur in a large unlabeled corpus and low scores for *nonsense* n-grams where one word is replaced by a random word. This word representation with NLM has been used to good effect, for example in (Turian et al., 2010; Collobert et al., 2011; Huang et al., 2012) where induced word representations are used with sophisticated features to improve performance in various NLP tasks.

Specifically, we first represent the word sequence as a vector $\mathbf{x} = [\mathbf{d}(w_1); \mathbf{d}(w_2); \ldots; \mathbf{d}(w_m)]$, where w_i is i_{th} word in the sequence, m is the window size, $\mathbf{d}(w)$ is the vector representation of word w (an n-dimensional column vector) and $[\mathbf{d}(w_1); \mathbf{d}(w_2); \ldots; \mathbf{d}(w_m)]$ is the concatenation of word vectors as an input of neural network. Second, we compute the score of the sequence,

$$score(\mathbf{x}) = \mathbf{s}^{\mathrm{T}}(\tanh(\mathbf{W}\mathbf{x} + \mathbf{b})) \quad (2)$$

where $\mathbf{W} \in \mathbb{R}^{h \times (mn)}$ and $\mathbf{s} \in \mathbb{R}^h$ are the first and second layer weights of the neural network, and $\mathbf{b} \in \mathbb{R}^h$ is the bias unit of hidden layer. The superscript T represents transposition, and $\tanh$ is applied element-wise. We also create a corrupted sequence $\mathbf{x}_c = [\mathbf{d}(w_1); \mathbf{d}(w_2); \ldots; \mathbf{d}(w_m{}')]$ where $w_m{}'$ is chosen randomly from the vocabulary. We compute the score of this implicit negative sequence $\mathbf{x}_c$ with the same neural network, $score(\mathbf{x}_c) = \mathbf{s}^{\mathrm{T}}(\tanh(\mathbf{W}\mathbf{x}_c + \mathbf{b}))$. Finally, we get the cost function of this training algorithm as follow.

$$J = \max(0, 1 - score(\mathbf{x}) + score(\mathbf{x}_c)) \quad (3)$$

In order to minimize this cost function, we optimize the parameters $\theta = (\mathbf{s}, \mathbf{W}, \mathbf{b}, \mathbf{x})$ via backpropagation with stochastic gradient descent (SGD).

3 The Model

3.1 Prototype Projection

Generative Lexicon Theory (Pustejovsky, 1995) makes a distinction between accidental polysemy (homonyms, e.g. *bank* as financial institution vs. as river side) and logical polysemy (e.g. figure and ground meanings of *door*). Our model handles both cases using the concept of projection to latent *prototype* space. The fundamental idea is that for each word w and a syntactic/semantic (binary) relation R (such as verb-object relation), w has a set of prototype words with which it frequently occurs in relation R. For example, if w is a word *company*, and R is the object-verb relation, prototype words should include *start*, *build*, and *buy* (Figure 1). For each word-relation pair, we pre-compute the latent semantic subspace spanned by these prototype words.

Later, when we encounter a phrase expressing a relation R between two words w_1 and w_2, each word is first projected onto a latent subspace determined by the other word and relation R. The projection operation shifts the meaning of individual words in accordance with context, and through this operation we realize coercion/co-composition. And finally, the meaning of the phrase is computed from the two projected points in the semantic space.

Let us describe how to compute the latent subspace associated with a word w_0 and a relation R. First, we collect from a corpus a set of prototype words that occur frequently in relation R with target word w_0. So for example in Figure 1, if $w_0 =$

verb	object	landmark	similarity(verb, landmark)	similarity(projected verb, landmark)
run	company	operate	0.40	0.70
meet	criterion	satisfy	0.49	0.71
spell	name	write	0.04	0.50

Table 1: Examples of verb-object pairs. Original verb and landmark verb similarity, prototype projected verb and landmark verb similarity, as measure by cosine using Collobert and Weston's word embeddings. *Meet* has a abstract meaning itself, but after prototype projection with matrix constructed by word vectors of $W(\textit{VerbOf}, \textit{criterion})$, *meet* is more close to meaning of *satisfy*.

company, and R = *VerbOf* is the object-verb relation,

$$W(\textit{VerbOf}, \textit{company}) = \{\textit{start}, \textit{build}, \ldots, \textit{buy}\}.$$

Now let $W(R, w_0) = \{w_1, w_2, \cdots, w_m\}$ be the m prototype words we collected, and let $\mathbf{d}(w)$ denote the n-dimensional (column) vector representation of word w (either by SDS or NLM representation). We make an $m \times n$ matrix $\mathbf{C}_{(R,w_0)}$ by stacking the prototype word vectors, i.e.,

$$\mathbf{C}_{(R,w_0)} = [\mathbf{d}(w_1), \mathbf{d}(w_2), \cdots, \mathbf{d}(w_m)]^{\mathrm{T}} \quad (4)$$

and then apply Singular Value Decomposition (SVD) to extract the latent space from this matrix:

$$\mathbf{C}_{(R,w_0)} \approx \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}}. \quad (5)$$

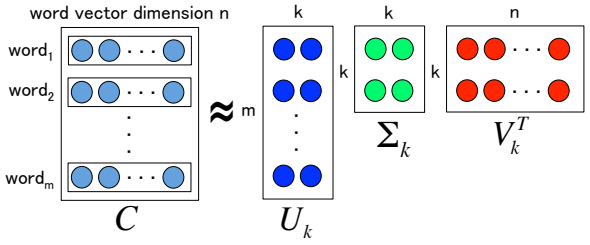


Figure 2: Graphical representation of SVD in our model.

Figure 2 shows the graphical representation of this matrix factorization. In NLP tasks, SVD is often applied to a term-document matrix, but in our model, we apply SVD to the matrix consisting of word vectors.

Intuitively, $\mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}}$ represents the latent subspace formed by prototypical words $W(R, w_0) = \{w_1, w_2, \cdots, w_m\}$. We call this matrix the *prototype space* of word w_0 with respect to relation R.

Note that the matrix of orthogonal projection onto this prototype space is given by $\mathbf{P}_{(R,w_0)} = (\mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}})^{\mathrm{T}} (\mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}})$. Hence, when we observe a relation $R(w_0, w)$, the projected representation of word w in this context is computed by $\mathbf{prpj}_{(R,w_0)}(w)$ defined as follows:

$$\mathbf{prpj}_{(R,w_0)}(w) = \mathbf{P}_{(R,w_0)} \mathbf{d}(w). \quad (6)$$

Table 1 shows several examples of how meanings change after prototype projection using word embeddings of Collobert and Weston (2008).[2]

3.2 Co-Compositionality

In order to model co-compositionality, we apply prototype projection to both the verb and the object. In particular, suppose verb is w_v and object is w_o, $\mathbf{C}_{(\textit{VerbOf},w_o)}$ is used to project w_v and $\mathbf{C}_{(\textit{ObjOf},w_v)}$ is used to project w_o. The vector that represents the overall meaning of verb-object with prototype projection is computed by:

$$\mathbf{cocomp}(w_v, w_o) = f(\mathbf{prpj}_{(\textit{VerbOf},w_o)}(w_v), \mathbf{prpj}_{(\textit{ObjOf},w_v)}(w_o)) \quad (7)$$

Function f can be a compositional computation like simple addition or element-wise multiplication of two vectors. This is graphically shown in Figure 1.

4 Unsupervised Learning of Co-Compositionality

In this section, we propose a new neural language model that learns word representations while jointly accounting for compositional semantics. One central assumption of our work (and many other works in compositional semantics) is that a single vector

[2] ronan.collobert.com/senna/

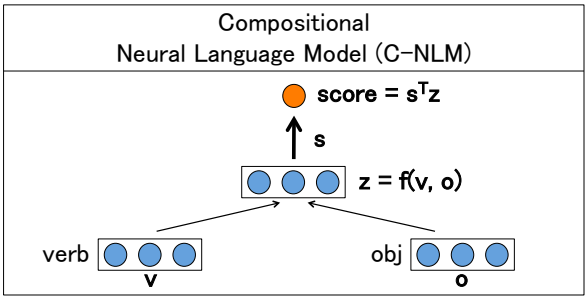


Figure 3: Compositional Neural Language Model (C-NLM).

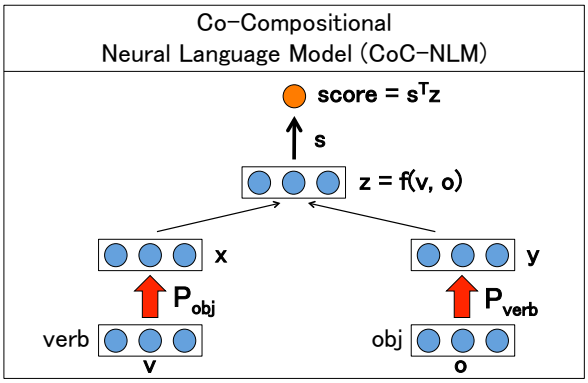


Figure 4: Co-Compositional Neural Language Model (CoC-NLM) is C-NLM with prototype projection.

per word type sufficiently represents the multiple meanings and usage patterns of a word.[3] That means that for a polysemous word, its word vector actually represents an aggregation of the distinctly different contexts it occurs in. We will show that such an assumption is quite reasonable under our model, since the prototype projections successfully tease out the proper semantics from these aggregate representations.

However, it is natural to wonder whether one can do better if one incorporates the compositional model into the training of the word representations in the first place. To do so, we formulate a novel model called *Compositional Neural Language Model* (Section 4.1). This model is a combination of an unsupervised training algorithm with basic compositionality (addition/multiplications). Then, we extend this model with the projection idea in section 3.2 to formulate a *Co-Compositional Neural Language Model* (Section 4.2).

4.1 Compositional Neural Language Model (C-NLM)

Compositional Neural Language Model (C-NLM) is a combination of a word representation learning method and compositional rule. In contrast to other compositional models based on machine learning, our model has no complex parameters for modeling composition. Composition is modeled using straightforward vector addition/multiplications; instead, what is learned is the word representation.

Figure 3 shows the C-NLM. The learning algorithm is unsupervised, and works by artificially generating negative examples in a fashion analogous to the NLM learning algorithm of (Collobert and Weston, 2008) and contrastive estimation (Smith and Eisner, 2005). First, given some initial word representations and raw sentences, we compute the compositional vector with function f (in this section, we will assume that we will be using the addition operator). Second, in order to obtain the score of compositional vector, we compute the dot product with vector $\mathbf{s} \in \mathbb{R}^n$ (n is the dimension of the word vector space): verb vector $\mathbf{v} = \mathbf{d}(w_v)$ and object vector $\mathbf{o} = \mathbf{d}(w_o)$.

$$score(\mathbf{v}, \mathbf{o}) = \mathbf{s}^{\mathrm{T}} f(\mathbf{v}, \mathbf{o}) = \mathbf{s}^{\mathrm{T}}(\mathbf{v} + \mathbf{o}) \qquad (8)$$

We also create a corrupted pair by substituting a random verb $w_{verb}{}'$. The cost function $J = \max(0, 1 - score(\mathbf{v}, \mathbf{o}) + score(\mathbf{v}_c, \mathbf{o}))$, where $\mathbf{v}_c$ is the word vector of $w_{verb}{}'$, encourages that the score of correct pair is higher than the score of the corrupt pair. Let $\mathbf{z} = \mathbf{v} + \mathbf{o}$, our model parameters are $\theta = (\mathbf{s}, \mathbf{z}, \mathbf{v})$. The optimization is divided into two steps:

1. Optimize $\mathbf{s}$ and $\mathbf{z}$ via SGD.

2. Let $\mathbf{z}_{new}$ be the updated $\mathbf{z}$ via step 1. The new verb vector $\mathbf{v}_{new}$ trained within additive compositionality is just $\mathbf{v}_{new} = \mathbf{z}_{new} - \mathbf{o}$. Note that if we also want to optimize $\mathbf{o}$, we may want to also corrupt the object and run SGD in step 2 as well.

4.2 Co-Compositional Neural Language Model (CoC-NLM)

We now add prototype projection into C-NLM, making our final model: *Co-Compositional Neural*

[3] There are works on *multiple* representations, e.g., (Reisinger and Mooney, 2010); we focus on single representation here.

Language Model (CoC-NLM). We define the score function as dot product of $\mathbf{s}$ and additional vector of prototype projected vectors (Figure 4). Let $\mathbf{P}_{obj} = \mathbf{P}_{(VerbOf,w_o)}$ and $\mathbf{P}_{verb} = \mathbf{P}_{(ObjOf,w_v)}$,

$$score(\mathbf{v}, \mathbf{o}) = \mathbf{s}^{\mathrm{T}}(\mathbf{P}_{obj}\mathbf{v} + \mathbf{P}_{verb}\mathbf{o}). \quad (9)$$

Let $\mathbf{x} = \mathbf{P}_{obj}\mathbf{v}$, $\mathbf{y} = \mathbf{P}_{verb}\mathbf{o}$ and $\mathbf{z} = \mathbf{x} + \mathbf{y}$. Our model parameters are $\theta = (\mathbf{s}, \mathbf{z}, \mathbf{v})$. The optimization algorithm of CoC-NLM is divided into three steps like C-NLM. First, we optimize $\mathbf{s}$ and $\mathbf{z}$. Second, the projected verb vector is updated as $\mathbf{x}_{new} = \mathbf{z}_{new} - \mathbf{y}$. Finally we optimize $\mathbf{v}$ to minimize the Euclidean distance between $\mathbf{x}_{new}$ and $\mathbf{P}_{obj}\mathbf{v}$, where λ is a regularization hyper-parameter:

$$J(\mathbf{v}) = \frac{1}{2}||\mathbf{x}_{new} - \mathbf{P}_{obj}\mathbf{v}||^2 + \frac{\lambda}{2}\mathbf{v}^{\mathrm{T}}\mathbf{v} \quad (10)$$

5 Evaluation

5.1 Dataset

In order to evaluate the performance of our new co-compositional model with prototype projection and word representation learning algorithm, we make use of the disambiguation task of transitive sentences developed by Grefenstette and Sadrzadeh (2011). This is an extension of the two words phrase similarity task defined in Mitchell and Lapata (2008), and constructed according to similar guidelines. The dataset consists of similarity judgments between a *landmark* verb and a triple consisting of a transitive target verb, subject and object extracted from the BNC corpus. Human judges give scores between 1 to 7, with higher scores implying higher semantic similarity. For example, Table 2 shows some examples from the data: we see that the verb *meet* with subject *system* and object *criterion* is judged similar to the landmark verb *satisfy* but not *visit*. The dataset contains a total of 2500 similarity judgements, provided by 25 participants.[4] The task is to have the model produce a score for each pair of landmark verb and verb-subject-object triple. Models are evaluated by computing the Spearman's ρ correlation between its similarity scores and that of the human judgments.

[4] http://www.cs.ox.ac.uk/people/edward.grefenstette/

verb	subj	obj	landmark	sim
meet	system	criterion	satisfy	6
meet	system	criterion	visit	1
write	student	name	spell	7
write	student	paper	spell	2

Table 2: Examples from the disambiguation task developed by Grefenstette and Sadrzadeh (2011). Human judges give scores between 1 to 7, with higher scores implying higher semantic similarity. Verb *meet* with subject *system* and object *criterion* is judged similar to the landmark verb *satisfy* but not *visit*.

5.2 Baselines

We compare our model against multiple baselines for semantic compositionality:

1. Mitchell and Lapata's (2008) additive and element-wise multiplicative model as simplest baselines.

2. Grefenstette and Sadrzadeh's (2011) model based on the abstract categorical framework (Coecke et al., 2010). This model computes the outer product of the subject and object vector, the outer product of the verb vector with itself, and then the element-wise product of both results.

3. Erk and Padó's (2008) model, which adapts the word vectors based on context and is the most similar in terms of motivation to ours.

4. Van de Cruy et al. (2013) multi-way interaction model based on matrix factorization. This achieves the best result for this task to date.

A detailed explanation of these models will be provided in Section 7. For the underlying word representations, we experiment with sparse 2000-dim SDS and dense 50-dim NLM. These are provided by Blacoe and Lapata (2012)[5] and trained on the British National Corpus (BNC). We are interested in knowing how sensitive each model is to the underlying word representation. In general, this is a challenging task: the upper-bound of $\rho = 0.62$ is the inter-annotator agreement.

[5] http://homepages.inf.ed.ac.uk/s1066731/index.php?page=resources

5.3 Implementation details

In terms of implementation detail, our model and our re-implementation of Erk and Pado's model make use of the ukWaC corpus (Baroni et al., 2009).[6] This corpus is a two billion word corpus automatically harvested from the web and parsed by the MaltParser (Nivre et al., 2006). We use ukWaC corpus to collect $W(\textit{VerbOf}, w_o)$ and $W(\textit{ObjOf}, w_v)$ for prototype projections. We also extract about 5000 verb-object pairs that relevant for testdata from this corpus to train our neural network learning algorithm. In our co-compositional model, the contribution ratio of SVD is set to 80% (i.e. automatically fixing k in SVD to include 80% of the top singular values). We set the number of prototype vectors to be $m = 20$, where $W(\textit{VerbOf}, w_o)$ is filtered with high frequency words and $W(\textit{ObjOf}, w_v)$ is filtered with both high frequency and high similarity words. In our model, we output the scores for SVO triple sentence dataset as (subject=w_s, verb=w_v, object=w_o, f = Addition/Multiplication):

$$\mathbf{cocomp}(w_s, w_v, w_o) = f(\mathbf{d}(w_s), \mathbf{cocomp}(w_v, w_o)) \quad (11)$$

6 Results and Discussion

6.1 Main Results: The Correlation

Table 3 shows the correlation scores of various models. Our observations are as follows:

1. The best reported result for this task (Van de Cruys et al., 2013) is $\rho = 0.37$. Our model (with NLM as word representation and f=Addition as operator) achieves $\rho = 0.44$, outperforming it by a large margin. To the best of our knowledge, this is now state-of-the-art result for this task.

2. Our model is not very sensitive to the underlying word representation. With f=Addition, we have $\rho = 0.41$ for SDS vs $\rho = 0.44$ for NLM. With f=Multiply, we have $\rho = 0.37$ for SDS vs. $\rho = 0.35$ for NLM. This implies that the prototype projection is robust to the underlying word representation, which is a desired characteristic of compositional models.

[6] http://wacky.sslmit.unibo.it/doku.php?id=corpora

Model	ρ
Grefenstette and Sadrzadeh (2011)	0.21
Add (SDS)	0.31
Add (NLM)	0.31
Multiply (SDS)	0.35
Multiply (NLM)	0.30
Van de Cruys et al. (2013)	0.37
Erk and Padó (SDS)	0.39
Erk and Padó (NLM)	0.03
Co-Comp with f=Add (SDS)	0.41
Co-Comp with f=Add (NLM)	**0.44**
Co-Comp with f=Multiply (SDS)	0.37
Co-Comp with f=Multiply (NLM)	0.35
Upper bound	0.62

Table 3: Results of the different compositionality models on the similarity task. The number of prototype words $m = 20$ in all our models. Our model (f=Addition and NLM) achieves the new state-of-the-art performance for this task ($\rho = 0.44$).

3. The contextual model of Erk and Padó (SDS) also performed relatively well ($\rho = 0.39$), in fact outperforming the Van de Cruy et al. (2013) result as well. This means that the general idea of adapting word representations based on context is a very powerful one. However, Erk and Padó's model using the NLM representation is extremely poor ($\rho = 0.03$). The reason is that it uses a product operation under-the-hood to adapt the vectors, which inherently assumes a sparse representation. In this sense, our projection approach is more robust.

The state-of-the-art result for our model in Table 3 does not yet make use of the training algorithm described in Section 4. It is simply implementing the co-compositionality idea using prototype projections (Section 3.2). Next in Section 6.2 we will show additional gains using unsupervised learning.

6.2 Improvements from unsupervised learning

In this experiment, we examine how much gain is possible by re-training the word representation of verbs using the unsupervised algorithm described in Section 4. We focus on the additive model of Compositional NLM, both basic and prototype projection. The initial word representation is from

model	original representation	re-trained
C-NLM	0.31	0.38
CoC-NLM	0.44	**0.47**

Table 4: Results of re-training the word representation for C-NLM and CoC-NLM. Learning rate $\alpha = 0.01$, regularization $\lambda = 10^{-4}$ and iteration = 20. One iteration is one run through the dataset of 5000 verb-object pairs which we made from the ukWaC corpus.

NLM. Table 4 shows the gains in correlation score.

This result shows that our learning model successfully captures good representation within co-compositionality of additive model. In contrast to other previous compositional models, our model does not require estimating a large number of parameters for computation of compositional vectors and word representation itself is more suitable for it. Furthermore, learning is very fast, taking about 10 minutes for C-NLM on a standard machine with Intel Core i7 2.93Ghz CPU and 8GB of RAM.

6.3 The number of prototype words

The number of prototype words (m in Figure 1) we use to generate the prototype space is one hyper-parameter that our model has. Here, we analyze the effect of the choice of m. Figure 5 shows the relation of m and the performance of co-compositional model with prototype projections using either SDS or NLM representations. In general, both NLM and SDS show relatively smooth and flat curves across m, indicating the relative robustness of the approach. Nevertheless, results do degrade for large m, due to increase in noise from non-prototype words. Further, it does appear that NLM has a slower drop in correlation with increasing m compared with SDS. This suggests that NLM is more robust, which is possibly attributable to the dense and low-dimensional distributed features.

6.4 Variations in model configuration

We have presented a compositional model of the form $\mathbf{d}(w_s) + \mathbf{cocomp}(w_v, w_o)$, where prototype projections are performed on both w_v and w_o and w_s is composed as is without projection. In general, we have the freedom to choose what to project and what not to project under this co-compositional framework. Here in Table 5 we show the results of

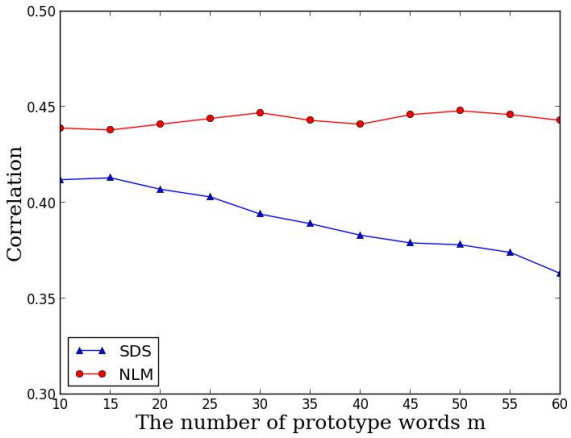


Figure 5: The relation between the number of prototype words and correlation of SDS or NLM. In general, NLM has higher correlation than SDS and is more robust across the m.

Subj	Verb	Obj	NLM ρ	SDS ρ
prpj	prpj	prpj	0.39	0.37
+	prpj	prpj	0.44	0.41
	prpj	prpj	0.45	0.41
+	prpj	+	0.43	0.38
	prpj	+	0.43	0.38
+	+	+	0.31	0.31

Table 5: Variants of the full co-compositional model, based on how subject, verb, and object vector representations are included. prpj indicates that prototype projection is used. + indicates that the vector is added without projection first. Blank indicates that the vector is not used in the final compositional score.

these variants, using f =Addition and SDS/NLM representations without re-training. We note that our positive results mainly come from the verb projections. Subject information actually does not help. We believe this best configuration is task-dependent; in this test collection, the subjects appear to have little contribution to the landmark verb.

7 Related work

In recent years, several sophisticated vector space models have been proposed for computing compositional semantics. Mitchell and Lapata (2010), Erk (2012) and Baroni et al. (2013) are recommended survey papers.

One of the first approaches is the vector addition/multiplication idea of Mitchell and Lapata (2008). The appeal of this kind of simple approach is its intuitive geometric interpretation and its robustness to various datasets. However, it may not be sufficiently expressive to represent the various factors involved in compositional semantics, such as syntax and context. To this end, Baroni and Zamparelli (2010) present a compositional model for adjectives and nouns. In their model, an adjective is a matrix operator that modifies the noun vector into an adjective-noun vector. Zanzotto et al. (2010) and Guevara (2010) also proposed linear transformation models for composition and address the issue of estimating large matrices with least squares or regression techniques. Socher et al. (2012) extend this linear transformation approach with the more powerful model of Matrix-Vector Recursive Neural Networks (MV-RNN). Each node in a parse tree is assigned both a vector and a matrix. The vector captures the actual meaning of the word itself, while the matrix is modeled as a operator that modify the meaning of neighboring words and phrases. This model captures semantic change phenomenon like *not bad* is similar to *good* due to a composition of the *bad* vector with a meaning-flipping *not* matrix. But this MV-RNN also need to optimize all matrices of words from initial value (identity plus a small amount of Gaussian noise) with supervised dataset like movie reviews. Our prototype projection model is similar to these models as a matrix-vector operation, except that the matrix is not learned and computed from prototype words. In future work, we can imagine integrating the two models, using these prototype projection matrices as initial values for MV-RNN training (Socher et al., 2012).

Another approach is exemplified by Coecke et al. (2010). In their mathematical framework unifying categorical logic and vector space models, the sentence vector is modeled as a function of the Kronecker product of its word vectors. Grefenstette and Sadrzadeh (2011) implement this based on unsupervised learning of matrices for relational words and apply them to the vectors of their arguments. Their idea is that words with relational types, such as verbs, adjectives, and adverbs are matrices that act as a filter on their arguments. They also developed a new semantic similarity task based on transitive verbs, which is the dataset we used here. The previous state-of-the-art result for this task comes from the model of Van de Cruys et al. (2013). They model compositionality as a multi-way interaction between latent factors, which are automatically constructed from corpus data via matrix factorization.

Composition Operator	Parameter
Add: $w_1 u + w_2 v$	$w_1, w_2 \in \mathbb{R}$
Multiply: $u^{w_1} \odot v^{w_2}$	$w_1, w_2 \in \mathbb{R}$
FullAdd: $W_1 u + W_2 v$	$W_1, W_2 \in \mathbb{R}^{n \times n}$
LexFunc: $A_u v$	$A_u \in \mathbb{R}^{n \times n}$
FullLex: $\sigma([W_1 A_u v, W_2 A_v u])$	$A_u, A_v \in \mathbb{R}^{n \times n}$ $W_1, W_2 \in \mathbb{R}^{n \times n}$
Ours (Add): $P_{(R,v)} u + P_{(R,u)} v$	SVD's (m, k)
Ours (Mult): $P_{(R,v)} u \odot P_{(R,u)} v$	SVD's (m, k)

Table 6: Comparison of composition operators that combine two word vector representations, $u, v \in \mathbb{R}^n$ and their learning parameters. Our model only needs two hyper-parameters: the number of prototype words m and dimensional reduction k in SVD

Comprehensive evaluation of various existing models are reported in (Blacoe and Lapata, 2012; Dinu et al., 2013). Blacoe and Lapata (2012) highlight the importance of jointly examining word representations and compositionality operators. However, two out of three composition methods they evaluate are parameter-free, so that they can side-step the issue of parameter estimation. Dinu et al. (2013) describe the relation between word vector and compositionality in more detail with free parameters. Table 6 summarizes some ways to compose the meaning of two word vectors (u, v), following (Dinu et al., 2013). These range from simple operators (e.g. Add and Multiply) to expressive models with many free parameters (e.g. LexFunc, FullLex). Many of these models need to optimize $n \times n$ parameters, which may be large. On the other hand, our model only needs two hyper-parameters: the number of prototype words m and dimensional reduction k in SVD (Table 6). Furthermore, our model performance with neural language model word embeddings is robust to variations in m.

Most closely related to our work is the work by Erk and Padó (2008; 2009) and Thater et al. (2010; 2011), which falls under the research theme of computing *word meaning in context*. Both methods are characterized by the use of selectional prefer-

ence information for subjects, verbs, and objects in context; our prototype word vectors are essentially equivalent to this idea. The main difference is in how we modify the target word representation $\mathbf{v}$ using this information: whereas we project $\mathbf{v}$ onto a latent subspace formed by collection of prototype vectors, Erk and Padó (2008; 2009) and Thater et al. (2010; 2011) use the prototype vectors to directly modify the elements of $\mathbf{v}$, i.e. by element-wise product with the centroid prototype vector. Intuitively, both our method and theirs essentially delete part of a word vector representation to adapt the meaning in context. We believe the projection is more robust to the underlying word representation (and this is shown in the results for SDS vs. NLM representations), but we note that we may be able to borrow some of more sophisticated ways to find prototype vectors from Erk and Padó (2008; 2009) and Thater et al. (2010; 2011).

8 Conclusion and Future Work

We began this work by asking how it is possible to handle polysemy issues in compositional semantics, especially when adopting distributional semantics methods that construct only one representation per word type. After all, the different senses of the same word are all conflated into a single vector representation. We found our inspiration in Generative Lexicon Theory (Pustejovsky, 1995), where ambiguity is resolved due to *co-compositionality* of the words in the sentence, i.e., the meaning of an ambiguous verb is generated by the properties the object it takes, and vice versa. We implement this idea in a novel neural network model using *prototype projections*. The advantages of this model is that it is robust to the underlying word representation used and that it enables an effective joint learning of word representations. The model achieves the current state-of-the-art performance ($\rho = 0.47$) on the semantic similarity task of transitive verbs (Grefenstette and Sadrzadeh, 2011).

Directions for future research include:

- Experiments on other semantics tasks, such as paraphrase detection, word sense induction, and word meaning in context.
- Extension to more holistic sentence-level composition using a matrix-vector recursive framework like (Socher et al., 2012).
- Explore further the potential synergy between Distributional Semantics and the Generative Lexicon.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number 24800041, JSPS KAKENHI 2430057 and Microsoft Research CORE Project. We would like to thank Hiroyuki Shindo and anonymous reviewers for their helpful comments.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2013. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technologies*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *CoRR*, abs/1003.4394.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch.

Journal of Machine Learning Research, 12:2493–2537.

Georgiana Dinu, Nghia The Pham, and Marco Barori. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

G Frege. 1892. Über sinn und bedeutung. In *Zeitschfrift für Philosophie und philosophische Kritik, 100*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the Workshop on GEometrical Models of Natural Language Semantics*.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge, MA.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Asian Federation of Natural Language Processing (IJCNLP)*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Studying the recursive behaviour of adjectival modification with compositional distributional semantics

Eva Maria Vecchi and **Roberto Zamparelli** and **Marco Baroni**
Center for Mind/Brain Sciences (University of Trento, Italy)
(evamaria.vecchi|roberto.zamparelli|marco.baroni)@unitn.it

Abstract

In this study, we use compositional distributional semantic methods to investigate restrictions in adjective ordering. Specifically, we focus on properties distinguishing Adjective-Adjective-Noun phrases in which there is flexibility in the adjective ordering from those bound to a rigid order. We explore a number of measures extracted from the distributional representation of AAN phrases which may indicate a word order restriction. We find that we are able to distinguish the relevant classes and the correct order based primarily on the degree of modification of the adjectives. Our results offer fresh insight into the semantic properties that determine adjective ordering, building a bridge between syntax and distributional semantics.

1 Introduction

A prominent approach for representing the meaning of a word in Natural Language Processing (NLP) is to treat it as a numerical vector that codes the pattern of co-occurrence of that word with other expressions in a large corpus of language (Sahlgren, 2006; Turney and Pantel, 2010). This approach to semantics (sometimes called *distributional semantics*) scales well to large lexicons and does not require words to be manually disambiguated (Schütze, 1997). Until recently, however, this method had been almost exclusively limited to the level of single content words (nouns, adjectives, verbs), and had not directly addressed the problem of *compositionality* (Frege, 1892; Montague, 1970; Partee, 2004), the crucial property of natural language which allows speakers to derive the meaning of a complex linguistic constituent from the meaning of its immediate syntactic subconstituents.

Several recent proposals have strived to extend distributional semantics with a component that also generates vectors for complex linguistic constituents, using compositional operations in the vector space (Baroni and Zamparelli, 2010; Guevara, 2010; Mitchell and Lapata, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012). All of these approaches construct distributional representations for novel phrases starting from the corpus-derived vectors for their lexical constituents and exploiting the geometric quality of the representation. Such methods are able to capture complex semantic information of adjective-noun (AN) phrases, such as characterizing modification (Boleda et al., 2012; Boleda et al., 2013), and can detect semantic deviance in novel phrases (Vecchi et al., 2011). Furthermore, these methods are naturally recursive: they can derive a representation not only for, e.g., *red car*, but also for *new red car*, *fast new red car*, etc. This aspect is appealing since trying to extract meaningful representations for all recursive phrases directly from a corpus will result in a problem of sparsity, since most large phrases will never occur in any finite sample.

Once we start seriously looking into recursive modification, however, the issue of modifier ordering restrictions naturally arises. Such restrictions have often been discussed in the theoretical linguistic literature (Sproat and Shih, 1990; Crisma, 1991; Scott, 2002), and have become one of the key in-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 141–151,
Seattle, Washington, USA, 18-21 October 2013.

gredients of the 'cartographic' approach to syntax (Cinque, 2002). In this paradigm, the ordering is derived by assigning semantically different classes of modifiers to the specifiers of distinct functional projections, whose sequence is hard-wired. While it is accepted that in different languages movement can lead to a principled rearrangement of the linear order of the modifiers (Cinque, 2010; Steddy and Samek-Lodovici, 2011), one key assumption of the cartographic literature is that exactly one intonationally unmarked order for stacked adjectives should be possible in languages like English. The possibility of alternative orders, when discussed at all, is attributed to the presence of idioms (*high American building*, but *American high officer*), to asyndetic conjunctive meanings (e.g. *new creative idea* parsed as *[new & creative] idea*, rather than *[new [creative idea]]*), or to semantic category ambiguity for any adjective which appears in different orders (see Cinque (2004) for discussion).

In this study, we show that the existence of both rigid and flexible order cases is robustly attested at least for adjectival modification, and that flexible ordering is unlikely to reduce to idioms, coordination or ambiguity. Moreover, we show that at least for some recursively constructed adjective-adjective-noun phrases (AANs) we can extract meaningful representations from the corpus, approximating them reasonably well by means of compositional distributional semantic models, and that the semantic information contained in these models characterizes which AA will have rigid order (as with *rapid social change* vs. **social rapid change*), or flexible order (e.g. *total estimated population* vs. *estimated total population*). In the former case, we find that the same distributional semantic cues discriminate between correct and wrong orders.

To achieve these goals, we consider various properties of the distributional representation of AANs (both corpus-extracted and compositionally-derived), and explore their correlation with restrictions in adjective ordering. We conclude that measures that quantify the degree to which the modifiers have an impact on the distributional meaning of the AAN can be good predictors of ordering restrictions in AANs.

2 Materials and methods

2.1 Semantic space

Our initial step was to construct a *semantic space* for our experiments, consisting of a matrix where each row represents the meaning of an adjective, noun, AN or AAN as a distributional vector, each column a semantic dimension of meaning. We first introduce the source corpus, then the vocabulary of words and phrases that we represent in the space, and finally the procedure adopted to build the vectors representing the vocabulary items from corpus statistics, and obtain the semantic space matrix. We work here with a traditional, window-based semantic space, since our focus is on the effect of different composition methods given a common semantic space. In addition, Blacoe and Lapata (2012) found that a vanilla space of this sort performed best in their composition experiments, when compared to a syntax-aware space and to neural language model vectors such as those used for composition by Socher et al. (2011).

Source corpus We use as our source corpus the concatenation of the Web-derived ukWaC corpus, a mid-2009 dump of the English Wikipedia and the British National Corpus[1]. The corpus has been tokenized, POS-tagged and lemmatized with the TreeTagger (Schmid, 1995), and it contains about 2.8 billion tokens. We extract all statistics at the lemma level, meaning that we consider only the canonical form of each word ignoring inflectional information, such as pluralization and verb inflection.

Semantic space vocabulary The words/phrases in the semantic space must of course include the items that we need for our experiments (adjectives, nouns, ANs and AANs used for model training, as input to composition and for evaluation). Therefore, we first populate our semantic space with a core vocabulary containing the 8K most frequent nouns and the 4K most frequent adjectives from the corpus.

The ANs included in the semantic space are composed of adjectives with very high frequency in the corpus so that they are generally able to combine with many classes of nouns. They are composed of the 700 most frequent adjectives and 4K most frequent nouns in the corpus, which were manually

[1] http://wacky.sslmit.unibo.it, http://en.wikipedia.org, http://www.natcorp.ox.ac.uk

controlled for problematic cases – excluding adjectives such as *above*, *less*, or *very*, and nouns such as *cant*, *mph*, or *yours* – often due to tagging errors. We generated the set of ANs by crossing the filtered 663 adjectives and 3,910 nouns. We include those ANs that occur at least 100 times in the corpus in our vocabulary, which amounted to a total of 128K ANs.

Finally, we created a set of AAN phrases composed of the adjectives and nouns used to generate the ANs. Additional preprocessing of the generated A_xA_yNs includes: (i) control that both A_xN and A_yN are attested in the corpus; (ii) discard any A_xA_yN in which A_xN or A_yN are among the top 200 most frequent ANs in the source corpus (as in this case, order will be affected by the fact that such phrases are almost certainly highly lexicalized); and (iii) discard AANs seen as part of a conjunction in the source corpus (i.e., where the two adjectives appear separated by comma, *and*, or *or*; this addresses the objection that a flexible order AAN might be a hidden A(&)A conjunction: we would expect that such a conjunction should also appear overtly elsewhere). The set of AANs thus generated is then divided into two types of adjective ordering:

1. **Flexible Order** (FO): phrases where *both* orders, A_xA_yN and A_yA_xN, are attested ($f>10$ in both orders).

2. **Rigid Order** (RO): phrases with *one* order, A_xA_yN, attested ($20<f<200$)[2] and A_yA_xN unattested.

All AANs that did not meet either condition were excluded from our semantic space vocabulary. The preserved set resulted in 1,438 AANs: 621 flexible order and 817 rigid order. Note that there are almost as many flexible as rigid order cases; this speaks against the idea that free order is a marginal phenomenon, due to occasional ambiguities that reassign the adjective to a different semantic class. The existence of freely ordered stacked adjectives is a robust phenomenon, which needs to be addressed.

[2]The upper threshold was included as an additional filter against potential multiword expressions. Of course, the boundary between phrases that are at least partially compositional and those that are fully lexicalized is not sharp, and we leave it to further work to explore the interplay between the semantic factors we study here and patterns of lexicalization.

Model	ρ	*M&L*
CORP	0.41	0.43
W.ADD	0.41	0.44
F.ADD	0.40	–
MULT	0.33	0.46
LFM	0.40	–

Table 1: Correlation scores (Spearman's ρ, all significant at $p<0.001$) between cosines of corpus-extracted or model-generated AN vectors and phrase similarity ratings collected in Mitchell and Lapata (2010), as well as best reported results from Mitchell & Lapata (M&L).

Semantic vector construction For each of the items in our vocabulary, we first build 10K-dimensional vectors by recording the item's sentence-internal co-occurrence with the top 10K most frequent content lemmas (nouns, adjectives, verbs or adverbs) in the corpus. We built a rank of these co-occurrence counts, and excluded as stop words from the dimensions any element of any POS whose rank was from 0 to 300. The raw co-occurrence counts were then transformed into (positive) Pointwise Mutual Information (pPMI) scores (Church and Hanks, 1990). Next, we reduce the full co-occurrence matrix to 300 dimensions applying the Non-negative Matrix Factorization (NMF) operation (Lin, 2007). We did not tune the semantic vector construction parameters, since we found them to work best in a number of independent earlier experiments.

Corpus-extracted vectors (*corp*) were computed for the ANs and for the flexible order and attested rigid order AANs, and then mapped onto the 300-dimension NMF-reduced semantic space. As a sanity check, the first row of Table 1 reports the correlation between the AN phrase similarity ratings collected in Mitchell and Lapata (2010) and the cosines of corpus-extracted vectors in our space, for the same ANs. For the AAN vectors, which are sparser, we used human judgements to build a reliable subset to serve as our gold standard, as detailed in Section 2.4.

2.2 Composition models

We focus on four composition functions proposed in recent literature with high performance in a number of semantic tasks. We first consider methods proposed by Mitchell and Lapata (2010) in

which the model-generated vectors are simply obtained through component-wise operations on the constituent vectors. Given input vectors $\vec{u}$ and $\vec{v}$, the multiplicative model (MULT) computes a composed vector by component-wise multiplication ($\odot$) of the constituent vectors, where the i-th component of the composed vector is given by $p_i = u_i v_i$.[3] Given an $A_x A_y N$ phrase, this model extends naturally to the recursive setting of this experiment, as seen in Equation (1).

$$\vec{p} = \vec{a}_x \odot \vec{a}_y \odot \vec{n} \quad (1)$$

This composition method is order-insensitive, the formula above corresponding to the representation of both $A_x A_y N$ and $A_y A_x N$.

In the weighted additive model (W.ADD), we obtain the composed vector as a weighted sum of the two component vectors: $\vec{p} = \alpha\vec{u} + \beta\vec{v}$, where α and β are scalars. Again, we can easily apply this function recursively, as in Equation (2).

$$\vec{p} = \alpha\vec{a}_x + \beta(\alpha\vec{a}_y + \beta\vec{n}) = \alpha\vec{a}_x + \alpha\beta\vec{a}_y + \beta^2\vec{n} \quad (2)$$

We also consider the full extension of the additive model (F.ADD), presented in Guevara (2010) and Zanzotto et al. (2010), such that the component vectors are pre-multiplied by weight matrices before being added: $\vec{p} = \mathbf{W}_1\vec{u} + \mathbf{W}_2\vec{v}$. Similarly to the W.ADD model, Equation (3) describes how we apply this function recursively.

$$\begin{aligned}\vec{p} &= \mathbf{W}_1\vec{a}_x + \mathbf{W}_2(\mathbf{W}_1\vec{a}_y + \mathbf{W}_2\vec{n}) \quad (3)\\ &= \mathbf{W}_1\vec{a}_x + \mathbf{W}_2\mathbf{W}_1\vec{a}_y + \mathbf{W}_2^2\vec{n}\end{aligned}$$

Finally, we consider the lexical function model (LFM), first introduced in Baroni and Zamparelli (2010), in which attributive adjectives are treated as functions from noun meanings to noun meanings. This is a standard approach in Montague semantics (Thomason, 1974), except noun meanings here are distributional vectors, not denotations, and adjectives are (linear) functions learned from a large corpus. In this model, predicted vectors are generated by multiplying a function matrix $\mathbf{U}$ with a component vector: $\vec{p} = \mathbf{U}\vec{v}$. Given a weight matrix, $\mathbf{A}$, for each adjective in the phrase, we apply the functions in sequence recursively as shown in Equation (4).

$$\vec{p} = \mathbf{A}_x(\mathbf{A}_y\vec{n}) \quad (4)$$

Composition model estimation Parameters for W.ADD, F.ADD and LFM were estimated following the strategy proposed by Guevara (2010) and Baroni and Zamparelli (2010), recently extended to all composition models by Dinu et al. (2013b). Specifically, we learn parameter values that optimize the mapping from the noun to the AN as seen in examples of corpus-extracted N-AN vector pairs, using least-squares methods. All parameter estimations and phrase compositions were implemented using the DISSECT toolkit[4] (Dinu et al., 2013a), with a training set of 74,767 corpus-extracted N-AN vector pairs, ranging from 100 to over 1K items across the 663 adjectives. Importantly, while below we report experimental results on capturing various properties of recursive AAN constructions, no AAN was seen during training, which was based entirely on mapping from N to AN. Table 1 reports the results attained by our model implementations on the Mitchell and Lapata AN similarity data set.

2.3 Measures of adjective ordering

Our general goal is to determine which linguistically-motivated factors distinguish the two types of adjective ordering. We hypothesize that in cases of flexible order, the two adjectives will have a similarly strong effect on the noun, thus transforming the meaning of the noun equivalently in the direction of both adjectives and component ANs. For example, in the phrase *creative new idea*, the *idea* is both *new* and *creative*, so we would expect a similar impact of modification by both adjectives.

On the other hand, we predict that in rigid order cases, one adjective, the one closer to the noun, will dominate the meaning of the phrase, distorting the meaning of the noun by a significant amount. For example, the phrase *different architectural style* intuitively describes an *architectural style* that is *dif-*

[3] We conjecture that the different performance of our multiplicative model and M&L's (cf. Table 1) is due to the fact that we use log-transformed pPMI scores, making their multiplicative model more akin to our additive approach.

[4] http://clic.cimec.unitn.it/composes/toolkit

ferent, rather than a *style* that is to the same extent *architectural* and *different*.

We consider a number of measures that could capture our intuitions and quantify this difference, exploring the distance relationship between the AAN vectors and each of the AAN subparts. First, we examine how the similarity of an AAN to its component adjectives affects the ordering, using the cosine between the A_xA_yN vector and each of the component A vectors as an expression of similarity (we abbreviate this as $cosA_x$ and $cosA_y$ for the first and second adjective, respectively).[5] Our hypothesis predicts that flexible order AANs should remain similarly close to both component As, while rigid order AANs should remain systematically closer to their A_y than to their A_x.

Next, we consider the similarity between the A_xA_yN vector and its component N vector ($cosN$). This measure is aimed at verifying if the degree to which the meaning of the head noun is distorted could be a property that distinguishes the two types of adjective ordering. Again, vectors for flexible order AANs should remain closer to their component nouns in the semantic space, while rigid order AANs should distort the meaning of the head noun more notably.

We also inspect how the similarity of the AAN to its component AN vectors affects the type of adjective ordering ($cosA_xN$ and $cosA_yN$). Considering the examples above, we predict that the flexible order AAN *creative new idea* will share many properties with both *creative idea* and *new idea*, as represented in our semantic space, while rigid order AANs, like *different architectural style*, should remain quite similar to the A_yN, i.e., *architectural style*, and relatively distant from the A_xN, i.e., *different style*.

Finally, we consider a measure that does not exploit distributional semantic representations, namely the difference in PMI between A_xN and A_yN (ΔPMI). Based on our hypothesis described for the other measures, we expect the association in the corpus of A_yN to be much greater than A_xN for rigid order AANs, resulting in a large negative ΔPMI values. While flexible order AANs should have similar association strengths for both A_xN and A_yN, thus we expect ΔPMI to be closer to 0 than for rigid order AANs.

[5] In the case of LFM, we compare the similarity of the AAN with the AN centroids for each adjective, since the model does not make use of A vectors (Baroni and Zamparelli, 2010).

2.4 Gold standard

To our knowledge, this is the first study to use distributional representations of recursive modification; therefore we must first determine if the composed AAN vector representations are semantically coherent objects. Thus, for vector analysis, a *gold standard* of 320 corpus-extracted AAN vectors were selected and their quality was established by inspecting their nearest neighbors. In order to create the gold standard, we ran a crowdsourcing experiment on CrowdFlower[6] (Callison-Burch and Dredze, 2010; Munro et al., 2010), as follows.

First, we gathered a randomly selected set of 600 corpus-extracted AANs, containing 300 flexible order and 300 attested rigid order AANs. We then extracted the top 3 nearest neighbors to the corpus-extracted AAN vectors as represented in the semantic space[7]. Each AAN was then presented with each of the nearest neighbors, and participants were asked to judge "how strongly related are the two phrases?" on a scale of 1-7. The rationale was that if we obtained a good distributional representation of the AAN, its nearest neighbors should be closely related words and phrases. Each pair was judged 10 times, and we calculated a *relatedness* score for the AAN by taking the average of the 30 judgments (10 for each of the three neighbors).

The final set for the gold standard contains the 320 AANs (152 flexible order and 168 attested rigid order) which had a relatedness score over the median-split (3.9). Table 2 shows examples of gold standard AANs and their nearest neighbors. As these example indicate, the gold standard AANs reside in semantic neighborhoods that are populated by intuitively strongly related expressions, which makes them a sensible target for the compositional models to approximate.

We also find that the neighbors for the AANs represent an interesting variety of types of semantic

[6] http://www.crowdflower.com

[7] The top 3 neighbors included adjectives, nouns, ANs and AANs. The preference for ANs and AANs, as seen in Table 2, is likely a result of the dominance of those elements in the semantic space (c.f. Section 2.1).

medieval old town	*contemp. political issue*
fascinating town	cultural topic
impressive cathedral	contemporary debate
medieval street	contemporary politics
rural poor people	*British naval power*
poor rural people	naval war
rural infrastructure	British navy
rural people	naval power
friendly helpful staff	*last live performance*
near hotel	final gig
helpful staff	live dvd
quick service	live release
creative new idea	*rapid social change*
innovative effort	social conflict
creative design	social transition
dynamic part	cultural consequence
national daily newspaper	*new regional government*
national newspaper	regional government
major newspaper	local reform
daily newspaper	regional council
daily national newspaper	*fresh organic vegetable*
national daily newspaper	organic vegetable
well-known journalist	organic fruit
weekly column	organic product

Table 2: Examples of the nearest neighbors of the gold standard, both flexible order (left column) and rigid order (right column) AANs.

similarity. For example, the nearest neighbors to the corpus-extracted vectors for *medieval old town* and *rapid social change* include phrases which describe quite complex associations, cf. Table 2. In addition, we find that the nearest neighbors for flexible order AAN vectors are not necessarily the same for both adjective orders, as seen in the difference in neighbors of *national daily newspaper* and *daily national newspaper*. We can expect that the change in order, when acceptable and frequent, does not necessarily yield synonymous phrases, and that corpus-extracted vector representations capture subtle differences in meaning.

3 Results

3.1 Quality of model-generated AAN vectors

Our nearest neighbor analysis suggests that the corpus-extracted AAN vectors in the gold standard are meaningful, semantically coherent objects. We can thus assess the quality of AANs recursively generated by composition models by how closely they approximate these vectors. We find that the performances of most composition models in approximating the vectors for the gold AANs is quite satisfactory (cf. Table 3). To put this evaluation into perspective, note that 99% of the simulated distribution of pairwise cosines of corpus-extracted AANs is below the mean cosine of the worst-performing model (MULT), that is, a cosine of 0.424 is very significantly above what is expected by chance for two random corpus-extracted AAN vectors. Also, observe that the two more parameter-rich models are better than W.ADD, and that LFM also significantly outperforms F.ADD.

	Gold	*FO*	*RO*
W.ADD	0.565	0.572	0.558
F.ADD	0.618	0.622	0.614
MULT	0.424	0.468	0.384
LFM	0.655	0.675	0.637

Table 3: Mean cosine similarities between the corpus-extracted and model-generated gold AAN vectors. All pairwise differences between models are significant according to Bonferroni-corrected paired t-tests ($p<0.001$). For MULT and LFM, the difference between mean flexible order (FO) and rigid order (RO) cosines is also significant.

Further, the results show that the models are able to approximate flexible order AAN vectors better than rigid order AANs, significantly so for LFM and MULT. This result is quite interesting because it suggests that flexible order AANs express a more literal (or intersective) modification by both adjectives, which is what we would expect to be better captured by compositional models. Clearly, a more complex modification process is occurring in the case of rigid order AANs, as we predicted to be the case.

3.2 Distinguishing flexible vs. rigid order

In the results reported below, we test how both our baseline ΔPMI measure and the distance from the AAN and its component parts changes depending on the type of adjective ordering to which the AAN belongs. From this point forward, we only use gold standard items, where we are sure of the quality of the corpus-extracted vectors. The first block of Table 4 reports the t-normalized difference between flexible order and rigid order mean cosines for the corpus-extracted vectors.

	Measure	*t*	*sig.*	
CORP	cosA_x	2.478		
	cosA_y	-4.348	*	RO>FO
	cosN	4.656	*	FO>RO
	cosA_xN	5.913	*	FO>RO
	cosA_yN	1.970		
W.ADD	cosA_x	4.805	*	FO>RO
	cosA_y	-1.109		
	cosN	1.140		
	cosA_xN	1.059		
	cosA_yN	0.584		
F.ADD	cosA_x	2.050		
	cosA_y	-1.451		
	cosN	4.493	*	FO>RO
	cosA_xN	-0.445		
	cosA_yN	2.300		
MULT	cosA_x	3.830	*	FO>RO
	cosA_y	-0.503		
	cosN	5.090	*	FO>RO
	cosA_xN	4.435	*	FO>RO
	cosA_yN	3.900	*	FO>RO
LFM	cosA_x	-1.649		
	cosA_y	-1.272		
	cosN	5.539	*	FO>RO
	cosA_xN	3.336	*	FO>RO
	cosA_yN	4.215	*	FO>RO
ΔPMI		8.701	*	FO>RO

Table 4: **Flexible vs. Rigid Order AANs.** t-normalized differences between flexible order (FO) and rigid order (FO) mean cosines (or mean ΔPMI values) for corpus-extracted and model-generated vectors. For significant differences ($p<0.05$ after Bonferroni correction), the last column reports whether mean cosine (or ΔPMI) is larger for flexible order (FO) or rigid order (RO) class.

These results show, in accordance with our considerations in Section 2.3 above: (i) flexible order A_xA_yNs are closer to A_xN and the component N than rigid order A_xA_yNs, and (ii) rigid order A_xA_yNs are closer to their A_y (flexible order AANs are also closer to A_x but the effect does not reach significance).[8] The results imply that the degree of modification of the A_y on the noun is a significant indicator of the type of ordering present.

[8]As an aside, the fact that mean cosines are significantly larger for the flexible order class in two cases but for the rigid order class in another addresses the concern, raised by a reviewer, that the words and phrases in one of the two classes might systematically inhabit denser regions of the space than those of the other class, thus distorting results based on comparing mean cosines.

In particular, rigid order A_xA_yNs are heavily modified by A_y, distorting the meaning of the head noun in the direction of the closest adjective quite drastically, and only undergoing a slight modification when the A_x is added. In other words, in rigid order phrases, for example *rapid social change*, the A_yN expresses a single concept (probably a "kind", in the terminology of formal semantics), strongly related to *social*, *social change*, which is then modified by the A_x. Thus, the *change* is not both *social* and *rapid*, rather, the *social change* is *rapid*. On the other hand, flexible order AANs maintain the semantic value of the head noun while being modified only slightly by both adjectives, almost equivalently. For example, in the phrase *friendly helpful staff*, one is saying that the *staff* is both *friendly* and *helpful*. Most importantly, the corpus-extracted distributional representations are able to model this phenomenon inherently and can significantly distinguish the two adjective orders.

The results of the composition models (cf. Table 4) show that for all models at least some properties do distinguish flexible and rigid order AANs, although only MULT and LFM capture the two properties that show the largest effect for the corpus-extracted vectors, namely the asymmetry in similarity to the noun and the A_xN (flexible order AANs being more similar to both).

It is worth remarking that MULT approximated the patterns observed in the corpus vectors quite well, despite producing order-insensitive representations of recursive structures. For flexible order AANs, order is indeed only slightly affecting the meaning, so it stands to reason that MULT has no problems modeling this class. For rigid order AANs, where we consider here the attested-order only, evidently the order-insensitive MULT representation is sufficient to capture their relations to their constituents.

Finally, we see that the ΔPMI measure is the best at distinguishing between the two classes of AAN ordering. This confirms our hypothesis that a lot has to do with how integrated A_y and N are. While it is somewhat disappointing that ΔPMI outperforms all distributional semantic cues, note that this measure conflates semantic and lexical factors, as the high PMI of A_yN in at least some rigid order AANs might be also a cue of the fact that the latter bigram is a lexicalized phrase (as discussed in footnote 2, it

is unlikely that our filtering strategies sifted out all multiword expressions). Moreover, ΔPMI does not produce a semantic representation of the phrase (see how composed distributional vectors approximate of high quality AAN vectors in Table 3). Finally, this measure will not scale up to cases where the ANs are not attested, whereas measures based on composition only need corpus-harvested representations of adjectives and nouns.

3.3 Properties of the correct adjective order

Having shown that flexible order and rigid order AANs are significantly distinguished by various properties, we proceed now to test whether those same properties also allow us to distinguish between correct (corpus-attested) and wrong (unattested) adjective ordering in rigid AANs (recall that we are working with cases where the attested-order occurs more than 20 times in the corpus, and both adjectives modify the nouns at least 10 times, so we are confident that there is a true asymmetry).

We expect that the fundamental property that distinguishes the orders is again found in the degree of modification of both component adjectives. We predict that the single concept created by the A_yN in attested-order rigid AANs, such as *legal status* in *formal legal status*, is an effect of the modification strength of the A_y on the head noun, and when seen in the incorrect ordering, i.e., $^?$*legal formal status*, the strong modification of *legal* will still dominate the meaning of the AAN. Composition models should be able to capture this effect based on the distance from both the component adjectives and ANs.

Clearly, we cannot run these analyses on corpus-extracted vectors since the unattested order, by definition, is not seen in our corpus, and therefore we cannot collect co-occurrence statistics for the AAN phrase. Thus, we test our measures of adjective ordering on the model-generated AAN vectors, for all gold rigid order AANs in both orders.

We also consider the ΔPMI measure which was so effective in distinguishing flexible vs. rigid order AANs. We expect that the greater association with A_yN for attested-order AANs will again lead to large, negative differences in PMI scores, while the expectation that unattested-order AANs will be highly associated with their A_xN will correspond to large, positive differences in PMI.

	Measure	*t*	*sig.*	
W.ADD	cosA_x	-7.840	*	U>A
	cosA_y	7.924	*	A>U
	cosN	2.394		
	cosA_xN	-5.462	*	U>A
	cosA_yN	3.627	*	A>U
F.ADD	cosA_x	-8.418	*	U>A
	cosA_y	6.534	*	A>U
	cosN	-1.927		
	cosA_xN	-3.583	*	U>A
	cosA_yN	-2.185		
MULT	cosA_x	-5.100	*	U>A
	cosA_y	5.100	*	A>U
	cosN	0.000		
	cosA_xN	-0.598		
	cosA_yN	0.598		
LFM	cosA_x	-7.498	*	U>A
	cosA_y	7.227	*	A>U
	cosN	-2.172		
	cosA_xN	-5.792	*	U>A
	cosA_yN	0.774		
ΔPMI		-11.448	*	U>A

Table 5: **Attested- vs. unattested-order rigid order AANs.** t-normalized mean paired cosine (or ΔPMI) differences between attested (A) and unattested (U) AANs with their components. For significant differences (paired t-test $p<0.05$ after Bonferroni correction), last column reports whether cosines (or ΔPMI) are on average larger for A or U.

Across all composition models, we find that the distance between the model-generated AAN and its component adjectives, A_x and A_y, are significant indicators of attested vs. unattested adjective ordering (cf. Table 5). Specifically, we find that rigid order AANs in the correct order are closest to their A_y, while we can detect the unattested order when the rigid order AAN is closer to its A_x. This finding is quite interesting, since it shows that the order in which the composition functions are applied does not alter the fact that the modification of one adjective in rigid order AANs (the A_y in the case of attested-order rigid order AANs) is much stronger than the other. Unlike the measures that differentiated flexible and rigid order AANs, here we see that the distance from the component N is not an indicator of the correct adjective ordering (trivially so for MULT, where attested and unattested AANs are identical).

Next, we find that for W.ADD, F.ADD and LFM,

the distance from the component A_xN is a strong indicator of attested- vs. unattested-order rigid order AANs. Specifically, attested-order AANs are further from their A_xN than unattested-order AANs. This finding is in line with our predictions and follows the findings of the impact of the distance from the component adjectives.

ΔPMI, as seen in the ability to distinguish flexible vs. rigid order AANs, is the strongest indicator of correct vs wrong adjective ordering. This measure confirms that the association of one adjective (the A_y in attested-order AANs) with the head noun is indeed the most significant factor distinguishing these two classes. However, as we mentioned before, this measure has its limitations and is likely not to be entirely sufficient for future steps in modeling recursive modification.

4 Conclusion

While AN constructions have been extensively studied within the framework of compositional distributional semantics (Baroni and Zamparelli, 2010; Boleda et al., 2012; Boleda et al., 2013; Guevara, 2010; Mitchell and Lapata, 2010; Turney, 2012; Vecchi et al., 2011), for the first time, we extended the investigation to recursively built AAN phrases.

First, we showed that composition functions applied recursively can approximate corpus-extracted AAN vectors that we know to be of high semantic quality.

Next, we looked at some properties of the same high-quality corpus-extracted AAN vectors, finding that the distinction between "flexible" AANs, where the adjective order can be flipped, and "rigid" ones, where the order is fixed, is reflected in distributional cues. These results all derive from the intuition that the most embedded adjective in a rigid AAN has a very strong effect on the distributional semantic representation of the AAN. Most compositional models were able to capture at least some of the same cues that emerged in the analysis of the corpus-extracted vectors.

Finally, similar cues were also shown to distinguish (compositional) representations of rigid AANs in the "correct" (corpus-attested) and "wrong" (unattested) orders, again pointing to the degree to which the (attested-order) closest adjective affects the overall AAN meaning as an important factor.

Comparing the composition functions, we find that the linguistically motivated LFM approach has the most consistent performance across all our tests. This model significantly outperformed all others in approximating high-quality corpus-extracted AAN vectors, it provided the closest approximation to the corpus-observed patterns when distinguishing flexible and rigid AANs, and it was one of the models with the strongest cues distinguishing attested and unattested orders of rigid AANs.

From an applied point of view, a natural next step would be to use the cues we proposed as features to train a classifier to predict the preferred order of adjectives, to be tested also in cases where neither order is found in the corpus, so direct corpus evidence cannot help. For a full account of adjectival ordering, non-semantic factors should also be taken into account. As shown by the effectiveness in our experiments of PMI, which is a classic measure used to harvest idioms and other multiword expressions (Church and Hanks, 1990), ordering is affected by arbitrary lexicalization patterns. Metrical effects are also likely to play a role, like they do in the well-studied case of "binomials" such as *salt and pepper* (Benor and Levy, 2006; Copestake and Herbelot, 2011). In a pilot study, we found that indeed word length (roughly quantified by number of letters) is a significant factor in predicting adjective ordering (the shorter adjective being more likely to occur first), but its effect is not nearly as strong as that of the semantic measures we considered here. In our future work, we would like to develop an order model that exploits semantic, metrical and lexicalization features jointly for maximal classification accuracy.

Adjectival ordering information could be useful in parsing: in English, it could tell whether an AANN sequence should be parsed as A[[AN]N] or A[A[NN]]; in languages with pre- and post-N adjectives, like Italian or Spanish, it could tell whether ANA sequences should be parsed as A[NA] or [AN]A. The ability to detect ordering restrictions could also help Natural Language Generation tasks (Malouf, 2000), especially for the generation of unattested combinations of As and Ns.

From a theoretical point of view, we would like to extend our analysis to adjective coordination (what's

the difference between *new and creative idea* and *new creative idea*?). Additionally, we could go more granular, looking at whether compositional models can help us to understand why certain classes of adjectives are more likely to precede or follow others (why is size more likely to take scope over color, so that *big red car* sounds more natural than *red big car*?) or studying the behaviour of specific adjectives (can our approach capture the fact that *strong alcoholic drink* is preferable to *alcoholic strong drink* because *strong* pertains to the alcoholic properties of the drink?).

In the meantime, we hope that the results we reported here provide convincing evidence of the usefulness of compositional distributional semantics in tackling topics, such as recursive adjectival modification, that have been of traditional interest to theoretical linguists from a new perspective.

Acknowledgments

We would like to thank the anonymous reviewers, Fabio Massimo Zanzotto, Yao-Zhong Zhang and the members of the COMPOSES team. This research was supported by the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Sarah Bunin Benor and Roger Levy. 2006. The chicken or the egg? A probabilistic analysis of english binomials. *Language*, pages 233–278.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on EMNLP and CoNLL*, pages 546–556, Jeju Island, Korea.

Gemma Boleda, Eva Maria Vecchi, Miquel Cornudella, and Louise McNally. 2012. First-order vs. higher-order modification in distributional semantics. In *Proceedings of the 2012 Joint Conference on EMNLP and CoNLL*, pages 1223–1233, Jeju Island, Korea.

Gemma Boleda, Marco Baroni, Louise McNally, and Nghia Pham. 2013. Intensionality was only alleged: On adjective-noun composition in distributional semantics. In *Proceedings of IWCS*, pages 35–46, Potsdam, Germany.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12, Los Angeles, CA.

Kenneth Church and Peter Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Guglielmo Cinque, editor. 2002. *Functional Structure in DP and IP - The Carthography of Syntactic Structures*, volume 1. Oxford University Press.

Guglielmo Cinque. 2004. Issues in adverbial syntax. *Lingua*, 114:683–710.

Guglielmo Cinque. 2010. *The syntax of adjectives: a comparative study*. MIT Press.

Ann Copestake and Aurélie Herbelot. 2011. Exciting and interesting: issues in the generation of binomials. In *Proceedings of the UCNLG+ Eval: Language Generation and Evaluation Workshop*, pages 45–53, Edinburgh, UK.

Paola Crisma. 1991. Functional categories inside the noun phrase: A study on the distribution of nominal modifiers. "Tesi di Laurea", University of Venice.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013a. DISSECT: DIStributional SEmantics Composition Toolkit. In *Proceedings of the System Demonstrations of ACL 2013*, East Stroudsburg, PA.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013b. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the ACL 2013 Workshop on Continuous Vector Space Models and their Compositionality (CVSC 2013)*, East Stroudsburg, PA.

Gottlob Frege. 1892. Über sinn und bedeutung. *Zeitschrift fuer Philosophie un philosophische Kritik*, 100.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, Edinburgh, UK.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the ACL GEMS Workshop*, pages 33–37, Uppsala, Sweden.

Chih-Jen Lin. 2007. Projected gradient methods for Nonnegative Matrix Factorization. *Neural Computation*, 19(10):2756–2779.

Robert Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of ACL*, pages 85–92, East Stroudsburg, PA.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Richard Montague. 1970. Universal Grammar. *Theoria*, 36:373–398.

Robert Munro, Steven Bethard, Victor Kuperman, Vicky Tzuyin Lai, Robin Melnick, Christopher Potts, Tyler Schnoebelen, and Harry Tily. 2010. Crowdsourcing and language studies: the new generation of linguistic data. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 122–130, Los Angeles, CA.

Barbara Partee. 2004. Compositionality. In *Compositionality in Formal Semantics: Selected Papers by Barbara H. Partee*. Blackwell, Oxford.

Magnus Sahlgren. 2006. *The Word-Space Model*. Dissertation, Stockholm University.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the EACL-SIGDAT Workshop*, Dublin, Ireland.

Hinrich Schütze. 1997. *Ambiguity Resolution in Natural Language Learning*. CSLI, Stanford, CA.

Gary-John Scott. 2002. Stacked adjectival modification and the structure of nominal phrases. In Guglielmo Cinque, editor, *Functional Structure in DP and IP. The Carthography of Syntactic Structures*, volume 1. Oxford University Press.

Richard Socher, E.H. Huang, J. Pennington, Andrew Y. Ng, and C.D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems*, 24:801–809.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Edinburgh, UK.

Richard Sproat and Chilin Shih. 1990. The cross-linguistics distribution of adjective ordering restrictions. In C. Georgopoulos and Ishihara R., editors, *Interdisciplinary approaches to language: essays in honor of Yuki Kuroda*, pages 565–593. Kluver, Dordrecht.

Sam Steddy and Vieri Samek-Lodovici. 2011. On the ungrammaticality of remnant movement in the derivation of greenberg's universal 20. *Linguistic Inquiry*, 42(3):445–469.

Richmond H. Thomason, editor. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New York.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (Linear) maps of the impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the ACL Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, OR.

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.

Learning Latent Word Representations for Domain Adaptation using Supervised Word Clustering

Min Xiao and **Feipeng Zhao** and **Yuhong Guo**
Department of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
{minxiao,feipeng.zhao,yuhong}@temple.edu

Abstract

Domain adaptation has been popularly studied on exploiting labeled information from a source domain to learn a prediction model in a target domain. In this paper, we develop a novel representation learning approach to address domain adaptation for text classification with automatically induced discriminative latent features, which are generalizable across domains while informative to the prediction task. Specifically, we propose a hierarchical multinomial Naive Bayes model with latent variables to conduct supervised word clustering on labeled documents from both source and target domains, and then use the produced cluster distribution of each word as its latent feature representation for domain adaptation. We train this latent graphical model using a simple expectation-maximization (EM) algorithm. We empirically evaluate the proposed method with both cross-domain document categorization tasks on Reuters-21578 dataset and cross-domain sentiment classification tasks on Amazon product review dataset. The experimental results demonstrate that our proposed approach achieves superior performance compared with alternative methods.

1 Introduction

Supervised prediction models typically require a large amount of labeled data for training. However, manually collecting data annotations is expensive in many real-world applications such as document categorization or sentiment classification. Recently, domain adaptation has been proposed to exploit existing labeled data in a related source domain to assist the prediction model training in the target domain (Ben-David et al., 2006; Blitzer et al., 2006; Daumé III, 2007; Blitzer et al., 2011; Chen et al., 2012). As an effective tool to reduce annotation effort, domain adaptation has achieved success in various cross-domain natural language processing (NLP) systems such as document categorization (Dai et al., 2007), sentiment classification (Blitzer et al., 2007; Chen et al., 2012; Mejova and Srinivasan, 2012; Chen et al., 2011), email spam detection (Jiang and Zhai, 2007), and a number of other NLP tasks (Blitzer et al., 2011; Daumé III, 2007).

One primary challenge of domain adaptation lies in the distribution divergence of the two domains in the original feature representation space. For example, documents about *books* may contain very different high-frequency words and discriminative words from documents about *kitchen*. A good cross-domain feature *representation* thus has been viewed as critical for bridging the domain divergence gap and facilitating domain adaptation in the NLP area (Ben-David et al., 2006, 2010). Many domain adaptation works have been proposed to learn new cross-domain feature representations (Blitzer et al., 2006, 2011). Though demonstrated good performance on certain problems, these works mostly induce new feature representations in an unsupervised way, without taking the valuable label information into account.

In this work, we present a novel supervised representation learning approach to discover a latent representation of words which is not only generalizable across domains but also informative to the classification task. Specifically, we propose a hier-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 152–162,
Seattle, Washington, USA, 18-21 October 2013.

archical multinomial Naive Bayes model with latent word cluster variables to perform supervised word clustering on labeled documents from both domains. Our model directly models the relationships between the observed document label variables and the latent word cluster variables. The induced cluster representation of each word thus will be informative for the classification labels, and hence discriminative for the target classification task. We train this directed graphical model using an expectation-maximization (EM) algorithm, which maximizes the log-likelihood of the observations of labeled documents. The induced cluster distribution of each word can then be used as its generalizable representation to construct new cluster-based representation of each document. For domain adaptation, we train a supervised learning system with labeled data from both domains in the new representation space and apply it to categorize test documents in the target domain. In order to evaluate the proposed technique, we conduct extensive experiments on the Reuters-21578 dataset for cross-domain document categorization and on Amazon product review dataset for cross-domain sentiment classification. The experimental results show the proposed approach can produce more effective representations than the comparison domain adaptation methods.

2 Related Work

Domain adaptation has recently been popularly studied in natural language processing and a variety of domain adaptation approaches have been developed, including instance weighting adaptation methods and feature representation learning methods.

Instance weighting adaptation methods improve the transferability of a prediction model by training an instance weighted learning system. Much work in this category has been developed to address different weighting schemas (Sugiyama et al., 2007; Wan et al., 2011). Jiang and Zhai (2007) applied instance weighting algorithms to tackle cross-domain NLP tasks and proposed to remove misleading source training data and assign less weights to labeled data from the source domain than labeled data from the target domain. Dai et al. (2007) proposed to increase the weights of mistakenly predicted instances from the target domain and decrease the weights of incorrectly predicted instances from the source domain during an iterative training process.

Representation learning methods bridge domain divergence either by differentiating domain-invariant features from domain-specific features (Daumé III, 2007; Daumé III et al., 2010; Blitzer et al., 2011; Finkel and Manning, 2009) or seeking generalizable latent features across domains (Blitzer et al., 2006, 2007; Prettenhofer and Stein, 2010). Daumé III (2007); Daumé III et al. (2010) proposed a simple heuristic feature replication method to represent common, source specific and target specific features. Finkel and Manning (2009) proposed a former version of it based on the use of a hierarchical Bayesian prior. Blitzer et al. (2011) proposed a coupled subspace learning method, which learns two projectors, one for each domain, to project the original features into domain-sharing and domain-specific features. Blitzer et al. (2006) proposed a structural correspondence learning (SCL) method to model the correlation between pivot features and non-pivot features. It uses the correlation to induce latent domain-invariant features as augmenting features for supervised learning. Extensions of this work include improving pivot feature selection (Blitzer et al., 2007; Prettenhofer and Stein, 2010), and improving the correlation modeling between pivot and non-pivot features (Tan, 2009).

The proposed approach in this paper belongs to representation learning methods. However, unlike the unsupervised representation learning methods reviewed above, our proposed approach learns generalizable feature representations of words by exploiting data labels from the two domains.

3 Learning Latent Word Representations using Supervised Word Clustering

In this paper, we address domain adaptation for text classification. Given a source domain $\mathcal{D}_{\mathcal{S}}$ with plenty of labeled documents, and a target domain $\mathcal{D}_{\mathcal{T}}$ with a very few labeled documents, the task is to learn a classifier from the labeled documents in both domains, and use it to classify the unlabeled documents in the target domain. The documents in the two domains share the same universal vocabulary $\mathcal{V} = \{w_1, w_2, \cdots, w_n\}$, but the word distributions in the two domains are typically different.

Therefore, training the classification model directly from the original word feature space $\mathcal{V}$ may not generalize well in the target domain.

We propose to address this problem by first learning a supervised mapping function $\phi : \mathcal{V} \longrightarrow \mathcal{Z}$ from the labeled documents in both domains, which maps the input word features in the large vocabulary set $\mathcal{V}$ into a low dimensional latent feature space $\mathcal{Z}$. By filtering out unimportant details and noises, we expect the low dimensional mapping can capture the intrinsic structure of the input data that is discriminative for the classification task and generalizable across domains. In particular, we learn such a mapping function by conducting supervised word clustering on the labeled documents using a hierarchical multinomial Naive Bayes model. Below, we will first introduce this supervised word clustering model and then use the mapping function produced to transform documents in different domains into the same low-dimensional space for training cross domain text classification systems.

3.1 Supervised Word Clustering

Given all labeled documents from the source and target domains, $D = \{(\mathbf{w}_t, y_t)\}_{t=1}^{T}$, where the t-th labeled document is expressed as a bag of words, $\mathbf{w}_t = \{w_{t1}, w_{t2}, \cdots, w_{tN_t}\}$, and its label value is $y_t \in \mathcal{Y}$ for $\mathcal{Y} = \{1, \cdots, K\}$, we propose to perform supervised word clustering by modeling the document-label pair distribution using a hierarchical multinomial Naive Bayes model given in Figure 1, which has a middle layer of latent cluster variables.

In this plate model, the variable Y_t denotes the observed class label for the t-th document, and all the label variables, $\{Y_t\}_{t=1}^{T}$, share the same multinomial distribution θ_Y across documents. The latent variable $C_{t,i}$ denotes the cluster membership of the word $W_{t,i}$, and all the cluster variables, $\{C_{t,i}\}_{t=1,i=1}^{T,N_t}$, share the same set of conditional distributions $\{\theta_{C|y}\}_{y=1}^{K}$ across documents and words. The variable $W_{t,i}$ denotes the i-th observed word in the t-th document, and all the word variables, $\{W_{t,i}\}_{t=1,i=1}^{T,N_t}$, share the same set of conditional distributions $\{\theta_{W|c}\}_{c=1}^{m}$. Here we assume the number of word clusters is m. For simplicity, we do not show the distribution parameter variables in the Figure.

Following the *Markov property* of directed graphical models, we can see that given the cluster variable values, the document label variables will be completely independent of the word variables. By learning this latent directed graphical model, we thus expect the important classification information expressed in the input observation words can be effectively summarized into the latent cluster variables. This latent model is much simpler than the supervised topic models (Blei and McAuliffe, 2007), but we will show later that it can suitably produce a generalizable feature mapping function for domain adaptation.

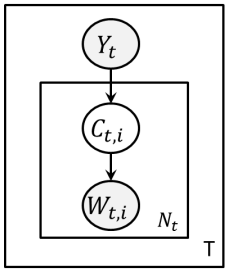


Figure 1: Supervised word clustering model.

To train the latent graphical model in Figure 1 on labeled documents D, we use a standard expectation-maximization (EM) algorithm (Dempster et al., 1977) to maximize the marginal log-likelihood of the observations:

$$LL(D; \boldsymbol{\theta}) = \sum_{t} \log P(y_t, \mathbf{w}_t | \boldsymbol{\theta}) \qquad (1)$$

The EM algorithm is an iterative procedure. In each iteration, it takes an alternative E-step and M-step to maximize the lower bound of the marginal log-likelihood function. In our experiments, we start from a random initialization of the model parameters and the latent variable values, and then perform iterative EM updates until converge to a local optimal solution.

3.2 Induced Word Representation

After training the supervised clustering model using EM algorithm, a set of local optimal model parameters $\boldsymbol{\theta}^*$ will be returned, which define a joint distribution over the three groups of variables in the directed graphical model. Next we define a supervised latent feature mapping function ϕ from this trained

model to map each word w in the vocabulary $\mathcal{V}$ into a conditional distribution vector over the word cluster variable, such as

$$\phi(w) = [P(c=1|w, \boldsymbol{\theta}^*), \cdots, P(c=m|w, \boldsymbol{\theta}^*)]. \quad (2)$$

The conditional distributions involved in this mapping function can be computed as

$$P(c|w, \boldsymbol{\theta}^*) = \frac{\sum_{y \in \mathcal{Y}} P(w|c, \boldsymbol{\theta}^*) P(c|y, \boldsymbol{\theta}^*) P(y|\boldsymbol{\theta}^*)}{P(w)} \quad (3)$$

where $P(w|c, \boldsymbol{\theta}^*) = \boldsymbol{\theta}^*_{w|c}$ $P(c|y, \boldsymbol{\theta}^*) = \boldsymbol{\theta}^*_{c|y}$ and $P(y|\boldsymbol{\theta}^*) = \boldsymbol{\theta}^*_y$ can be determined from the model parameters directly, and $p(w)$ can be computed as the empirical frequency of word w among all the other words in all the training documents.

We then define a transformation matrix $\Pi \in \mathbb{R}^{n \times m}$ based on the mapping function ϕ defined in Eq. (2), such that $\Pi_{i:} = \phi(w_i)$ where w_i is the i-th word in the vocabulary $\mathcal{V}$. That is, each row of Π is the induced representation vector for one word. Π can be viewed as a soft word clustering matrix, and $\Pi_{i,j}$ denotes the probability of word w_i belongs to the j-th cluster. Given the original document-word frequency matrix $X_{tr} \in \mathbb{R}^{T \times n}$ for the labeled training documents from the two domains, we can construct its representations $Z_{tr} \in \mathbb{R}^{T \times m}$ in the predictive latent clustering space by performing the following transformation:

$$Z_{tr} = X_{tr} \Pi. \quad (4)$$

Similarly, we can construct the new representation matrix Z_{ts} for the test data X_{ts} in the target domain. We then train a classification model on the labeled data Z_{tr} and apply it to classify the test data Z_{ts}.

4 Experiments

We evaluate the proposed approach with experiments on cross domain document categorization of Reuters data and cross domain sentiment classification of Amazon product reviews, comparing to a number of baseline and existing domain adaptation methods. In this section, we report the experimental setting and results on these two data sets.

4.1 Approaches

We compared our proposed supervised word clustering approach (*SWC*) with the following five comparison methods for domain adaptation:

(1) *BOW*: This is a bag-of-word baseline method, which trains a SVM classifier with labeled data from both domains using the original bag-of-word features.

(2) *PLSA*: This is an unsupervised word clustering method, which first applies the probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) to obtain word clusterings with both labeled and unlabeled data from the two domains and then uses the soft word clusterings as augmenting features to train SVM classifiers.

(3) *FDLDA*: This is an alternative supervised word clustering method we built by training the Fast-Discriminative Latent Dirichlet Allocation model (Shan et al., 2009) with all labeled data from the two domains. After training the model, we used the learned topic distribution $p(z)$ and the conditional word distributions $p(w|z)$ to compute the conditional distribution over topics $p(z|w)$ for each word as the soft clustering of the word. We then used the soft word clusterings as augmenting features to train SVM classifiers.

(4) *SCL*: This is the structural correspondence learning based domain adaptation method (Blitzer et al., 2006). It first induces generalizable features with all data from both domains by modeling the correlations between pivot features and non-pivot features, and then uses the produced generalizable features as augmenting features to train SVM classifiers.

(5) *CPSP*: This is coupled subspace learning based domain adaptation method (Blitzer et al., 2011). It first learns two domain projectors using all data from the two domains by approximating multi-view dimensionality reduction, and then projects the labeled data to low dimensional latent feature space to train SVM Classifiers.

We used the LIBSVM package (Chang and Lin, 2011) with its default parameter setting to train linear SVM classifiers as the base classification model for all comparison methods.

Table 1: Average results (accuracy±standard deviation) for three cross-domain document categorization tasks on Reuters-21578 dataset.

Task	BOW	PLSA	FDLDA	SCL	CPSP	SWC
Orgs vs People	76.07±0.39	76.50±0.10	76.95±0.23	78.71±0.20	77.58±0.21	**81.27±0.23**
Orgs vs Places	73.88±0.58	74.68±0.20	74.87±0.29	76.71±0.23	75.76±0.28	**78.33±0.64**
People vs Places	61.80±0.44	63.36±0.40	63.46±0.40	64.65±0.40	62.73±0.53	**67.48±0.20**

4.2 Experiments on Reuters Data Set

We used the popularly studied Reuters-21578 dataset (Dai et al., 2007), which contains three cross-domain document categorization tasks, *Orgs vs People, Orgs vs Places, People vs Places*. The source and target domains of each task contain documents sampled from different non-overlapping subcategories. From example, the task of *Orgs vs People* assigns a document into one of the two top categories (*Orgs, People*), and the source domain documents and the target domain documents are sampled from different subcategories of *Orgs* and *People*. There are 1237 source documents and 1208 target documents for the task of *Orgs vs People*, 1016 source documents and 1043 target documents for the task of *Orgs vs Places*, and 1077 source documents and 1077 target documents for the task of*People vs Places*. For each task, we built a unigram vocabulary based on all the documents from the two domains and represented each document as a feature vector containing term frequency values.

4.2.1 Experimental Results for Cross-Domain Document Categorization

For each of the three cross-domain document categorization tasks on Reuters-21578 dataset, we used all the source documents as labeled training data while randomly selecting 100 target documents as labeled training data and setting the rest as unlabeled test data. For the BOW baseline method, we used the term-frequency features. The other five approaches are based on representation learning, and we selected the dimension size of the representation learning, i.e., the cluster number in our proposed approach, from $\{5, 10, 20, 50, 100\}$ according to the average classification results over 3 runs on the task of *Orgs vs People*. The dimension sizes of the induced representations for the five approaches, *PLSA, FDLDA, SCL, CPSP* and *SWC* are *20, 20, 100, 100* and *20* respectively.

We then repeated each experiment 10 times on each task with different random selections of the 100 labeled target documents to compare the six comparison approaches. The average classification results in terms of accuracy and standard deviations are reported in Table 1. We can see that by simply combining labeled documents from the two domains without adaptation, the *BOW* method performs poorly across the three tasks. The *PLSA* method outperforms the *BOW* method over all the three tasks with small improvements. The supervised word clustering method *FDLDA*, though performing slightly better than the unsupervised clustering method *PLSA*, produces poor performance comparing to the proposed *SWC* method. One possible reason is that the *FDLDA* model is not specialized for supervised word clustering, and it uses a logistic regression model to predict the labels from the word topics, while the final soft word clustering is computed from the learned distribution $p(z)$ and $p(w|z)$. That is, in the FDLDA model the labels only influence the word clusterings indirectly and hence its influence can be much smaller than the influence of labels as direct parent variables of the word cluster variables in the *SWC* model. The two domain adaptation approaches, *SCL* and *CPSP*, both produce significant improvements over *BOW*, *PLSA* and *FDLDA* on the two tasks of *Orgs vs People* and *Orgs vs Places*, while the *CPSP* method produces slightly inferior performance than *PLSA* and *FDLDA* on the task of *People vs Places*. The proposed method *SWC* on the other hand consistently and significantly outperforms all the other comparison methods across all the three tasks.

We also studied the sensitivity of the proposed approach with respect to the number of clusters,

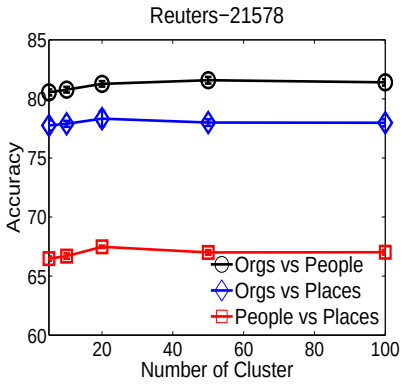


Figure 2: Sensitivity analysis of the proposed approach w.r.t. the number of clusters for the three cross-domain document categorization tasks on Reuters-21578 dataset.

i.e., the dimension size of the learned representation. We experimented with a set of different values $m \in \{5, 10, 20, 50, 100\}$ as the number of clusters. For each m value, we used the same experimental setting as above and repeated the experiments 10 times to obtain the average comparison results. The classification accuracy results on the three tasks are reported in Figure 2. We can see that the proposed method is not very sensitive to the number of clusters, across the set of increasing values we considered, and its performance becomes very stable after the cluster number reaches 20.

4.2.2 Document Categorization Accuracy vs Label Complexity in Target Domain

We next conducted experiments to compare the six approaches by varying the amount of the labeled data from the target domain. We tested a set of different values, $s \in \{100, 200, 300, 400, 500\}$, as the number of labeled documents from the target domain. For each different s value, we repeated the experiments 10 times by randomly selecting s labeled documents from the target domain using the same experimental setting as before. The comparison results across the set of s values are plotted in Figure 3. We can see that in general the performance of each method improves with the increase of the number of labeled documents from the target domain. The proposed method *SWC* and the domain adaptation method *SCL* clearly outperform the other four methods. Moreover, the proposed method *SWC* not only maintains consistent and significant advantages over all other methods across the range of different s values, its performance with 300 labeled target instances is even superior to the other methods with 500 labeled target instances. All these results suggest the proposed approach is very effective for adapting data across domains.

4.3 Experiments on Amazon Product Reviews

We conducted cross-domain sentiment classification on the widely used Amazon product reviews (Blitzer et al., 2007), which contains review documents distributed in four categories: *Books(B)*, *DVD(D)*, *Electronics(E)* and *Kitchen(K)*. Each category contains 1000 positive and 1000 negative reviews. We constructed 12 cross-domain sentiment classification tasks, one for each source-target domain pair, *B2D*, *B2E*, *B2K*, *D2B*, *D2E*, *D2K*, *E2B*, *E2D*, *E2K*, *K2B*, *K2D*, *K2E*. For example, the task *B2D* means that we use the *Books* reviews as the source domain and the *DVD* reviews as the target domain. For each pair of domains, we built a vocabulary with both unigram and bigram features extracted from all the documents of the two domains, and then represented each review document as a feature vector with term frequency values.

4.3.1 Experimental Results for Cross-Domain Sentiment Classification

For each of the twelve cross-domain sentiment classification tasks on Amazon product reviews, we used all the source reviews as labeled data and randomly selected 100 target reviews as labeled data while treating the rest as unlabeled test data. For the baseline method *BOW*, we used binary indicator values as features, which has been shown to work better than the term-frequency features for sentiment classification tasks (Pang et al., 2002; Na et al., 2004). For all the other representation learning based methods, we selected the dimension size of learned representation according to the average results over 3 runs on the *B2D* task. The dimension sizes selected for the methods *PLSA*, *FDLDA*, *SCL*, *CPSP*, and *SWC* are 10, 50, 50, 100 and 10, respectively.[1]

[1] 50 and 100 are also the suggested values for SCL (Blitzer et al., 2007) and CPSP (Blitzer et al., 2011) respectively on this cross-domain sentiment classification dataset.

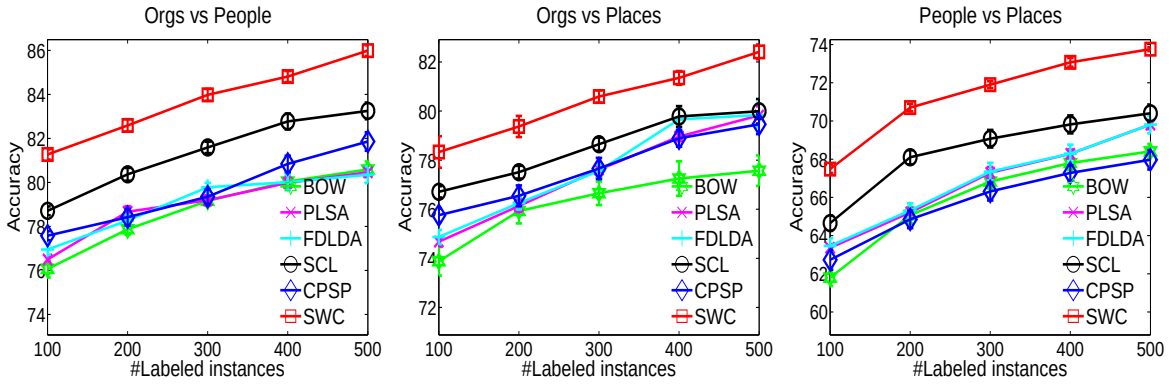


Figure 3: Average classification results for three cross-domain document categorization tasks on Reuters-21578 dataset by varying the amount of labeled training data from the target domain.

Table 2: Average results (accuracy±standard deviation) for twelve cross-domain sentiment classification tasks on Amazon product reviews.

Task	**BOW**	**PLSA**	**FDLDA**	**SCL**	**CPSP**	**SWC**
B2D	76.58±0.14	76.01±0.10	75.95±0.16	80.17±0.16	77.53±0.14	**81.66**±0.23
B2K	75.48±0.34	74.68±0.20	74.87±0.15	78.13±0.21	76.38±0.15	**82.26**±0.20
B2E	72.92±0.37	73.36±0.19	73.46±0.21	74.79±0.19	73.31±0.17	**77.04**±0.64
D2B	74.10±0.29	74.04±0.20	74.08±0.18	78.73±0.23	77.07±0.15	**79.95**±0.25
D2K	75.19±0.33	75.37±0.31	75.44±0.31	76.98±0.19	76.77±0.10	**82.13**±0.20
D2E	73.01±0.34	74.21±0.30	74.09±0.31	75.69±0.25	73.83±0.21	**76.98**±0.54
E2B	67.58±0.24	68.48±0.15	68.44±0.17	70.21±0.16	70.47±0.16	**72.11**±0.46
E2D	70.15±0.27	70.16±0.23	70.06±0.22	72.83±0.25	71.76±0.20	**73.81**±0.59
E2K	82.23±0.12	82.24±0.18	82.26±0.19	84.69±0.11	81.31±0.14	**85.33**±0.16
K2B	70.67±0.18	72.18±0.21	72.18±0.16	73.91±0.21	72.18±0.19	**75.78**±0.55
K2D	71.51±0.26	72.00±0.18	72.05±0.19	74.82±0.26	72.59±0.18	**76.88**±0.49
K2E	80.81±0.12	80.39±0.18	80.46±0.18	82.96±0.11	80.81±0.14	**84.78**±0.19

We then repeated each experiment 10 times based on different random selections of 100 labeled reviews from the target domain to compare the six methods on the twelve tasks. The average classification results are reported in Table 2. We can see that the *PLSA* and *FDLDA* methods do not show much advantage over the baseline method *BOW*. *CPSP* performs better than *PLSA* and *BOW* on many of the twelve tasks, but with small advantages, while *SCL* outperforms *CPSP* on most tasks. The proposed method *SWC* however demonstrates a clear advantage over all the other methods and produces the best results on all the twelve tasks.

We also conducted sensitivity analysis over the proposed approach regarding the number of clusters on the twelve cross-domain sentiment classification tasks, by testing a set of cluster number values $m = \{5, 10, 20, 50, 100\}$. The average results are plotted in Figure 5. Similar as before, we can see the proposed approach has stable performance across the set of different cluster numbers. Moreover, these results also clearly show that domain adaptation is not a symmetric process, as we can see it is easier to conduct domain adaptation from the source domain *Books* to the target domain *Kitchen* (with an accuracy around 82%), but it is more difficult to make domain adaptation from the source domain *Kitchen* to the target domain *Books* (with an ac-

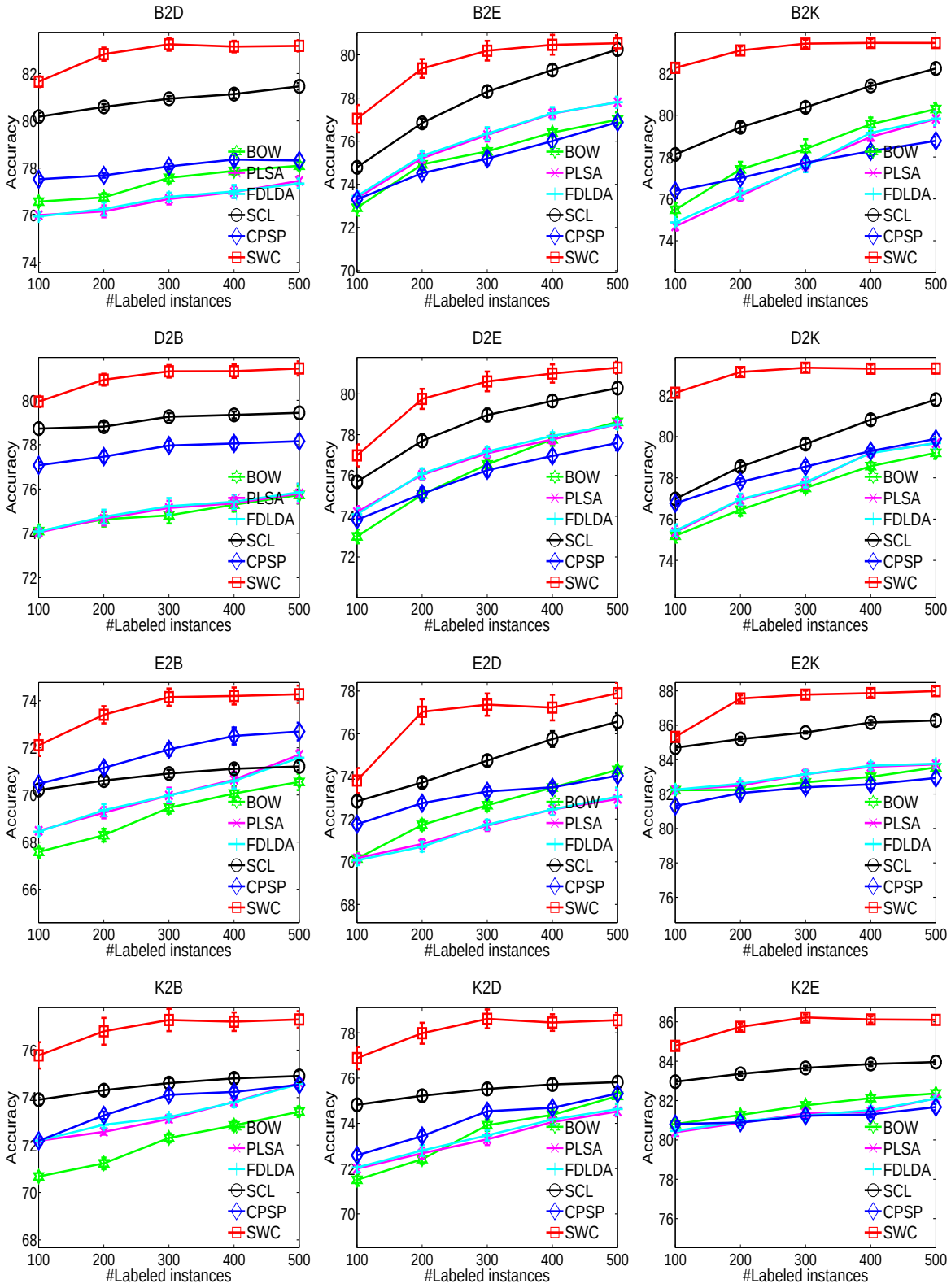


Figure 4: Average results (accuracy±standard deviation) for the 12 cross-domain sentiment classification tasks on Amazon product reviews with different numbers of labeled training data from the target domain.

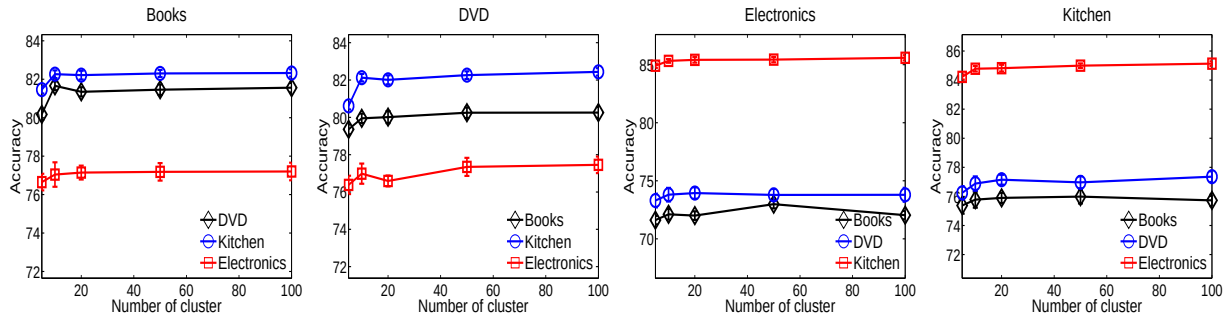


Figure 5: Sensitivity analysis of the proposed approach wrt the number of clusters for the twelve cross-domain sentiment classification tasks. Each figure shows experimental results for three tasks with the same source domain.

curacy around 75%). It also shows that the degree of relatedness of the two domains is an important factor for the effectiveness of knowledge adaptation. For example, one can see that it is much easier to conduct domain adaptation from *Kitchen* to *Electronics* (with an accuracy around 84%) than from *Kitchen* to *Books* (with an accuracy around 75%), as *Kitchen* is more closely related to *Electronics* than *Books*.

4.3.2 Sentiment Classification Accuracy vs Label Complexity in Target Domain

Similar as before, we tested the proposed approach using a set of different values $s \in \{100, 200, 300, 400, 500\}$ as the number of labeled reviews from the target domain. For each given s value, we conducted the comparison experiments using the same setting above. The average results are reported in Figure 4. We can see that the performance of each approach in general improves with the increase of the number of labeled reviews from the target domain. The proposed approach maintains a clear advantage over all the other methods on all the twelve tasks across different label complexities. All those empirical results demonstrate the effectiveness of the proposed approach for cross-domain sentiment classification.

4.3.3 Illustration of the Word Clusters

Finally, we would also like to demonstrate the *hard* word clusters produced by the proposed supervised word clustering method. We assign a word into the cluster it most likely belongs to according to its soft clustering representation, such as $c^* = \arg\max_c P(c|w, \boldsymbol{\theta}^*)$. Table 3 presents the top representative words (i.e., the most frequent words) of the 10 word clusters produced on the task of *B2K*. We can see that the first three clusters (C1, C2, and C3) contain words with *positive* sentiment polarity in different degrees. The two clusters (C4 and C5) contain words used to express the degree of opinions. The next four clusters (C6, C7, C8, and C9) contain content words related to *Books* or *Kitchen*. The last cluster (C10) contains words of *negative* sentiment polarity. These results demonstrate that the proposed supervised word clustering can produce task meaningful word clusters and hence label-informative latent features, which justifies its effectiveness.

5 Conclusion

In this paper, we proposed a novel supervised representation learning method to tackle domain adaptation by inducing predictive latent features based on supervised word clustering. With the soft word clustering produced, we can transform all documents from the two domains into a unified low-dimensional feature space for effective training of cross-domain NLP prediction system. We conducted extensive experiments on cross-domain document categorization tasks on Reuters-21578 dataset and cross-domain sentiment classification tasks on Amazon product reviews. Our empirical results demonstrated the efficacy of the proposed approach.

References

S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adapta-

Table 3: Clustering illustration for the task of *B2K* on Amazon product reviews.

C1	recommend excellent wonderful beautiful love powerful happy satisfied outstanding
C2	enjoyed fantastic glad i_liked nicely was_great benefits pleasure amazingly
C3	good_and made_me most_people ordered_this standards accurately check_out
C4	was_a kind_of basically is_only half_of first_of as_if and_still anything_about have_some
C5	ever may still going maybe either at_least of_all totally sort_of are_very
C6	life work machine size design bottom business picture hand hook gas sink turner shelves
C7	way coffee pan keep cooking maker heat job working children handle meet core wine
C8	people us world come fact man place stars during example went short bathroom apple price
C9	pot friends daily light fire tells knew holds keep_the continued meal hooked silver wind
C10	disappointed waste unfortunately worse poorly sorry weak not_worth stupid fails awful useless

tion. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan. A theory of learning from different domains. *Machine Learng*, 79(1-2):151–175, 2010.

D. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

J. Blitzer, D. Foster, and S. Kakade. Domain adaptation with coupled subspaces. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

M. Chen, K. Weinberger, and J. Blitzer. Co-training for domain adaptation. In *Advances in Neural Inform. Process. Systems (NIPS)*, 2011.

M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proc. of the International Conf. on Machine Learning (ICML)*, 2012.

W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *Proc. of the International Conf. on Machine Learning (ICML)*, 2007.

H. Daumé III. Frustratingly easy domain adaptation. In *Proc. of the Annual Meeting of the Association for Comput. Linguistics (ACL)*, 2007.

H. Daumé III, A. Kumar, and A. Saha. Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society*, 39 (1):1–38, 1977.

J. Finkel and C. Manning. Hierarchical bayesian domain adaptation. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2009.

T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2007.

Y. Mejova and P. Srinivasan. Crossing media streams with sentiment: Domain adaptation in

blogs, reviews and twitter. In *Proc. of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2012.

J. Na, H. Sui, C. Khoo, S. Chan, and Y. Zhou. Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews. In *Proc. of the Conf. of the Inter. Society for Knowledge Organization*, 2004.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

P. Prettenhofer and B. Stein. Cross-language text classification using structural correspondence learning. In *Proc. of the Annual Meeting of the Association for Comput. Linguistics (ACL)*, 2010.

H. Shan, A. Banerjee, and N. Oza. Discriminative mixed-membership models. In *Proc. of the IEEE Inter. Conference on Data Mining (ICDM)*, 2009.

M. Sugiyama, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

S. Tan. Improving scl model for sentiment-transfer learning. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2009.

C. Wan, R. Pan, and J. Li. Bi-weighting domain adaptation for cross-language text classification. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

Appropriately Incorporating Statistical Significance in PMI

Om P. Damani and Shweta Ghonge
IIT Bombay
India
{damani,shwetaghonge}@cse.iitb.ac.in

Abstract

Two recent measures incorporate the notion of statistical significance in basic PMI formulation. In some tasks, we find that the new measures perform worse than the PMI. Our analysis shows that while the basic ideas in incorporating statistical significance in PMI are reasonable, they have been applied slightly inappropriately. By fixing this, we get new measures that improve performance over not just PMI but on other popular co-occurrence measures as well. In fact, the revised measures perform reasonably well compared with more resource intensive non co-occurrence based methods also.

1 Introduction

The notion of *word association* is used in many language processing and information retrieval applications and it is important to have low-cost, high-quality association measures. Lexical co-occurrence based word association measures are popular because they are computationally efficient and they can be applied to any language easily. One of the most popular co-occurrence measure is Pointwise Mutual Information (PMI) (Church and Hanks, 1989).

One of the limitations of PMI is that it only works with relative probabilities and ignores the absolute amount of evidence. To overcome this, recently two new measures have been proposed that incorporate the notion of statistical significance in basic PMI formulation. In (Washtell and Markert, 2009), statistical significance is introduced in PMI_{sig} by multiplying PMI value with the square root of the evidence. In contrast, in (Damani, 2013), cPMId is introduced by bounding the probability of observing a given deviation between a given word pair's co-occurrence count and its expected value under a null model where with each word a global unigram generation probability is associated. In Table 1, we give the definitions of PMI, PMI_{sig}, and cPMId.

While these new measures perform better than PMI on some of the tasks, on many other tasks, we find that the new measures perform worse than the PMI. In Table 3, we show how these measures perform compared to PMI on four different tasks. We find that PMI_{sig} degrades performance in three out of these four tasks while cPMId degrades performance in two out of these four tasks. The experimental details and discussion are given in Section 4.2.

Our analysis shows that while the basic ideas in incorporating statistical significance are reasonable, they have been applied slightly inappropriately. By fixing this, we get new measures that improve performance over not just PMI, but also on other popular co-occurrence measures on most of these tasks. In fact, the revised measures perform reasonably well compared with more resource intensive non co-occurrence based methods also.

2 Adapting PMI for Statistical Significance

In (Washtell and Markert, 2009), it is assumed that the statistical significance of a word pair association is proportional to the square root of the evidence. The question of what constitutes the evidence is answered by taking the lesser of the frequencies of the two words in the word pair, since at most that many pairings are possible. Hence the PMI value is multi-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 163–169,
Seattle, Washington, USA, 18-21 October 2013.

Method	Formula	Revised Formula
PMI (Church and Hanks, 1989)	$log\frac{f(x,y)}{f(x)*f(y)/W}$	
PMI_{sig} (Washtell and Markert, 2009)	$log\frac{f(x,y)}{f(x)*f(y)/W} * \sqrt{\mathbf{min(f(x),f(y))}}$	PMIs: $log\frac{f(x,y)}{f(x)*f(y)/W} * \sqrt{\mathbf{max(f(x),f(y))}}$
cPMId (Damani, 2013)	$log\frac{d(x,y)}{\mathbf{d(x)*d(y)}/D+\sqrt{\mathbf{d(x)}}*\sqrt{\frac{\ln\delta}{(-2.0)}}}$	sPMId: $log\frac{d(x,y)}{\mathbf{max(d(x),d(y))*min(d(x),d(y))}/D+\sqrt{\mathbf{max(d(x),d(y))}}*\sqrt{\frac{\ln\delta}{(-2.0)}}}$

Terminology:

W	Total number of words in the corpus	D	Total number of documents in the corpus
$f(x), f(y)$	unigram frequencies of x, y respectively in the corpus	$d(x), d(y)$	Total number of documents in the corpus containing at least one occurrence of x and y respectively
$f(x,y)$	Span-constrained (x,y) word pair frequency in the corpus	$d(x,y)$	Total number of documents in the corpus having at-least one span-constrained occurrence of the word pair (x,y)
δ	a parameter varying between 0 and 1		

Table 1: Definitions of PMI and its statistically significant adaptations. The sub-parts in bold represent the changes between the original formulas and the revised formulas. The product $max(d(x),d(y)) * min(d(x),d(y))$ in sPMId formula can be simplified to $f(x) * f(y)$, however, we left it this way to emphasize the transformation from cPMId.

plied by $\sqrt{min(f(x),f(y))}$ to get PMI_{sig}.

In (Damani, 2013), statistical significance is introduced by bounding the probability of observing a given number of word-pair occurrences in the corpus, just by chance, under a null model of independent unigram occurrences. For this computation, one needs to decide what constitutes a random trial when looking for a word-pair occurrence. Is it the occurrence of the first word (say x) in the pair, or the second (say y). In (Damani, 2013), occurrences of x are arbitrarily chosen to represent the sites of the random trial. Using Hoeffdings Inequality:

$$P[f(x,y) \geq f(x) * f(y)/W + f(x) * t] \leq \exp(-2 * f(x) * t^2)$$

By setting $t = \sqrt{\ln\delta/(-2 * f(x))}$, we get δ as an upper bound on probability of observing more than $f(x) * f(y)/W + f(x) * t$ bigram occurrences in the corpus, just by chance. Based on this *Corpus Level Significant PMI*(cPMI) is defined as:

$$cPMI(x,y) = log\frac{f(x,y)}{f(x) * f(y)/W + f(x) * t}$$
$$= log\frac{f(x,y)}{f(x) * f(y)/W + \sqrt{f(x)} * \sqrt{\ln\delta/(-2)}}$$

In (Damani, 2013), several variants of cPMI are introduced that incorporate different notions of statistical significance. Of these *Corpus Level Significant PMI based on Document count*(cPMId - defined in Table 1) is found to be the best performing, and hence we consider this variant only in this work.

2.1 Choice of Random Trial

While considering statistical significance, one has to decide what constitutes a random trial. When looking for a word-pair (x,y)'s occurrences, y can potentially occur near each occurrence of x, or x can potentially occur near each occurrence of y. Which of these two set of occurrences should be considered the sites of random trial. We believe that the occurrences of the more frequent of x and y should be considered, since near each of these occurrences the other word could have occurred. Hence $f(x)$ and $f(y)$ in cPMI definition should be replaced with $max(f(x),f(y))$ and $min(f(x),f(y))$ respectively. Similarly, $d(x)$ and $d(y)$ in cPMId formula should be replaced with $max(d(x),d(y))$ and $min(d(x),d(y))$ respectively to give a new measure *Significant PMI based on Document count*(sPMId).

Using the same logic, $\sqrt{min(f(x),f(y))}$ in PMI_{sig} formula should be replaced with $\sqrt{max(f(x),f(y))}$ to give the formula for a new measure *PMI-significant*(PMIs). The definitions of sPMId and PMIs are also given in Table 1.

3 Related Work

There are three main types of word association measures: Knowledge based, Distributional Similarity based, and Lexical Co-occurrence based.

Based on Firth's *You shall know a word by the company it keeps* (Firth, 1957), distributional similarity based measures characterize a word by the distribution of other words around it and compare

Method	Formula
ChiSquare (χ^2)	$\sum_{i,j} \frac{(f(i,j)-Ef(i,j))^2}{Ef(i,j)}$
Dice (Dice, 1945)	$\frac{f(x,y)}{f(x)+f(y)}$
GoogleDistance (L.Cilibrasi and Vitany, 2007)	$\frac{max(\log d(x), \log d(y)) - \log d(x,y)}{\log D - min(\log d(x), \log d(y))}$
Jaccard (Jaccard, 1912)	$\frac{f(x,y)}{f(x)+f(y)-f(x,y)}$
LLR (Dunning, 1993)	$\sum_{\substack{x' \in \{x, \neg x\} \\ y' \in \{y, \neg y\}}} f(x', y') log \frac{f(x', y')}{f(x')f(y')}$
nPMI (Bouma, 2009)	$\frac{log \frac{f(x,y)}{f(x)*f(y)/W}}{log \frac{1}{f(x,y)/W}}$
Ochiai (Janson and Vegelius, 1981)	$\frac{f(x,y)}{\sqrt{f(x)f(y)}}$
PMI^2 (Daille, 1994)	$log \frac{\frac{f(x,y)}{f(x)*f(y)/W}}{\frac{1}{f(x,y)/W}} = log \frac{f(x,y)^2}{f(x)*f(y)}$
Simpson (Simpson, 1943)	$\frac{f(x,y)}{min(f(x), f(y))}$
SCI (Washtell and Markert, 2009)	$\frac{f(x,y)}{f(x)\sqrt{f(y)}}$
T-test	$\frac{f(x,y)-Ef(x,y)}{\sqrt{f(x,y)\left(1-\frac{f(x,y)}{W}\right)}}$

Table 2: Definition of other co-occurrence measures being compared in this work. The terminology used here is same as that in Table 1, except that E in front of a variable name means the expected value of that variable.

Task	Semantic Relatedness	Sentence Similarity	Synonym Selection	
Dataset	WordSim	Li	ESL	TOEFL
Metric	Spearman Rank Correlation	Pearson Correlation	Fraction Correct	Fraction Correct
PMI	<u>0.68</u>	0.69	0.62	0.59
PMI$_{sig}$	0.67	**0.85**	0.58	0.56
cPMId	**0.72**	0.67	0.56	0.59
PMIs	0.66	**0.85**	<u>0.66</u>	**0.61**
sPMId	**0.72**	0.75	**0.70**	**0.61**
ChiSquare (χ^2)	0.62	<u>0.80</u>	0.62	0.58
Dice	0.58	0.76	0.56	0.57
GoogleDistance	0.53	0.75	0.09	0.19
Jaccard	0.58	0.76	0.56	0.57
LLR	0.50	0.18	0.18	0.27
nPMI	**0.72**	0.35	0.54	0.54
Ochiai/ PMI^2	0.62	0.77	0.62	<u>0.60</u>
SCI	0.65	**0.85**	0.62	<u>0.60</u>
Simpson	0.59	0.78	0.58	0.57
TTest	0.44	0.63	0.44	0.52
Semantic Net (Li et al., 2006)		0.82		
ESA (Gabrilovich and Markovitch, 2007)	0.74			
(reimplemented in (Yeh et al., 2009))	0.71			
Distributional Similarity (on web corpus) (Agirre et al., 2009))	0.65			
Context Window based Distributional Similarity (Agirre et al., 2009))	0.60			
Latent Semantic Analysis (on web corpus) (Finkelstein et al., 2002)	0.56			
WordNet::Similarity (Recchia and Jones, 2009)			0.70	0.87
PMI-IR3 (using context) (Turney, 2001)				0.73

Table 3: 5-fold cross-validation results for different co-occurrence measures. The results for the best, and second best co-occurrence measures for each data-set is shown in bold and underline respectively. Except GoogleDistance and LLR, all results for all co-occurrence measures are statistically significant at $p = .05$. For each task, the best known result for different non co-occurrence based methods is also shown.

two words for distributional similarity (Agirre et al., 2009; Wandmacher et al., 2008; Bollegala et al., 2007; Chen et al., 2006). They are also used for modeling the meaning of a phrase or a sentence (Grefenstette and Sadrzadeh, 2011; Wartena, 2013; Mitchell, 2011; G. Dinu and Baroni, 2013; Kartsaklis et al., 2013).

Knowledge-based measures use knowledge-sources like thesauri, semantic networks, or taxonomies (Milne and Witten, 2008; Hughes and Ramage, 2007; Gabrilovich and Markovitch, 2007; Yeh et al., 2009; Strube and Ponzetto, 2006; Finkelstein et al., 2002; Liberman and Markovitch, 2009).

Co-occurrence based measures (Pecina and Schlesinger, 2006) simply rely on unigram and bigram frequencies of the words in a pair. In this work, our focus is on the co-occurrence based measures, since they are resource-light and can easily be used for resource-scarce languages.

3.1 Co-occurrence Measures being Compared

Co-occurrence based measures of association between two entities are used in several domains like ecology, psychology, medicine, language processing, etc. To compare the performance of our newly introduced measures with other co-occurrence measures, we have selected a number of popular co-occurrence measures like ChiSquare (χ^2), Dice (Dice, 1945), GoogleDistance (L.Cilibrasi and Vitany, 2007), Jaccard (Jaccard, 1912), LLR (Dunning, 1993), Simpson (Simpson, 1943), and T-test from these domains.

In addition to these popular measures, we also experiment with other known variations of PMI like nPMI (Bouma, 2009), PMI^2 (Daille, 1994), Ochiai (Janson and Vegelius, 1981), and SCI (Washtell and Markert, 2009). Since PMI^2 is a monotonic transformation of Ochiai, we present their results together. In Table 2, we present the definitions of these measures. While the motivation given for SCI in (Washtell and Markert, 2009) is slightly different, in light of the discussion in Section 2.1, we can assume that SCI is PMI adapted for statistical significance (multiplied by $\sqrt{\mathbf{f(y)}}$), where the site of random trial is taken to be the occurrences of the second word y, instead of the less frequent word, as in the case of PMI_{sig}.

When counting co-occurrences, we only consider the non-overlapping *span*-constrained occurrences. The span of a word-pair's occurrence is the direction-independent distance between the occurrences of the members of the pair. We consider only those co-occurrences where span is less than a given threshold. Therefore, span threshold is a parameter for all the co-occurrence measures being considered.

4 Performance Evaluation

Having introduced the revised measures PMIs and sPMId, we need to evaluate the performance of these measures compared to PMI and the original measures introducing significance. In addition, we also wish to compare the performance of these measures with other co-occurrence measures. To compare the performance of these measures with more resource heavy non co-occurrence based measures, we have chosen those tasks and datasets on which published results exist for distributional similarity and knowledge based word association measures.

4.1 Task Details

We evaluate these measures on three tasks: Sentence Similarity(65 sentence-pairs from (Li et al., 2006)), Synonym Selection(50 questions ESL (Turney, 2001) and 80 questions TOEFL (Landauer and Dutnais, 1997) datasets), and, Semantic Relatedness(353 words Wordsim (Finkelstein et al., 2002) dataset).

For each of these tasks, gold standard human judgment results exist. For sentence similarity, following (Li et al., 2006), we evaluate a measure by the Pearsons correlation between the ranking produced by the measure and the human ranking. For synonym selection, we compute the percentage of correct answers, since there is a unique answer for each challenge word in the datasets. Semantic relatedness has been evaluated by Spearman's rank correlation with human judgment instead of Pearsons correlation in literature and we follow the same practice to make results comparable.

For sentence similarity detection, the algorithm used by us (Li et al., 2006) assumes that the association scores are between 0 and 1. Hence we normalize the value produced by each measure using

Challenge x	Option y (correct)	Option z (incorrect)	$f(x)$	$f(y)$	$f(z)$	$f(x,y)$	$f(x,z)$	PMI$_{sig}$ (x,y)	PMI$_{sig}$ (x,z)	PMIs (x,y)	PMIs (x,z)
brass	metal	plastic	15923	125088	24985	228	75	14	24	40	30
twist	intertwine	curl	11407	153	2047	1	9	7	17	61	41
saucer	dish	frisbee	2091	12453	1186	5	1	9	14	21	18
mass	lump	element	90398	1595	43321	14	189	4	10	29	15
applause	approval	friends	1998	19673	11689	8	6	9	11	29	28
confession	statement	plea	7687	47299	5232	76	12	18	22	45	26
swing	sway	bounce	33580	2994	4462	13	17	7	8	24	21
sheet	leaf	book	20470	20979	586581	20	194	7	2	7	12

Table 4: Details of ESL word-pairs, correctness of whose answers changes between PMI$_{sig}$ and PMIs. Except for the gray-row, for all other questions, incorrect answers becomes correct on using PMIs instead of PMI$_{sig}$, and vice-versa for the gray-row. The association values have been suitably scaled for readability. To save space, of the four choices, options not selected by either of the methods have been omitted. These results are for a 10 word span.

max-min normalization:

$$v' = \frac{v - min}{max - min}$$

where *max and* min are computed over all association scores for the entire task for a given measure.

4.2 Experimental Results

We use a 1.24 Gigawords Wikipedia corpus for getting co-occurrence statistics. Since co-occurrence methods have span-threshold as a parameter, we follow the standard methodology of five-fold cross validation. Note that, in addition to span-threshold, cPMId and sPMId have an additional parameter δ.

In Table 3, we present the performance of all the co-occurrence measures considered on all the tasks. Note that, except GoogleDistance and LLR, all results for all co-occurrence measures are statistically significant at p = .05. For completeness of comparison, we also include the best known results from literature for different non co-occurrence based word association measures on these tasks.

4.3 Performance Analysis and Conclusions

We find that on average, PMI$_{sig}$ and cPMId, the recently introduced measures that incorporate significance in PMI, do not perform better than PMI on the given datasets. Both of them perform worse than PMI on three out of four datasets. By appropriately incorporating significance, we get new measures PMIs and sPMId that perform better than PMI(also PMI$_{sig}$ and cPMId respectively) on most datasets. PMIs improves performance over PMI on three out of four datasets, while sPMId improves performance on all four datasets.

The performance improvement of PMIs over PMI$_{sig}$ and of sPMId over cPMId, is not random. For example, on the ESL dataset, while the percentage of correct answers increases from 58 to 66 from PMI$_{sig}$ to PMIs, it is not the case that on moving from PMI$_{sig}$ to PMIs, several correct answers become incorrect and an even larger number of incorrect answers become correct. As shown in Table 4, only one correct answers become incorrect while seven incorrect answers get corrected. The same trend holds for most parameters values, and for moving from cPMId to sPMId. This substantiates the claim that the improvement is not random, but due to the appropriate incorporation of significance, as discussed in Section 2.1.

PMIs and sPMId perform better than not just PMI, but they perform better than all popular co-occurrence measures on most of these tasks. When compared with any other co-occurrence measure, on three out of four datasets each, both PMIs and sPMId perform better than that measure. In fact, PMIs and sPMId perform reasonably well compared with more resource intensive non co-occurrence based methods as well. Note that different non co-occurrence based measures perform well on different tasks. We are comparing the performance of a single measure (say sPMId or PMIs) against the best measure for each task.

Acknowledgements

We thank Dipak Chaudhari for his help with the implementation.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL-HLT 2009, Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *WWW 2007, The World Wide Web Conference*, pages 757–766.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction, from form to meaning: Processing texts automatically. In *GSCL 2009, Proceedings of the Biennial International Conference of the German Society for Computational Linguistics and Language Technology*.

Hsin-Hsi Chen, Ming-Shun Lin, and Yu-Chuan Wei. 2006. Novel association measures using web search with double checking. In *COLING/ACL 2006, Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information and lexicography. In *ACL 1989, Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 76–83.

B. Daille. 1994. *Approche mixte pour l'extraction automatique de terminologie: statistiques lexicales etltres linguistiques*. Ph.D. thesis, Universitie Paris 7.

Om P. Damani. 2013. Improving pointwise mutual information (pmi) by incorporating significant cooccurrence. In *CoNLL 2013, Conference on Computational Natural Language Learning*.

L. R. Dice. 1945. Measures of the amount of ecological association between species. *Ecology*, 26:297–302.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

J. R. Firth. 1957. A synopsis of linguistics theory. *Studies in Linguistic Analysis*, pages 1930–1955.

N. Pham G. Dinu and M. Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *CVSC 2013, Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI 2007, International Joint Conference on Artificial Intelligence*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *EMNLP 2011, Conference on Empirical Methods on Natural Language Processing*, pages 1394–1404.

T Hughes and D Ramage. 2007. Lexical semantic relatedness with random graph walks. In *EMNLP 2007, Conference on Empirical Methods on Natural Language Processing*.

P. Jaccard. 1912. The distribution of the flora of the alpine zone. *New Phytologist*, 11:37–50.

Svante Janson and Jan Vegelius. 1981. Measures of ecological association. *Oecologia*, 49:371–376.

Dimitrios Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *CoNLL 2013, Conference on Computational Natural Language Learning*.

Thomas K Landauer and Susan T. Dutnais. 1997. A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240.

Rudi L.Cilibrasi and Paul M.B. Vitany. 2007. The google similarity distance. *Psychological review*, 19(3).

Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1138–1150, August.

Sonya Liberman and Shaul Markovitch. 2009. Compact hierarchical explicit semantic representation. In *WikiAI 2009, Proceedings of the IJCAI Workshop on User-Contributed Knowledge and Artificial Intelligence: An Evolving Synergy*, Pasadena, CA, July.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *ACL 2008, Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jeffrey Mitchell. 2011. *Composition in Distributional Models of Semantics*. Ph.D. thesis, The University of Edinburgh.

Pavel Pecina and Pavel Schlesinger. 2006. Combining association measures for collocation extraction. In *ACL 2006, Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Gabriel Recchia and Michael N. Jones. 2009. More data trumps smarter algorithms: Comparing pointwise mutual information with latent semantic analysis. *Behavior Research Methods*, 3(41):647–656.

George G. Simpson. 1943. Mammals and the nature of continents. *American Journal of Science*, pages 1–31.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI 2006, Conference on Artificial Intelligence*, pages 1419–1424.

P. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *ECML 2001, European Conference on Machine Learning*.

T. Wandmacher, E. Ovchinnikova, and T. Alexandrov. 2008. Does latent semantic analysis reflect human associations? In *ESSLLI 2008, European Summer School in Logic, Language and Information*.

Christian Wartena. 2013. Hsh: Estimating semantic similarity of words and short phrases with frequency normalized distance measures. In *SemEval 2013, International Workshop on Semantic Evaluation*.

Justin Washtell and Katja Markert. 2009. A comparison of windowless and window-based computational association measures as predictors of syntagmatic human associations. In *EMNLP 2009, Conference on Empirical Methods on Natural Language Processing*, pages 628–637.

Eric Yeh, Daniel Ramage, Chris Manning, Eneko Agirre, and Aitor Soroa. 2009. Wikiwalk: Random walks on wikipedia for semantic relatedness. In *TextGraphs 2009, Proceedings of the ACL workshop on Graph-based Methods for Natural Language Processing*.

Growing Multi-Domain Glossaries from a Few Seeds using Probabilistic Topic Models

Stefano Faralli and **Roberto Navigli**
Dipartimento di Informatica
Sapienza Università di Roma
{faralli,navigli}@di.uniroma1.it

Abstract

In this paper we present a minimally-supervised approach to the multi-domain acquisition of wide-coverage glossaries. We start from a small number of hypernymy relation seeds and bootstrap glossaries from the Web for dozens of domains using Probabilistic Topic Models. Our experiments show that we are able to extract high-precision glossaries comprising thousands of terms and definitions.

1 Introduction

Dictionaries, thesauri and glossaries are useful sources of information for students, scholars and everyday readers, who use them to look up words of which they either do not know, or have forgotten, the meaning. With the advent of the Web an increasing number of dictionaries and technical glossaries has been made available online, thereby speeding up the definition search process. However, finding definitions is not always immediate, especially if the target term pertains to a specialized domain. Indeed, not even well-known services such as Google Define are able to provide definitions for scientific or technical terms such as *taxonomy learning* or *distant supervision* in AI or *figure-four leglock* and *suspended surfboard* in wrestling.

Domain-specific knowledge of a definitional nature is not only useful for humans, it is also useful for machines (Hovy et al., 2013). Examples include Natural Language Processing tasks such as Question Answering (Cui et al., 2007), Word Sense Disambiguation (Duan and Yates, 2010; Faralli and Navigli, 2012) and ontology learning (Velardi et al., 2013). Unfortunately, most of the Web dictionaries and glossaries available online comprise just a few hundred definitions, and they therefore provide only a partial view of a domain. This is also the case with manually compiled glossaries created by means of collaborative efforts, such as Wikipedia.[1] The coverage issue is addressed by online aggregation services such as Google Define, which bring together definitions from several online dictionaries. However, these services do not classify textual definitions by domain: they just present the collected definitions for all the possible meanings of a given term.

In order to automatically obtain large domain glossaries, in recent years computational approaches have been developed which extract textual definitions from corpora (Navigli and Velardi, 2010; Reiplinger et al., 2012) or the Web (Velardi et al., 2008; Fujii and Ishikawa, 2000). The methods involving corpora start from a given set of terms (possibly automatically extracted from a domain corpus) and then harvest textual definitions for these terms from the input corpus using a supervised system. Web-based methods, instead, extract text snippets from Web pages which match pre-defined lexical patterns, such as "X is a Y", along the lines of Hearst (1992). These approaches typically perform with high precision and low recall, because they fall short of detecting the high variability of the syntactic structure of textual definitions. To address the low-recall issue, recurring cue terms occurring

[1] See http://en.wikipedia.org/wiki/Portal:Contents/Glossaries

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 170–181,
Seattle, Washington, USA, 18-21 October 2013.

within dictionary and encyclopedic resources can be automatically extracted and incorporated into lexical patterns (Saggion, 2004). However, this approach is term-specific and does not scale to arbitrary terminologies and domains.

The goal of the new approach outlined in this paper is to enable the automatic harvesting of large-scale, full-fledged domain glossaries for dozens of domains, an outcome which should be very useful for both human activities and automatic tasks. We present ProToDoG (Probabilistic Topics for multi-Domain Glossaries), a framework for growing multi-domain glossaries which has three main novelties:

i) **minimal human supervision:** a small set of hypernymy relation seeds for each domain is used to bootstrap the multi-domain acquisition process;

ii) **jointness:** our approach harvests terms and glosses at the same time;

iii) **probabilistic topic models** are leveraged for a simultaneous, high-precision multi-domain classification of the extracted definitions, with substantial performance improvements over our previous work on glossary bootstrapping, i.e., GlossBoot (De Benedictis et al., 2013).

ProToDog is able to harvest definitions from the Web and thus drop the requirement of large corpora for each domain. Moreover, apart from the need to select a few seeds, it avoids the use of training data or manually defined sets of lexical patterns. It is thus applicable to virtually any language of interest.

2 ProToDoG

Given a set of domains $D = \{d_1, ..., d_n\}$, for each domain $d \in D$ ProToDoG harvests a domain glossary G_d containing pairs of the kind (t, g) where t is a domain term and g is its textual definition, i.e., gloss. We show the pseudocode of ProToDoG in Algorithm 1.

Step 1. Initial seed selection: Algorithm 1 takes as input a set of domains D and, for each domain $d \in D$, a small set of hypernymy relation seeds $S_d = \{(t_1, h_1), \ldots, (t_{|S_d|}, h_{|S_d|})\}$, where the seed pair (t_j, h_j) contains a term t_j and its generalization h_j (e.g., (*linux*, *operating system*)). This is the only human input to the entire glossary acquisition process. The selection of the input seeds plays a key role in the bootstrapping process, in that the pattern and gloss extraction process will be driven by them. The chosen hypernymy relations thus have to be as topical and representative as possible for the domain of interest (e.g., (*compiler*, *computer program*) is an appropriate pair for computer science, while (*byte*, *unit of measurement*) is not, as it might cause the extraction of out-of-domain glossaries of units and measures).

Algorithm 1 ProToDoG

Input: the set of domains D,
a set S_d of hypernymy seeds for each domain $d \in D$
Output: a multi-domain glossary G

1: $k \leftarrow 1$
2: **repeat**
3: **for each** domain $d \in D$ **do**
4: $G_d^k \leftarrow \emptyset$
5: **for each** seed $(t_j, h_j) \in S_d$ **do**
6: $pages \leftarrow webSearch(t_j, h_j, \text{“}glossary\text{”})$
7: $G_d^k \leftarrow G_d^k \cup extractGlossary(pages)$
8: **end for**
9: **end for**
10: *create* a topic model using glossaries from previous iterations
11: *infer* topic assignments for iteration-k glosses
12: *filter out* non-domain glosses for each domain
13: **for each** $d \in D$ **do**
14: $S_d \leftarrow seedSelectionForNextIteration(G_d^k)$
15: **end for**
16: $k \leftarrow k + 1$
17: **until** $k > max$
18: **for each** domain $d \in D$ **do**
19: *recover* filtered glosses into G_d^{max+1}
20: $G_d \leftarrow \bigcup_{j=1,\ldots,max+1} G_d^j$
21: **end for**
22: **return** $G = \{(G_d, d) : d \in D\}$

The algorithm first sets the iteration counter k to 1 (line 1) and starts the first iteration of the glossary bootstrapping process (lines 2-17), each involving steps 2-4, described below. After each iteration k, for each domain d we keep track of the set of glosses G_d^k acquired during that iteration. After the last iteration, we perform step (5) of gloss recovery (lines 18-21).

Step 2. Web search and glossary extraction (lines 3-9): For each domain d, we first initialize the domain glossary for iteration k: $G_d^k := \emptyset$ (line 4). Then, for each seed pair $(t_j, h_j) \in S_d$, we submit the following query to a Web search engine: "t_j" "h_j" `glossary` and collect the top-ranking results for each query (line 6).[2] Each resulting page is a candidate glossary for the domain d.

We then call the $extractGlossary$ function (line 7) which extracts terms and glosses from the retrieved pages as follows. From each candidate page, we harvest all the text snippets s starting with t_j and ending with h_j (e.g., "*linux* – an *operating system*"), i.e., $s = t_j \ldots h_j$. For each such text snippet s, we extract the following pattern instance:

$$p_L\ t_j\ p_M\ gloss_s(t_j)\ p_R,$$

where:

- p_M is the longest sequence of HTML tags and non-alphanumeric characters between t_j and the glossary definition (e.g., " –" between "linux" and "an" in the above example);
- $gloss_s(t_j)$ is the gloss of t_j obtained by moving to the right of p_M until we reach a non-formatting tag element (e.g., , ,), while ignoring formatting elements such as , and <a> which are typically included within a definition sentence;
- p_L and p_R are the longest sequences of HTML tags on the left of t_j and the right of $gloss_s(t_j)$, respectively.

For instance, given the HTML snippet "... linux – an operating system developed by Linus Torvalds ..." we extract the following pattern instance: p_L = " ", t_j = "*linux*", p_M = " –", $gloss_s(t_j)$ = "an *operating system* developed by Linus Torvalds", p_R =" ".

Then we generalize the above pattern instance by replacing t_j and $gloss_s(t_j)$ with *, obtaining:

$$p_L\ *\ p_M\ *\ p_R,$$

For the above example, we obtain the following pattern:

 * – * .

We add the first sentence of the retrieved gloss $gloss_s(t_j)$ to our glossary G_d^k, i.e., $G_d^k := G_d^k \cup \{(t_j, first(gloss_s(t_j)))\}$, where $first(g)$ returns the first sentence of gloss g. Finally, we look for additional pairs of terms/glosses in the Web page containing the snippet s by matching the page against the generalized pattern $p_L\ *\ p_M\ *\ p_R$, and adding them to G_d^k.

As a result of step (2), for each domain $d \in D$ we obtain a glossary G_d^k for the terms discovered at iteration k.

Step 3. Topic modeling and gloss filtering (lines 10-12): Unfortunately, not all (term, gloss) pairs in a glossary G_d^k will pertain to the domain d. For instance, we might end up retrieving interdisciplinary or even unrelated glossaries. In order to address this fuzziness, we model domains with a Probabilistic Topic Model (PTM) (Blei et al., 2003; Steyvers and Griffiths, 2007). PTMs model a given text document as a mixture of topics. In our case topics are domains and we, first, create a topic model from the domain glossaries acquired before the current iteration k, then, second, use the topic model to estimate the domain assignment of each new pair (term, gloss) in our glossaries G_d^k, i.e., obtained at iteration k, third, filter out non-domain glosses.

Creating the topic model (line 10): For a given iteration k and domain d, we first define the terminology accumulated up until iteration $k-1$ for that domain as the set $T_d^{1,k-1} := \bigcup_{j=1}^{k-1} T_d^j$, where T_d^j is the set of terms acquired at iteration j, i.e., $T_d^j := \{t : \exists (t, g) \in G_d^j\}$.[3] Then we define:

- $W := \bigcup_{d \in D} T_d^{1,k-1}$ as the entire terminology acquired up until iteration $k-1$ for all domains, i.e., the full set of terms independently of their domain;
- $M := \bigcup_{d \in D} \bigcup_{j=1}^{k-1} G_d^j$ as the multi-domain glossary acquired up until iteration $k-1$, i.e., the full set of pairs (term, gloss) independently of their domain;[4]

[2] We use the Google Ajax API, which returns the 64 top-ranking search results.

[3] For the first iteration, i.e., when $k = 1$, we define $T_d^{1,0} := \{t : \exists (t, g) \in G_d^1\}$, i.e., we use the terminology resulting from step (2) of the first iteration.

[4] For $k = 1$, $M := \bigcup_{d \in D} G_d^1$.

- Two count matrices, i.e., the word-domain matrix C^{WD} and the gloss-domain matrix C^{MD}, such that: $C^{WD}_{w,d}$ counts the number of times $w \in W$ is assigned to domain $d \in D$, i.e., it occurs in the glosses of domain d; $C^{MD}_{(t,g),d}$ counts the number of words in g assigned to domain d.

At this point, as shown by Steyvers and Griffiths (2007), we can estimate the probability $\phi^{(d)}_w$ for word w, and the probability $\theta^{(t,g)}_d$ for a term/gloss pair (t, g), of belonging to domain d:

$$\phi^{(d)}_w = \frac{C^{WD}_{w,d} + \beta}{\sum_{w'=1}^{|W|} C^{WD}_{w',d} + |W|\beta}; \quad (1)$$

$$\theta^{(t,g)}_d = \frac{C^{MD}_{(t,g),d} + \alpha}{\sum_{d'=1}^{|D|} C^{MD}_{(t,g),d'} + |D|\alpha} \quad (2)$$

where α and β are smoothing factors.[5] The two above probabilities represent the core of our topic model of the domain knowledge acquired up until iteration $k-1$.

Probabilistic modeling of iteration-k glosses (line 11): We now utilize the above topic model to estimate the probabilities in Formulas 1 and 2 for the newly acquired glosses at iteration k. To this end we define $M' := \bigcup_{d \in D} G^k_d$ as the union of the (term, gloss) pairs at iteration k and $W' := \bigcup_{d \in D} T^k_d \bigcap W$ as the union of terms acquired at iteration k, but also occurring in W (i.e., the entire terminology up until iteration $k-1$). Then we apply Gibbs sampling (Blei et al., 2003; Phan et al., 2008) to estimate the probability of each pair $(t, g) \in M'$ of pertaining to a domain d by computing:

$$\theta'^{(t,g)}_d = \frac{R^{M'D}_{(t,g),d} + \alpha}{\sum_{d'=1}^{|D|} R^{M'D}_{(t,g),d'} + |D|\alpha} \quad (3)$$

where the gloss-domain matrix $R^{M'D}$ is initially defined by counting random domain assignments for each word w' in the bag of words of each (term, gloss) pair $\in M'$. Next, the domain assignment counts in $R^{M'D}$ are iteratively updated using Gibbs sampling.[6]

Filtering out non-domain glosses (line 12): Now, for each domain $d \in D$, for each pair $(t, g) \in G^k_d$ we have a probability $\theta'^{(t,g)}_d$ of belonging to d. We mark (t, g) as a non-domain item if $\theta'^{(t,g)}_d < \delta$, where δ is a confidence threshold, or if $\theta'^{(t,g)}_d$ is not maximum among all domains in D. Non-domain pairs are removed from G^k_d and stored into a set A_d for possible recovery after the last iteration (see step (5)).

Step 4. Seed selection for next iteration (lines 13-15): For each domain $d \in D$, we now select the new set of hypernymy relation seeds to be used to start the next iteration. First, for each newly-acquired term/gloss pair $(t, g) \in G^k_d$, we automatically extract a candidate hypernym h from the textual gloss g. To do this we use a simple heuristic which just selects the first content term in the gloss.[7] Then we sort all the glosses in G^k_d by the number of seed terms found in each gloss. In the case of ties (i.e., glosses with the same number of seed terms), we further sort the glosses by $\theta'^{(t,g)}_d$. Finally we select the (term, hypernym) pairs corresponding to the $|S_d|$ top-ranking glosses as the new set of seeds for the next iteration.

Next, we increment k (line 16 of Algorithm 1) and if the maximum number of iterations is reached we jump to step (5). Otherwise, we go back to step (2) of our glossary bootstrapping algorithm with the new set of seeds S_d.

Step 5. Gloss recovery (line 19): After all iterations, the entire multi-domain terminology W (cf. step (3)) may contain several new terms which were not present when a given gloss g was filtered out. So, thanks to the last-iteration topic model, the gloss g might come back into play because its words are now important cues for a domain. To reassess the domain pertinence of (term, gloss) pairs in A_d for each d, we just reapply the entire step (3) by setting $G^{max+1}_d := A_d$ for each $d \in D$. As a result, we

[5] As experienced by Steyvers and Griffiths (2007), the values of $\alpha = 50/|D|$ and $\beta = 0.01$ work well with many different text collections.

[6] For the PTM part of ProToDoG we used the JGibbLDA library, a Java Implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling for Parameter Estimation and Inference, available at: `http://jgibblda.sourceforge.net/`

[7] While more complex strategies could be devised, e.g., lattice-based hypernym extraction (Navigli and Velardi, 2010), we found that this heuristic works well because, even when it is not a hypernym, the first term acts as a cue word for the defined term.

obtain an updated glossary G_d^{max+1} which contains all the recovered glosses.

Final output: For each domain $d \in D$ the final output of ProToDoG is a domain glossary $G_d := \bigcup_{j=1,...,max+1} G_d^j$. Finally the algorithm aggregates all glossaries G_d into a multi-domain glossary G (line 22).

3 Experimental Setup

3.1 Domains

For our experiments we selected 30 different domains ranging from Arts to Warfare, mostly following the domain classification of Wikipedia featured articles (full list at `http://lcl.uniroma1.it/protodog`). The set includes several technical domains, such as Chemistry, Geology, Meteorology, Mathematics, some of which are highly interdisciplinary. For instance, the Environment domain covers terms from fields such as Chemistry, Biology, Law, Politics, etc.

3.2 Gold Standard

Since our evaluations required considerable human effort, in what follows we calculated all performances on a random set of 10 domains, shown in the top row of Table 1. For each of these 10 domains we selected well-reputed glossaries on the Web as gold standards, including the Reuters glossary of finance, the Utah computing glossary and many others (full list at the above URL). We show the size of our 10 gold-standard datasets in Table 1.

3.3 Evaluation measures

We evaluated the quality of both terms and glosses, as jointly extracted by ProToDoG.

3.3.1 Terms

For each domain we calculated coverage, extra-coverage and precision of the acquired terms T. **Coverage** is the ratio of extracted terms in T also contained in the gold standard $\hat{T}$ over the size of $\hat{T}$. **Extra-coverage** is calculated as the ratio of the additional extracted terms in $T \setminus \hat{T}$ over the number of gold standard terms $\hat{T}$. Finally, **precision** is the ratio of extracted terms in T deemed to be within the domain. To calculate precision we randomly sampled 5% of the retrieved terms and asked two human annotators to manually tag their domain pertinence (with adjudication in case of disagreement; $\kappa = .62$, indicating substantial agreement). Note that by randomly sampling on the entire set T we calculate the precision of both terms in $T \cap \hat{T}$, i.e., in the gold standard, and terms in $T \setminus \hat{T}$, i.e., not in the gold standard, but which are not necessarily outside the domain.

3.3.2 Glosses

We calculated the precision of the extracted glosses as the ratio of glosses which were both well-formed textual definitions and specific to the target domain. Precision was determined on a random sample of 5% of the acquired glosses for each domain. The annotation was made by two annotators, with $\kappa = .675$, indicating substantial agreement. The annotators were provided with specific guidelines available on the ProToDoG Web site (see URL above).

3.4 Comparison

We compared ProToDog against:

- **BoW**: a bag-of-words variant in which step (3) is replaced by a simple bag-of-words scoring approach which assigns a score to each term/gloss pair $(t, g) \in G_d^k$ as follows:

$$score(g) = \frac{|Bag(g) \cap T_d^{1,k-1}|}{|Bag(g)|}. \quad (4)$$

  where $Bag(g)$ contains all content words in g. At iteration k, we filter out those glosses whose $score(g) < \sigma$, where σ is a threshold tuned in the same manner as δ (see Section 3.5). This approach essentially implements GlossBoot, our previous work on domain glossary bootstrapping (De Benedictis et al., 2013).

- **Wikipedia**: since Wikipedia is the largest collaborative resource, covering hundreds of fields of knowledge, we devised a simple heuristic for producing multi-domain glossaries from Wikipedia, so as to compare their performance against our gold standards. For each target domain we manually selected one

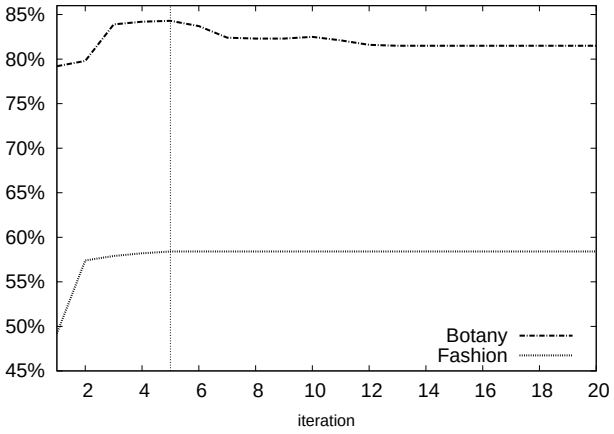


Figure 1: Harmonic mean of precision and coverage for Botany and Fashion (tuning domains) over 20 iterations ($|S_d|$=5, δ=0.03).

or more Wikipedia categories representing the domain (for instance, `Category:Arts` for Arts, `Category:Business` for Finance, etc.). Then, for each domain d, we picked out all the Wikipedia pages tagged either with the categories selected for d or their direct subcategories (e.g., `Category:Creative works`) or subsubcategories (e.g., `Category:Genres`). From each page we extracted a (page title, gloss) pair, where the gloss was obtained by extracting the first sentence of the Wikipedia page, as done, e.g., in BabelNet (Navigli and Ponzetto, 2012). Since subcategories might have more parents and might thus belong to multiple domains, we discarded pages assigned to more than 2 domains.

3.5 Parameter tuning

In order to choose the optimal values of the parameters of ProToDoG (number $|S_d|$ of seeds per domain, number max of iterations, and filtering threshold δ) and BoW (σ threshold) we selected two extra domains, i.e., Botany and Fashion, not used in our tests, together with the corresponding gold standard Web glossaries.

As regards the number of seeds, we defined an initial pool of 10 seeds for each of the two tuning domains and studied the average performance of 5 random sets of x seeds (from the initial pool), when $x = 1, 3, 5, 7, 9$. As regards the number of iterations, we explored all values between 1 and 20. Finally, for the filtering thresholds δ and σ for ProToDoG PTM and its BoW variant, we tried values of $\delta \in \{0, 0.03, 0.06, \ldots, 0.6\}$ and $\sigma \in \{0, 0.05, 0.1, \ldots, 1.0\}$, respectively.

Given the high number of possible parameter value configurations, we first explored the entire search space automatically by calculating the coverage of ProToDoG PTM (and BoW) with each configuration against our tuning gold standards. Then we identified as optimal candidates those "frontier" configurations for which, when moving from a lower-coverage configuration, coverage reached a maximum. We then calculated the precision of each optimal candidate configuration by manually validating a 3% random sample of the resulting glossaries for the two tuning domains. The optimal configuration for ProToDoG was $|S_d| = 5$, $max = 5$, $\delta = 0.03$, while for BoW was $\sigma = 0.1$.

In Figure 1 we show the performance trend over iterations for our two tuning domains when $|S_d| = 5$ and $\delta = 0.03$. Performance is calculated as the harmonic mean of precision and coverage of the acquired glossary after each iteration, from 1 to 20. We can see that after 5 iterations performance decreases for Botany (a highly interdisciplinary domain) due to lower precision, while it remains stable for Fashion due to the lack of newly-acquired glosses.

3.6 Seed Selection

For each domain d we manually selected five seed hypernymy relations as the seed sets S_d input to Algorithm 1 (see Section 3.5). The seeds were selected by the authors on the basis of just two conditions: i) the seeds should cover different aspects of the domain and, indeed, should identify the domain implicitly; ii) at least 10,000 results should be returned by the search engine when querying it with the seeds plus the `glossary` keyword (see line 6 of Algorithm 1). The seed selection was not fine-tuned (i.e., it was not adjusted to improve performance), so it might well be that better seeds would provide better results (see (Kozareva and Hovy, 2010a)). However, such a study is beyond the scope of this paper.

		Art	Business	Chemistry	Computing	Environment	Food	Law	Music	Physics	Sport
Gold	t/g	394	1777	164	421	713	946	180	218	315	146
PTM	t	4253	7370	2493	3412	3009	1526	1836	1647	3847	1696
	g	7386	9795	3841	4186	3552	2175	4141	2729	5197	2938
BoW	t	4012	7639	1174	3127	3644	1827	1773	1166	4471	1990
	g	5923	8999	1414	3662	4334	2601	4024	1249	6956	3425
Wiki	t,g	107.1k	48.4k	8137	32.0k	23.6k	5698	13.5k	84.1k	33.8k	267.5k

Table 1: Size of the gold standard and the automatically-acquired glossaries for 10 of the 30 selected domains (t: number of terms, g: number of glosses).

4 Results and Discussion

4.1 Terms

The size of the extracted terminologies for the 10 domains after five iterations is reported in Table 1 (the output for all 30 domains is available at the above URL, cf. Section 3.1). ProToDoG PTM and its BoW variant extract thousands of terms and glosses for each domain, whereas the number of glosses obtained from Wikipedia (cf. Section 3.4) varies depending upon the domain, from thousands to hundreds of thousands. Note that there is no overlap between the glossaries extracted by ProToDoG and the set of Wikipedia articles, since the latter are not organized as glossaries.

In Table 2 we show the percentage results in terms of precision (P), coverage (C), and extra-coverage (X, see Section 3.3 for definitions) for ProToDoG PTM and its BoW variant and for the Wikipedia glossary. With the exception of the Food domain, ProToDoG achieves the best precision. The Wikipedia glossary has fluctuating precision values, ranging between 25% and 90%, due to the heterogeneous nature of subcategories. ProToDog achieves the best coverage of gold standard terms on 6 of the 10 domains, with the BoW variant obtaining slightly higher coverage on 3 domains and +10% on the Food domain. The coverage of Wikipedia glossaries, instead, with the sole exception of Sport, is much lower, despite the use of (sub)subcategories (cf. Section 3.4). Both ProToDoG PTM and BoW achieve very high extra-coverage percentages, meaning that they are able to go substantially beyond our domain gold standards, but it is the Wikipedia glossary which achieves the highest extra-coverage values. To get a better insight into the quality of extra-coverage we calculated the percentage of named entities (i.e., encyclopedic) among the terms extracted by each of the different approaches. Comparing results across the (E) columns of Table 2 it can be seen that high percentages of the terms extracted by Wikipedia are named entities, which is in marked contrast to the 0%-1% extracted by ProToDog. This is as should be expected for an encyclopedia, whose coverage focuses on people, places, brands, etc. rather than concepts.

To summarize, ProToDoG PTM outperforms both BoW and Wikipedia in terms of precision, while at the same time achieving both competitive coverage and extra-coverage. The Wikipedia glossary suffers from fluctuating precision values across domains and overly encyclopedic coverage of terms.

4.2 Glosses

We show the results of gloss evaluation in Table 2 (last two columns) for ProToDoG PTM and BoW (we do not report the precision values for Wikipedia, as they are slightly lower than those obtained for terms). Precision ranges between 89% and 99% for ProToDoG PTM and between 82% and 97% for BoW. We observe that these results are strongly correlated with the precision of the extracted terms (cf. Table 2), because the retrieved glosses of domain terms are usually in-domain too, and follow a definitional style since they come from glossaries. Note, however, that the gloss precision could also be

	terms												glosses	
	PTM				BoW				Wiki				PTM	BoW
	P	C	X	E	P	C	X	E	P	C	X	E	P	P
Art	**92**	**26**	1053	1	86	25	992	0	81	19	23.4k	67	**93**	87
Business	**95**	41	374	0	90	**43**	387	0	37	15	2692	31	**96**	91
Chemistry	**99**	**77**	1410	0	95	73	643	0	49	18	12.9k	3	**98**	96
Computing	**95**	**43**	767	0	93	40	702	0	81	30	7506	36	**96**	94
Environment	**91**	**29**	393	0	84	28	482	0	25	9	3302	12	**89**	82
Food	91	21	1404	0	**97**	**31**	1621	0	81	9	3997	25	92	**95**
Law	**98**	**89**	931	0	95	87	897	0	35	34	7406	16	**99**	97
Music	**94**	**98**	660	0	93	84	453	0	90	50	37.1k	84	**96**	95
Physics	**97**	43	1178	0	91	**46**	1373	0	68	25	10.6k	10	**95**	89
Sport	**98**	22	1139	1	96	**23**	1339	1	87	44	178.2k	83	**97**	96

Table 2: Precision (P), coverage (C), extra-coverage (X), encyclopedic (E) percentages after 5 iterations.

	Art	Business	Chemistry	Computing	Environment	Food	Law	Music	Physics	Sport
Google Define	76	80	93	86	88	91	96	96	98	84
ProToDoG	27	41	81	40	37	19	85	98	47	27

Table 3: Number of domain glosses (from a random sample of 100 gold standard terms per domain) retrieved using Google Define and ProToDoG.

higher than term precision, thanks to many pertinent glosses being extracted for the same term (cf. Table 1).

In Table 4 we show an excerpt of the multi-domain glossary extracted by ProToDoG for the Art, Business and Sport domains.

5 Comparative Evaluation

5.1 Comparison with Google Define

We performed a comparison with Google Define,[8] a state-of-the-art definition search service. This service inputs a term query and outputs a list of glosses. First, we randomly sampled 100 terms from our gold standard for each domain. Next, for each domain, we manually calculated the fraction of terms for which at least one in-domain definition was provided by Google Define and ProToDoG. Table 3 shows the coverage results. In this experiment, Google Define outperforms our system on 9 of the 10 analyzed domains. However, we note that when searching for domain-specific knowledge only, Google Define: i) needs to know the domain term to be defined in advance, while ProToDoG jointly acquires domain terms and glosses starting from just a few seeds; ii) does not discriminate between glosses pertaining to the target domain and glosses pertaining to other fields or senses, whereas ProToDog extracts terms and glosses specific to each domain of interest.

5.2 Comparison with TaxoLearn

We also compared ProToDoG with the output of a state-of-the-art taxonomy learning framework, called TaxoLearn (Navigli et al., 2011). We did this because i) TaxoLearn extracts terms and glosses from domain corpora in order to create a domain taxonomy; ii) it is one of the few systems which extracts both terms and glosses from specialized corpora; iii) the extracted glossaries are available online.[9] Therefore we compared the performance of ProToDoG on two domains for which glossaries were extracted by TaxoLearn, i.e. AI and Finance. The glossaries were harvested from large collections of scholarly articles. For ProToDoG we selected 10 seeds to cover all the fields of AI, while for the financial domain we selected the same 5 seeds used in the Business

[8] Accessible from Google search with the `define:` keyword.

[9] `http://ontolearn.org` and `http://lcl.uniroma1.it/taxolearn`

Art	
rock art	includes pictographs (designs painted on stone surfaces) and petroglyphs (designs pecked or incised on stone surfaces).
impressionism	Late 19th-century French school dedicated to defining transitory visual impressions painted directly from nature, with light and color of primary importance.
point	Regarding paper, a unit of thickness equating 1/1000 inch.
Business	
hyperinflation	Extremely rapid or out of control inflation.
interbank rate	The rate of interest charged by a bank on a loan to another bank.
points	Amount of discount on a mortgage loan stated as a percentage; one point equals one percent of the face amount of the loan; a discount of one point raises the net yield on the loan by one-eighth of one percent.
Sport	
gross score	The actual number of strokes taken by a player for hole or round before the player's handicap is deducted.
obstructing	preventing the opponent from going around a player by standing in the path of movement.
points	a team statistic indicating its degree of success, calculated as follows: 2 points for a win (3 in the 1994 World Cup), 1 point for a tie, 0 points for a loss.

Table 4: An excerpt of the resulting multi-domain glossary obtained with ProToDoG.

domain of our experiments above (cf. Section 3).

We show the number of extracted terms and glosses for ProToDoG and TaxoLearn in Table 5. We also show the precision values calculated on a random sample of 5% of terms and glosses. As can be clearly seen, on both domains ProToDoG extracts a number of terms and glosses which is an order of magnitude greater than those obtained by TaxoLearn, while at the same time obtaining considerably higher precision.

6 Related Work

Current approaches to automatic glossary acquisition suffer from two main issues: i) the poor availability of large domain-specific corpora from which terms and glosses are extracted at different times; ii) the focus on individual domains. ProToDog addresses both issues by providing a joint multi-domain approach to term and glossary extraction.

Among the approaches which extract unrestricted textual definitions from open text, Fujii and Ishikawa (2000) determine the definitional nature of text fragments by using an n-gram model, whereas Klavans and Muresan (2001) apply pattern matching techniques at the lexical level guided by cue phrases such as "is called" and "is defined as". More recently, a domain-independent supervised approach, named Word-Class Lattices (WCLs), was presented which learns lattice-based definition classifiers applied to candidate sentences containing the input terms (Navigli and Velardi, 2010). To avoid the burden of manually creating a training dataset, definitional patterns can be extracted automatically. Faralli and Navigli (2013) utilized Wikipedia as a huge source of definitions and simple, yet effective heuristics to automatically annotate them. Reiplinger et al. (2012) experimented with two different approaches for the acquisition of lexical-syntactic patterns. The first approach bootstraps patterns from a domain corpus and then manually refines the acquired patterns. The second approach, instead, automatically acquires definitional sentences by using a more sophisticated syntactic and semantic processing. The results show high precision in both cases. However, all the above approaches need large domain corpora, the poor availability of which hampers the creation of wide-coverage glossaries for several domains. To avoid the need to use a large corpus, domain terminologies can be obtained by using Doubly-Anchored Patterns (DAPs)

	AI				Finance			
	# terms	P	# glosses	P	# terms	P	# glosses	P
ProToDoG	4983	83%	5326	84%	7370	95%	9795	96%
TaxoLearn	427	77%	834	79%	2348	86%	1064	88%

Table 5: Number and precision of terms and glosses extracted by ProToDoG and TaxoLearn in the Artificial Intelligence (AI) and Finance domains.

which, given a (term, hypernym) pair, extract from the Web sentences matching manually-defined patterns like "<hypernym> such as <term>, and *" (Kozareva and Hovy, 2010b). This term extraction process is further extended by harvesting new hypernyms using the corresponding inverse patterns (called DAP^{-1}) like "* such as <term$_1$>, and <term$_2$>". Similarly to ProToDoG, this approach drops the requirement of a domain corpus and starts from a small number of (term, hypernym) seeds. However, while DAPs have proven useful in the induction of domain taxonomies (Kozareva and Hovy, 2010b), they cannot be applied to the glossary learning task because the extracted sentences are not formal definitions. In contrast, ProToDoG performs the novel task of multi-domain glossary acquisition from the Web by bootstrapping the extraction process with a few (term, hypernym) seeds. Bootstrapping techniques (Brin, 1998; Agichtein and Gravano, 2000; Paşca et al., 2006) have been successfully applied to several tasks, including learning semantic relations (Pantel and Pennacchiotti, 2006), extracting surface text patterns for open-domain question answering (Ravichandran and Hovy, 2002), semantic tagging (Huang and Riloff, 2010) and unsupervised Word Sense Disambiguation (Yarowsky, 1995). ProToDoG synergistically integrates bootstrapping with probabilistic topic models so as to keep the glossary acquisition process within the target domains as much as possible.

7 Conclusions

In this paper we have presented ProToDoG, a new, minimally-supervised approach to multi-domain glossary acquisition. Starting from a small set of hypernymy seeds which identify each domain of interest, we apply a bootstrapping approach which iteratively obtains generalized patterns from Web glossaries and then applies them to the extraction of term/gloss pairs. To our knowledge, ProToDoG is the first approach to large-scale probabilistic glossary learning which jointly acquires thousands of terms and glosses for dozens of domains with minimal supervision.

At the core of ProToDoG lies our glossary bootstrapping approach, thanks to which we can drop the requirements of existing techniques such as the ready availability of domain corpora, which often do not contain enough definitions (cf. Table 5), and the manual definition of lexical patterns, which typically extract sentence snippets instead of formal glosses.

ProToDoG will be made available to the research community. Beyond the immediate usability of the output glossaries (we show an excerpt in Table 4), we also wish to show the benefit of ProToDoG in gloss-driven approaches to taxonomy learning (Navigli et al., 2011; Velardi et al., 2013) and Word Sense Disambiguation (Duan and Yates, 2010; Faralli and Navigli, 2012). The 30-domain glossaries and gold standards created for our experiments are available from `http://lcl.uniroma1.it/protodog`.

We remark that the terminologies covered with ProToDoG are not only precise, but are also one order of magnitude greater than those covered in individual online glossaries. As future work, we plan to study the ability of ProToDoG to acquire domain glossaries at different levels of specificity (i.e., domains vs. subdomains). Finally, we will adapt ProToDoG to other languages, by translating the `glossary` keyword used in step (2), along the lines of (De Benedictis et al., 2013).

Acknowledgments

The authors gratefully acknowledge the support of the "MultiJEDI" ERC Starting Grant No. 259234.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the 5th ACM conference on Digital Libraries*, pages 85–94, San Antonio, Texas, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Sergey Brin. 1998. Extracting patterns and relations from the World Wide Web. In *Proceedings of the International Workshop on The World Wide Web and Databases*, pages 172–183, London, UK.

Hang Cui, Min-Yen Kan, and Tat-Seng Chua. 2007. Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems*, 25(2):8.

Flavio De Benedictis, Stefano Faralli, and Roberto Navigli. 2013. GlossBoot: Bootstrapping Multilingual Domain Glossaries from the Web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 528–538, Sofia, Bulgaria.

Weisi Duan and Alexander Yates. 2010. Extracting glosses to disambiguate word senses. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 627–635, Los Angeles, CA, USA.

Stefano Faralli and Roberto Navigli. 2012. A New Minimally-supervised Framework for Domain Word Sense Disambiguation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1411–1422, Jeju, Korea.

Stefano Faralli and Roberto Navigli. 2013. A Java Framework for Multilingual Definition and Hypernym Extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 103–108, Sofia, Bulgaria.

Atsushi Fujii and Tetsuya Ishikawa. 2000. Utilizing the World Wide Web as an encyclopedia: extracting term descriptions from semi-structured texts. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 488–495, Hong Kong.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 539–545, Nantes, France.

Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

Ruihong Huang and Ellen Riloff. 2010. Inducing domain-specific semantic class taggers from (almost) nothing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 275–285, Uppsala, Sweden.

Judith Klavans and Smaranda Muresan. 2001. Evaluation of the DEFINDER system for fully automatic glossary construction. In *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, pages 324–328, Washington, D.C., USA.

Zornitsa Kozareva and Eduard H. Hovy. 2010a. Not all seeds are equal: Measuring the quality of text mining seeds. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 618–626, Los Angeles, California, USA.

Zornitsa Kozareva and Eduard H. Hovy. 2010b. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1110–1118, Cambridge, MA, USA.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli and Paola Velardi. 2010. Learning Word-Class Lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327, Uppsala, Sweden.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the 22th International Joint Conference on Artificial Intelligence*, pages 1872–1877, Barcelona, Spain.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Names and similarities on the Web: Fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 809–816, Sydney, Australia.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Sydney, Australia*, pages 113–120, Sydney, Australia.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international*

conference on World Wide Web, WWW '08, pages 91–100, New York, NY, USA.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47, Philadelphia, PA, USA.

Melanie Reiplinger, Ulrich Schäfer, and Magdalena Wolska. 2012. Extracting glossary sentences from scholarly articles: A comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 55–65, Jeju Island, Korea.

Horacio Saggion. 2004. Identifying definitions in text collections for question answering. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 1927–1930, Lisbon, Portugal.

Mark Steyvers and Tom Griffiths, 2007. *Probabilistic Topic Models*. Lawrence Erlbaum Associates.

Paola Velardi, Roberto Navigli, and Pierluigi D'Amadio. 2008. Mining the web to create specialized glossaries. *IEEE Intelligent Systems*, 23(5):18–25.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation rivaling supervised methods. In *Proceedings of the* 33^{rd} *Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA, USA.

Joint Learning of Phonetic Units and Word Pronunciations for ASR

Chia-ying Lee, Yu Zhang, James Glass
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{chiaying,yzhang87,jrg}@csail.mit.edu

Abstract

The creation of a pronunciation lexicon remains the most inefficient process in developing an Automatic Speech Recognizer (ASR). In this paper, we propose an unsupervised alternative – requiring no language-specific knowledge – to the conventional manual approach for creating pronunciation dictionaries. We present a hierarchical Bayesian model, which jointly discovers the phonetic inventory and the Letter-to-Sound (L2S) mapping rules in a language using only transcribed data. When tested on a corpus of spontaneous queries, the results demonstrate the superiority of the proposed joint learning scheme over its sequential counterpart, in which the latent phonetic inventory and L2S mappings are learned separately. Furthermore, the recognizers built with the automatically induced lexicon consistently outperform grapheme-based recognizers and even approach the performance of recognition systems trained using conventional supervised procedures.

1 Introduction

Modern automatic speech recognizers require a few essential ingredients such as a signal representation of the speech signal, a search component, and typically a set of stochastic models that capture 1) the acoustic realizations of the basic sounds of a language, for example, phonemes, 2) the realization of words in terms of these sounds, and 3) how words are combined in spoken language. When creating a speech recognizer for a new language the usual requirements are: first, a large speech corpus with word-level annotations; second, a pronunciation dictionary that essentially defines a phonetic inventory for the language as well as word-level pronunciations, and third, optional additional text data that can be used to train the language model. Given these data and some decision about the signal representation, e.g., centi-second Mel-Frequency Cepstral Coefficients (MFCCs) (Davis and Mermelstein, 1980) with various derivatives, as well as the nature of the acoustic and language model such as 3-state HMMs and n-grams, iterative training methods can be used to effectively learn the model parameters for the acoustic and language models. Although the details of the components have changed through the years, this basic ASR formulation was well established by the late 1980's, and has not really changed much since then.

One of the interesting aspects of this formulation is the inherent dependence on the dictionary, which defines both the phonetic inventory of a language, and the pronunciations of all the words in the vocabulary. The dictionary is arguably the cornerstone of a speech recognizer as it provides the essential transduction from sounds to words. Unfortunately, the dependency on this resource is a significant impediment to the creation of speech recognizers for new languages, since they are typically created by experts, whereas annotated corpora can be relatively more easily created by native speakers of a language.

The existence of an expert-derived dictionary in the midst of stochastic speech recognition models is somewhat ironic, and it is natural to ask why it continues to receive special status after all these years. Why can we not learn the inventory of sounds of a language and associated word pronunciations automatically, much as we learn our acoustic model parameters? If successful, we would move one step forward towards breaking the language barrier that

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 182–192,
Seattle, Washington, USA, 18-21 October 2013.

limits us from having speech recognizers for all languages of the world, instead of the less than 2% that currently exist.

In this paper, we investigate the problem of inferring a pronunciation lexicon from an annotated corpus without exploiting any language-specific knowledge. We formulate our approach as a hierarchical Bayesian model, which jointly discovers the acoustic inventory and the latent encoding scheme between the letters and the sounds of a language. We evaluate the quality of the induced lexicon and acoustic model through a series of speech recognition experiments on a conversational weather query corpus (Zue et al., 2000). The results demonstrate that our model consistently generates close performance to recognizers that are trained with expert-defined phonetic inventory and lexicon. Compared to grapheme-based recognizers, our model is capable of improving the Word Error Rates (WERs) by at least 15.3%. Finally, the joint learning framework proposed in this paper is proven to be much more effective than modeling the acoustic units and the letter-to-sound mappings separately, as shown in a 45% WER deduction our model achieves compared to a sequential approach.

2 Related Work

Various algorithms for learning sub-word based pronunciations were proposed in (Lee et al., 1988; Fukada et al., 1996; Bacchiani and Ostendorf, 1999; Paliwal, 1990). In these previous approaches, spoken samples of a word are gathered, and usually only one single pronunciation for the word is derived based on the acoustic evidence observed in the spoken samples. The major difference between our work and these previous works is that our model learns word pronunciations in the context of letter sequences. More specifically, our model learns letter pronunciations first and then concatenates the pronunciation of each letter in a word to form the word pronunciation. The advantage of our approach is that pronunciation knowledge learned for a particular letter in some arbitrary word can subsequently be used to help learn the letter's pronunciation in other words. This property allows our model to potentially learn better pronunciations for less frequent words.

The more recent work by Garcia and Gish (2006) and Siu et al. (2013) has made extensive use of self-organizing units for keyword spotting and other tasks for languages with limited linguistic resources. Others who have more recently explored the unsupervised space include (Varadarajan et al., 2008; Jansen and Church, 2011; Lee and Glass, 2012). The latter work introduced a non-parametric Bayesian inference procedure for automatically learning acoustic units that is most similar to our current work except that our model also infers word pronunciations simultaneously.

The concept of creating a speech recognizer for a language with only orthographically annotated speech data has also been explored previously by means of graphemes. This approach has been shown to be effective for alphabetic languages with relatively straightforward grapheme to phoneme transformations and does not require any unsupervised learning of units or pronunciations (Killer et al., 2003; Stüker and Schultz, 2004). As we explain in later sections, grapheme-based systems can actually be regarded as a special case of our model; therefore, we expect our model to have greater flexibilities for capturing pronunciation rules of graphemes.

3 Model

The goal of our model is to induce a word pronunciation lexicon from spoken utterances and their corresponding word transcriptions. No other language-specific knowledge is assumed to be available, including the phonetic inventory of the language. To achieve the goal, our model needs to solve the following two tasks:

- Discover the phonetic inventory.
- Reveal the latent mapping between the letters and the discovered phonetic units.

We propose a hierarchical Bayesian model for jointly discovering the two latent structures from an annotated speech corpus. Before presenting our model, we first describe the key latent and observed variables of the problem.

Letter (l_i^m) We use l_i^m to denote the i^{th} letter observed in the word transcription of the m^{th} training sample. To be sure, a training sample involves a speech utterance and its

corresponding text transcription. The letter sequence composed of l_i^m and its context, namely $l_{i-\kappa}^m, \cdots, l_{i-1}^m, l_i^m, l_{i+1}^m, \cdots, l_{i+\kappa}^m$, is denoted as $\vec{l}_{i,\kappa}^m$. Although l_i^m is referred to as a *letter* in this paper, it can represent any *character* observed in the text data, including space and symbols indicating sentence boundaries. The set of unique characters observed in the data set is denoted as G. For notation simplicity, we use $\mathcal{L}_\kappa$ to denote the set of letter sequences of length $2\kappa + 1$ that appear in the dataset and use $\vec{l}_\kappa$ to denote the elements in $\mathcal{L}_\kappa$. Finally, $\mathbb{P}(\vec{l}_\kappa)$ is used to represent the *parent* of $\vec{l}_\kappa$, which is a substring of $\vec{l}_\kappa$ with the first and the last characters truncated.

Number of Mapped Acoustic Units (n_i^m) Each letter l_i^m in the transcriptions is assumed to be mapped to a certain number of phonetic units. For example, the letter *x* in the word *fox* is mapped to 2 phonetic units /k/ and /s/, while the letter *e* in the word *lake* is mapped to 0 phonetic units. We denote this number as n_i^m and limit its value to be 0, 1 or 2 in our model. The value of n_i^m is always unobserved and needs to be inferred by the our model.

Identity of the Acoustic Unit ($c_{i,p}^m$) For each phonetic unit that l_i^m maps to, we use $c_{i,p}^m$, for $1 \leq p \leq n_i^m$, to denote the identity of the phonetic unit. Note that the phonetic inventory that describes the data set is unknown to our model, and the identities of the phonetic units are associated with the acoustic units discovered automatically by our model.

Speech Feature x_t^m The observed speech data in our problem are converted to a series of 25 ms 13-dimensional MFCCs (Davis and Mermelstein, 1980) and their first- and second-order time derivatives at a 10 ms analysis rate. We use $x_t^m \in \mathbb{R}^{39}$ to denote the t^{th} feature frame of the m^{th} utterance.

3.1 Generative Process

We present the generative process for a single training sample (i.e., a speech utterance and its corresponding text transcription); to keep notation simple, we discard the index variable m in this section.

For each l_i in the transcription, the model generates n_i, given $\vec{l}_{i,\kappa}$, from the 3-dimensional categorical distribution $\phi_{\vec{l}_{i,\kappa}}(n_i)$. Note that for every unique $\vec{l}_{i,\kappa}$ letter sequence, there is an associated $\phi_{\vec{l}_{i,\kappa}}(n_i)$

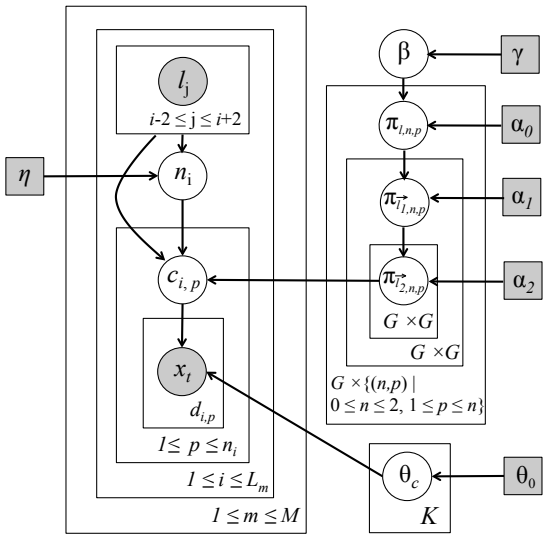


Figure 1: The graphical representation of the proposed hierarchical Bayesian model. The shaded circle denotes the observed text and speech data, and the squares denote the hyperparameters of the priors in our model. See Sec. 3 for a detailed explanation of the generative process of our model.

distribution, which captures the fact that the number of phonetic units a letter maps to may depend on its context. In our model, we impose a Dirichlet distribution prior $Dir(\eta)$ on $\phi_{\vec{l}_{i,\kappa}}(n_i)$.

If $n_i = 0$, l_i is not mapped to any acoustic units and the generative process stops for l_i; otherwise, for $1 \leq p \leq n_i$, the model generates $c_{i,p}$ from:

$$c_{i,p} \sim \pi_{\vec{l}_{i,\kappa},n_i,p} \tag{1}$$

where $\pi_{\vec{l}_{i,\kappa},n_i,p}$ is a K-dimensional categorical distribution, whose outcomes correspond to the phonetic units discovered by the model from the given speech data. Eq. 1 shows that for each combination of $\vec{l}_{i,\kappa}$, n_i and p, there is an unique categorical distribution. An important property of these categorical distributions is that they are coupled together such that their outcomes point to a consistent set of phonetic units. In order to enforce the coupling, we construct $\pi_{\vec{l}_{i,\kappa},n_i,p}$ through a hierarchical process.

$$\beta \sim Dir(\gamma) \tag{2}$$

$$\pi_{\vec{l}_{i,\kappa},n_i,p} \sim Dir(\alpha_\kappa \beta) \text{ for } \kappa = 0 \tag{3}$$

$$\pi_{\vec{l}_{i,\kappa},n_i,p} \sim Dir(\alpha_\kappa \pi_{\vec{l}_{i,\kappa-1},n_i,p}) \text{ for } \kappa \geq 1 \tag{4}$$

To interpret Eq. 2 to Eq. 4, we envision that the observed speech data are generated by a K-component mixture model, of which the components correspond to the phonetic units in the language. As a result, β in Eq. 2 can be viewed as the mixture weight over the components, which indicates how likely we are to observe each acoustic unit in the data overall. By adopting this point of view, we can also regard the mapping between l_i and the phonetic units as a mixture model, and $\pi_{l_i,n_i,p}$[1] represents how probable l_i is mapped to each phonetic unit given n_i and p. We apply a Dirichlet distribution prior parametrized by $\alpha_0\beta$ to $\pi_{l_i,n_i,p}$ as shown in Eq. 3. With this parameterization, the mean of $\pi_{l_i,n_i,p}$ is the global mixture weight β, and α_0 controls how similar $\pi_{l_i,n_i,p}$ is to the mean. More specifically, for large $\alpha_0 \gg K$, the Dirichlet distribution is highly peaked around the mean; on the contrary, for $\alpha_0 \ll K$, the mean lies in a valley. The parameters of a Dirichlet distribution can also be viewed as pseudo-counts for each category. Eq. 4 shows that the prior for $\pi_{\vec{l}_{i,\kappa},n_i,p}$ is seeded by pseudo-counts that are proportional to the mapping weights over the phonetic units of l_i in a shorter context. In other words, the mapping distribution of l_i in a shorter context can be thought of as a back-off distribution of l_i's mapping weights in a longer context.

Each component of the K-dimensional mixture model is linked to a 3-state Hidden Markov Model (HMM). These K HMMs are used to model the phonetic units in the language (Jelinek, 1976). The emission probability of each HMM state is modeled by a diagonal Gaussian Mixture Model (GMM). We use θ_c to represent the set of parameters that define the c^{th} HMM, which includes the state transition probability and the GMM parameters of each state emission distribution. The conjugate prior of θ_c is denoted as $H(\theta_0)$[2].

Finally, to finish the generative process, for each $c_{i,p}$ we use the corresponding HMM $\theta_{c_{i,p}}$ to generate the observed speech data x_t, and the generative process of the HMM determines the duration, $d_{i,p}$, of the speech segment. The complete generative model, with κ set to 2, is depicted in Fig. 1; M is the total number of transcribed utterances in the corpus, and L_m is the number of letters in utterance m. The shaded circles denote the observed data, and the squares denote the hyperparameters of the priors used in our model. Lastly, the unshaded circles denote the latent variables of our model, for which we derive inference algorithms in the next section.

[1] An abbreviation of $\pi_{\vec{l}_{i,0},n_i,p}$

[2] $H(\theta_0)$ includes a Dirichlet prior for the transition probability of each state, and a Dirichlet prior for each mixture weight of the three GMMs, and a normal-Gamma distribution for the mean and precision of each Gaussian mixture in the 3-state HMM.

4 Inference

We employ Gibbs sampling (Gelman et al., 2004) to approximate the posterior distribution of the latent variables in our model. In the following sections, we first present a message-passing algorithm for block-sampling n_i and $c_{i,p}$, and then describe how we leverage acoustic cues to accelerate the computation of the message-passing algorithm. Note that the block-sampling algorithm for n_i and $c_{i,p}$ can be parallelized across utterances. Finally, we briefly discuss the inference procedures for $\phi_{\vec{l}_\kappa}$, $\pi_{\vec{l}_\kappa,n,p}$, β, θ_c.

4.1 Block-sampling n_i and $c_{i,p}$

To understand the message-passing algorithm in this study, it is helpful to think of our model as a simplified Hidden Semi-Markov Model (HSMM), in which the letters represent the states and the speech features are the observations. However, unlike in a regular HSMM, where the state sequence is hidden, in our case, the state sequence is fixed to be the given letter sequence. With this point of view, we can modify the message-passing algorithms of Murphy (2002) and Johnson and Willsky (2013) to compute the posterior information required for block-sampling n_i and $c_{i,p}$.

Let $\mathbb{L}(x_t)$ be a function that returns the index of the letter from which x_t is generated; also, let $F_t = 1$ be a tag indicating that a new phone segment starts at $t+1$. Given the constraint that $0 \leq n_i \leq 2$, for $0 \leq i \leq L_m$ and $0 \leq t \leq T_m$, the backwards messages $B_t(i)$ and $B^*_t(i)$ for the m^{th} training sample can be defined and computed as in Eq. 5 and Eq. 7. Note that for clarity we discard the index variable m in the derivation of the algorithm.

$$
\begin{aligned}
B_t(i) &\triangleq p(x_{t+1:T}|\mathbb{L}(x_t) = i, F_t = 1) \\
&= \sum_{j=i+1}^{\min\{L,i+1+U\}} B_t^*(j) \prod_{k=i+1}^{j-1} p(n_k = 0|\vec{l}_{i,\kappa}) \\
&= \sum_{j=i+1}^{\min\{L,i+1+U\}} B_t^*(j) \prod_{k=i+1}^{j-1} \phi_{\vec{l}_{i,\kappa}}(0) \qquad (5)
\end{aligned}
$$

$$
\begin{aligned}
&B_t^*(i) \triangleq p(x_{t+1:T}|\mathbb{L}(x_{t+1}) = i, F_t = 1) \\
&= \sum_{d=1}^{T-t} p(x_{t+1:t+d}|\vec{l}_{i,\kappa}) B_{t+d}(i) \qquad (6) \\
&= \sum_{d=1}^{T-t} \{ \sum_{c_{i,1}=1}^{K} \phi_{\vec{l}_{i,\kappa}}(1) \pi_{\vec{l}_{i,\kappa},1,1}(c_{i,1}) p(x_{t+1:t+d}|\theta_{c_{i,1}}) \\
&+ \sum_{v=1}^{d-1} \sum_{c_{i,1}}^{K} \sum_{c_{i,2}}^{K} \phi_{\vec{l}_{i,\kappa}}(2) \pi_{\vec{l}_{i,\kappa},2,1}(c_{i,1}) \pi_{\vec{l}_{i,\kappa},2,2}(c_{i,2}) \\
&\times p(x_{t+1:t+v}|\theta_{c_{i,1}}) p(x_{t+v+1:t+d}|\theta_{c_{i,2}}) \} B_{t+d}(i) \qquad (7)
\end{aligned}
$$

We use $x_{t_1:t_2}$ to denote the segment consisting of $x_{t_1}, \cdots, x_{t_2}$. Our inference algorithm only allows up to U letters to emit 0 acoustic units in a row. The value of U is set to 2 for our experiments. $B_t(i)$ represents the total probability of all possible alignments between $x_{t+1:T}$ and $l_{i+1:L}$. $B_t^*(i)$ contains the probability of all the alignments between $x_{t+1:T}$ and $l_{i+1:L}$ that map x_{t+1} to l_i particularly. This alignment constraint between x_{t+1} and l_i is explicitly shown in the first term of Eq. 6, which represents how likely the speech segment $x_{t+1:t+d}$ is generated by l_i given l_i's context. This likelihood is simply the marginal probability of $p(x_{t+1:t+d}, n_i, c_{i,p}|\vec{l}_{i,\kappa})$ with n_i and $c_{i,p}$ integrated out, which can be expanded and computed as shown in the last three rows of Eq. 7. The index v specifies where the phone boundary is between the two acoustic units that l_i is aligned with when $n_i = 2$. Eq. 8 to Eq. 10 are the boundary conditions of the message passing algorithm. $B_0(0)$ carries the total probably of all possible alignments between $l_{1:L}$ and $x_{1:T}$. Eq. 9 specifies that at most U letters at the end of an sentence can be left unaligned with any speech features, while Eq. 10 indicates that all of the speech features in an utterance must be assigned to a letter.

Algorithm 1 Block-sample n_i and $c_{i,p}$ from $B_t(i)$ and $B_t^*(i)$

```
1:  i ← 0
2:  t ← 0
3:  while i < L ∧ t < T do
4:      next_i ← SampleFromB_t(i)
5:      if next_i > i + 1 then
6:          for k = i + 1 to k = next_i − 1 do
7:              n_k ← 0
8:          end for
9:      end if
10:     d, n_i, ⟨c_{i,p}⟩, v ← SampleFromB*_t(next_i)
11:     t ← t + d
12:     i ← next_i
13: end while
```

$$
B_0(0) = \sum_{j=1}^{\min\{L,U+1\}} B_0^*(j) \prod_{k=1}^{j-1} \phi_{\vec{l}_{i,\kappa}}(0) \qquad (8)
$$

$$
B_T(i) \triangleq \begin{cases} 1 & \text{if } i = L \\ \prod_{j=i+1}^{L} \phi_{\vec{l}_{i,\kappa}}(0) & \text{if } L - U \leq i < L \\ 0 & \text{if } i < L - U \end{cases} \qquad (9)
$$

$$
B_t(L) \triangleq \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise} \end{cases} \qquad (10)
$$

Given $B_t(i)$ and $B_t^*(i)$, n_i and $c_{i,p}$ for each letter in the utterance can be sampled using Alg. 1. The $SampleFromB_t(i)$ function in line 4 returns a random sample from the relative probability distribution composed by entries of the summation in Eq. 5. Line 5 to line 9 check whether l_i (and maybe l_{i+1}) is mapped to zero phonetic units. $next_i$ points to the letter that needs to be aligned with 1 or 2 phone segments starting from x_t. The number of phonetic units that l_{next_i} maps to and the identities of the units are sampled in $SampleFromB_t^*(i)$. This subroutine generates a tuple of d, n_i, $\langle c_{i,p} \rangle$ as well as v (if $n_i = 2$) from all the entries of the summation shown in Eq. 7[3].

[3]We use $\langle c_{i,p} \rangle$ to denote that $\langle c_{i,p} \rangle$ may consist of two numbers, $c_{i,1}$ and $c_{i,2}$, when $n_i = 2$.

4.2 Heuristic Phone Boundary Elimination

The variables d and v in Eq. 7 enumerate through every frame index in a sentence, treating each feature frame as a potential boundary between acoustic units. However, it is possible to exploit acoustic cues to avoid checking feature frames that are unlikely to be phonetic boundaries. We follow the presegmentation method described in Glass (2003) to skip roughly 80% of the feature frames and greatly speed up the computation of $B_t^*(i)$.

Another heuristic applied to our algorithm to reduce the search space for d and v is based on the observation that the average duration of phonetic units is usually no longer than 300 ms. Therefore, when computing $B_t^*(i)$, we only consider speech segments that are shorter than 300 ms to avoid aligning letters to speech segments that are too long to be phonetic units.

4.3 Sampling $\phi_{\vec{l}_\kappa}$, $\pi_{\vec{l}_\kappa,n_i,p}$, β and θ_c

Sampling $\phi_{\vec{l}_\kappa}$ To compute the posterior distribution of $\phi_{\vec{l}_\kappa}$, we count how many times $\vec{l}_\kappa$ is mapped to 0, 1 and 2 phonetic units from n_i^m. More specifically, we define $\mathcal{N}_{\vec{l}_\kappa}(j)$ for $0 \leq j \leq 2$ as follows:

$$\mathcal{N}_{\vec{l}_\kappa}(j) = \sum_{m=1}^{M}\sum_{i=1}^{L_m} \delta(n_i^m, j)\delta(\vec{l}_{i,\kappa}^m, \vec{l}_\kappa)$$

where we use $\delta(\cdot)$ to denote the discrete Kronecker delta. With $\mathcal{N}_{\vec{l}_\kappa}$, we can simply sample a new value for $\phi_{\vec{l}_\kappa}$ from the following distribution:

$$\phi_{\vec{l}_\kappa} \sim Dir(\eta + \mathcal{N}_{\vec{l}_\kappa})$$

Sampling $\pi_{\vec{l}_\kappa,n,p}$ and β The posterior distributions of $\pi_{\vec{l}_\kappa,n,p}$ and β are constructed recursively due to the hierarchical structure imposed on $\pi_{\vec{l}_\kappa,n,p}$ and β. We start with gathering counts for updating the π variables at the lowest level, i.e., $\pi_{\vec{l}_2,n,p}$ given that κ is set to 2 in our model implementation, and then sample pseudo-counts for the π variables at higher hierarchies as well as β. With the pseudo-counts, a new β can be generated, which allows $\pi_{\vec{l}_\kappa,n,p}$ to be re-sampled sequentially.

More specifically, we define $\mathcal{C}_{\vec{l}_2,n,p}(k)$ to be the number of times that $\vec{l}_2$ is mapped to n units and the unit in position p is the k^{th} phonetic unit. This value can be counted from the current values of $c_{i,p}^m$ as follows.

$$\mathcal{C}_{\vec{l}_2,n,p}(k) = \sum_{m=1}^{M}\sum_{i=1}^{L_m} \delta(\vec{l}_{i,2}, \vec{l}_2)\delta(n_i^m, n)\delta(c_{i,p}^m, k)$$

To derive the posterior distribution of $\pi_{\vec{l}_1,n,p}$ analytically, we need to sample pseudo-counts $\mathcal{C}_{\vec{l}_1,n,p}$, which is defined as follows.

$$\mathcal{C}_{\vec{l}_1,n,p}(k) = \sum_{\vec{l}_2 \in \mathcal{U}_{\vec{l}_1}} \sum_{i=1}^{\mathcal{C}_{\vec{l}_2,n,p}(k)} \mathbb{I}[\nu_i < \frac{\alpha_2 \pi_{\vec{l}_1,n,p}(k)}{i + \alpha_2 \pi_{\vec{l}_1,n,p}(k)}] \tag{11}$$

We use $\mathcal{U}_{\vec{l}_1} = \{\vec{l}_2 | \mathbb{P}(\vec{l}_2) = \vec{l}_1\}$ to denote the set of $\vec{l}_2$ whose parent is $\vec{l}_1$ and ν_i to represent random variables sampled from a uniform distribution between 0 and 1. Eq. 11 can be applied recursively to compute $\mathcal{C}_{\vec{l}_0,n,p}(k)$ and $\mathcal{C}_{\_,n,p}(k)$, the pseudo-counts that are applied to the conjugate priors of $\pi_{\vec{l}_0,n,p}$ and β. With the pseudo-count variables computed, new values for β and $\pi_{\vec{l}_\kappa,n,p}$ can be sampled sequentially as shown in Eq. 12 to Eq. 14.

$$\beta \sim Dir(\gamma + \mathcal{C}_{\_,n,p}) \tag{12}$$

$$\pi_{\vec{l}_\kappa,n,p} \sim Dir(\alpha_\kappa \beta + \mathcal{C}_{\vec{l}_\kappa,n,p}) \text{ for } \kappa = 0 \tag{13}$$

$$\pi_{\vec{l}_\kappa,n,p} \sim Dir(\alpha_\kappa \pi_{\vec{l}_{\kappa-1},n,p} + \mathcal{C}_{\vec{l}_\kappa,n,p}) \text{ for } \kappa \geq 1 \tag{14}$$

5 Experimental Setup

To test the effectiveness of our model for joint learning phonetic units and word pronunciations from an annotated speech corpus, we construct speech recognizers out of the training results of our model. The performance of the recognizers is evaluated and compared against three baselines: first, a grapheme-based speech recognizer; second, a recognizer built by using an expert-crafted lexicon, which is referred to as an expert lexicon in the rest of the paper for simplicity; and third, a recognizer built by discovering the phonetic units and L2S pronunciation rules *sequentially* without using a lexicon. In this section, we provide a detailed description of the experimental setup.

η	γ	α_0	α_1	α_2	θ_0	κ	K
$\langle 0.1 \rangle_3$	$\langle 10 \rangle_{100}$	1	0.1	0.2	*	2	100

Table 1: The values of the hyperparameters of our model. We use $\langle a \rangle_D$ to denote a D-dimensional vector with all entries being a. *We follow the procedure reported in (Lee and Glass, 2012) to set up the HMM prior θ_0.

5.1 Dataset

All the speech recognition experiments reported in this paper are performed on a weather query dataset, which consists of narrow-band, conversational telephone speech (Zue et al., 2000). We follow the experimental setup of McGraw et al. (2013) and split the corpus into a training set of 87,351 utterances, a dev set of 1,179 utterances and a test set of 3,497 utterances. A subset of 10,000 utterances is randomly selected from the training set. We use this subset of data for training our model to demonstrate that our model is able to discover the phonetic composition and the pronunciation rules of a language even from just a few hours of data.

5.2 Building a Recognizer from Our Model

The values of the hyperparameters of our model are listed in Table 1. We run the inference procedure described in Sec. 4 for 10,000 times on the randomly selected 10,000 utterances. The samples of $\phi_{\vec{l}_\kappa}$ and $\pi_{\vec{l}_\kappa n,p}$ from the last iteration are used to decode n_i^m and $c_{i,p}^m$ for each sentence in the entire training set by following the block-sampling algorithm described in Sec. 4.1. Since $c_{i,p}^m$ is the phonetic mapping of l_i^m, by concatenating the phonetic mapping of every letter in a word, we can obtain a pronunciation of the word represented in the labels of discovered phonetic units. For example, assume that word w appears in sentence m and consists of $l_3 l_4 l_5$ (the sentence index m is ignored for simplicity). Also, assume that after decoding, $n_3 = 1$, $n_4 = 2$ and $n_5 = 1$. A pronunciation of w is then encoded by the sequence of phonetic labels $c_{3,1} c_{4,1} c_{4,2} c_{5,1}$. By repeating this process for each word in every sentence for the training set, a list of word pronunciations can be compiled and used as a stochastic lexicon to build a speech recognizer.

In theory, the HMMs inferred by our model can be directly used as the acoustic model of a monophone speech recognizer. However, if we regard the $c_{i,p}$ labels of each utterance as the phone transcription of the sentence, then a new acoustic model can be easily re-trained on the entire data set. More conveniently, the phone boundaries corresponding to the $c_{i,p}$ labels are the by-products of the block-sampling algorithm, which are indicated by the values of d and v in line 10 of Alg. 1 and can be easily saved during the sampling procedure. Since these data are readily available, we re-build a context-independent model on the entire data set. In this new acoustic model, a 3-state HMM is used to model each phonetic unit, and the emission probability of each state is modeled by a 32-mixture GMM.

Finally, a trigram language model is built by using the word transcriptions in the full training set. This language model is utilized in all speech recognition experiments reported in this paper. Finite State Transducers (FSTs) are used to build all the recognizers used in this study. With the language model, the lexicon and the context-independent acoustic model constructed by the methods described in this section, we can build a speech recognizer from the learning output of the proposed model without the need of a pre-defined phone inventory and any expert-crafted lexicons.

5.2.1 Pronunciation Mixture Model Retraining

McGraw et al. (2013) presented the Pronunciation Mixture Model (PMM) for composing stochastic lexicons that outperform pronunciation dictionaries created by experts. Although the PMM framework was designed to incorporate and augment expert lexicons, we found that it can be adapted to polish the pronunciation list generated by our model.

In particular, the training procedure for PMMs includes three steps. First, train a L2S model from a manually specified expert-pronunciation lexicon; second, generate a list of pronunciations for each word in the dataset using the L2S model; and finally, use an acoustic model to re-weight the pronunciations based on the acoustic scores of the spoken examples of each word.

To adapt this procedure for our purposes, we simply plug in the word pronunciations and the acoustic model generated by our model. Once we obtain the re-weighted lexicon, we re-generate forced

phone alignments and retrain the acoustic model, which can be utilized to repeat the PMM lexicon reweighting procedure. For our experiments, we iterate through this model refining process until the recognition performance converges.

5.2.2 Triphone Model

Conventionally, to train a context-dependent acoustic model, a list of questions based on the linguistic properties of phonetic units is required for growing decision tree classifiers (Young et al., 1994). However, such language-specific knowledge is not available for our training framework; therefore, our strategy is to compile a question list that treats each phonetic unit as a unique linguistic class. In other words, our approach to training a context-dependent acoustic model for the automatically discovered units is to let the decision trees grow fully based on acoustic evidence.

5.3 Baselines

We compare the recognizers trained by following the procedures described in Sec. 5.2 against three baselines. The first baseline is a grapheme-based speech recognizer. We follow the procedure described in Killer et al. (2003) and train a 3-state HMM for each grapheme, which we refer to as the *monophone grapheme* model. Furthermore, we create a *singleton question set* (Killer et al., 2003), in which each grapheme is listed as a question, to train a *triphone grapheme* model. Note that to enforce better initial alignments between the graphemes and the speech data, we use a pre-trained acoustic model to identify the non-speech segments at the beginning and the end of each utterance before starting training the monophone grapheme model.

Our model jointly discovers the phonetic inventory and the L2S mapping rules from a set of transcribed data. An alternative of our approach is to learn the two latent structures sequentially. We follow the training procedure of Lee and Glass (2012) to learn a set of acoustic models from the speech data and use these acoustic models to generate a phone transcription for each utterance. The phone transcriptions along with the corresponding word transcriptions are fed as inputs to the L2S model proposed in Bisani and Ney (2008). A stochastic lexicon can be learned by applying the L2S model and the discovered acoustic models to PMM. This two-stage approach for training a speech recognizer without an expert lexicon is referred to as the *sequential model* in this paper.

unit(%)	Monophone
Our model	17.0
Oracle	13.8
Grapheme	32.7
Sequential model	31.4

Table 2: Word error rates generated by the four monophone recognizers described in Sec. 5.2 and Sec. 5.3 on the weather query corpus.

Finally, we compare our system against a recognizer trained from an *oracle* recognition system. We build the oracle recognizer on the same weather query corpus by following the procedure presented in McGraw et al. (2013). This oracle recognizer is then applied to generate forced-aligned phone transcriptions for the training utterances, from which we can build both monophone and triphone acoustic models. The expert-crafted lexicon used in the oracle recognizer is also used in this baseline. Note that for training the triphone model, we compose a singleton question list (Killer et al., 2003) that has every expert-defined phonetic unit as a question. We use this singleton question list instead of a more sophisticated one to ensure that this baseline and our system differ only in the acoustic model and the lexicon used to generate the initial phone transcriptions. We call this baseline the *oracle* baseline.

6 Results and Analysis

6.1 Monophone Systems

Table 2 shows the WERs produced by the four monophone recognizers described in Sec. 5.2 and Sec. 5.3. It can be seen that our model outperforms the grapheme and the sequential model baselines significantly while approaching the performance of the supervised oracle baseline. The improvement over the sequential baseline demonstrates the strength of the proposed joint learning framework. More specifically, unlike the sequential baseline, in which the acoustic units are discovered independently from the text data, our model is able to exploit the L2S mapping constraints provided by the word transcriptions to cluster speech segments.

By comparing our model to the grapheme baseline, we can see the advantage of modeling the pronunciations of a letter using a mixture model, especially for a language like English which has many pronunciation irregularities. However, even for languages with straightforward pronunciation rules, the concept of modeling letter pronunciations using mixture models still applies. The main difference is that the mixture weights for letters of languages with simple pronunciation rules will be sparser and spikier. In other words, in theory, our model should always perform comparable to, if not better than, grapheme recognizers.

Last but not least, the recognizer trained with the automatically induced lexicon performs similarly to the recognizer initialized by an oracle recognition system, which demonstrates the effectiveness of the proposed model for discovering the phonetic inventory and a pronunciation lexicon from an annotated corpus. In the next section, we provide some insights into the quality of the learned lexicon and into what could have caused the performance gap between our model and the conventionally trained recognizer.

6.2 Pronunciation Entropy

The major difference between the recognizer that is trained by using our model and the recognizer that is seeded by an oracle recognition system is that the former uses an automatically discovered lexicon, while the latter exploits an expert-defined pronunciation dictionary. In order to quantify, as well as to gain insights into, the difference between these two lexicons, we define the average pronunciation entropy, $\hat{H}$, of a lexicon as follows.

$$\hat{H} \equiv \frac{-1}{|V|} \sum_{w \in V} \sum_{b \in \mathcal{B}(w)} p(b) \log p(b) \qquad (15)$$

where V denotes the vocabulary of a lexicon, $\mathcal{B}(w)$ represents the set of pronunciations of a word w and $p(b)$ stands for the weight of a certain pronunciation b. Intuitively, we can regard $\hat{H}$ as an indicator of how much pronunciation variation that each word in a lexicon has on average. Table 3 shows that the $\hat{H}$ values of the lexicon induced by our model and the expert-defined lexicon as well as their respective PMM-refined versions[4]. In Table 3, we can see that the automatically-discovered lexicon and its PMM-reweighted versions have much higher $\hat{H}$ values than their expert-defined counterparts. These higher $\hat{H}$ values imply that the lexicon induced by our model contains more pronunciation variation than the expert-defined lexicon. Therefore, the lattices constructed during the decoding process for our recognizer tend to be larger than those constructed for the oracle baseline, which explains the performance gap between the two systems in Table 2 and Table 3.

Our model (Discovered lexicon)	*PMM iterations*		
	0	1	2
$\hat{H}$ (bit)	4.58	3.47	3.03
WER (%)	17.0	16.6	15.9
Oracle (Expert lexicon)	*PMM iterations*		
	0	1	2
$\hat{H}$ (bit)	0.69	0.90	0.92
WER (%)	13.8	12.8	12.4

Table 3: The upper-half of the table shows the average pronunciation entropies, $\hat{H}$, of the lexicons induced by our model and refined by PMM as well as the WERs of the monophone recognizers built with the corresponding lexicons for the weather query corpus. The definition of $\hat{H}$ can be found in Sec. 6.2. The first row of the lower-half of the table lists the average pronunciation entropies, $\hat{H}$, of the expert-defined lexicon and the lexicons generated and weighted by the L2P-PMM framework described in McGraw et al. (2013). The second row of the lower-half of the table shows the WERs of the recognizers that are trained with the expert-lexicon and its PMM-refined versions.

As shown in Table 3, even though the lexicon induced by our model is noisier than the expert-defined dictionary, the PMM retraining framework consistently refines the induced lexicon and improves the performance of the recognizers[5]. To the best of our knowledge, we are the first to apply PMM to lexicons that are created by a fully unsu-

[4]We build the PMM-refined version of the expert-defined lexicon by following the L2P-PMM framework described in McGraw et al. (2013).

[5]The recognition results all converge in $2 \sim 3$ PMM retraining iterations.

pronunciations	*pronunciation probabilities*		
	Our model	1 PMM	2 PMM
93 56 87 39 19	0.125	-	-
93 56 61 87 73 99	0.125	-	-
11 56 61 87 73 99	0.125	0.400	0.419
93 20 75 87 17 27 52	0.125	0.125	0.124
55 93 56 61 87 73 84 19	0.125	0.220	0.210
93 26 61 87 49	0.125	0.128	0.140
63 83 86 87 73 53 19	0.125	-	-
93 26 61 87 61	0.125	0.127	0.107

Table 4: Pronunciation lists of the word *Burma* produced by our model and refined by PMM after 1 and 2 iterations.

pervised method. Therefore, in this paper, we provide further analysis on how PMM helps enhance the performance of our model.

We compare the pronunciation lists for the word *Burma* generated by our model and refined iteratively by PMM in Table 4. The first column of Table 4 shows all the pronunciations of *Burma* discovered by our model, to which our model assigns equal probabilities to create a stochastic list[6]. As demonstrated in the third and the fourth columns of Table 4, the PMM framework is able to iteratively re-distribute the pronunciation weights and filter out less-likely pronunciations, which effectively reduces both the size and the entropy of the stochastic lexicon generated by our model. The benefits of using the PMM to refine the induced lexicon are twofold. First, the search space constructed during the recognition decoding process with the refined lexicon is more constrained, which is the main reason why the PMM is capable of improving the performance of the monophone recognizer that is trained with the output of our model. Secondly, and more importantly, the refined lexicon can greatly reduce the size of the FST built for the triphone recognizer of our model. These two observations illustrate why the PMM framework can be an useful tool for enhancing the lexicon discovered automatically by our model.

6.3 Triphone Systems

The best monophone systems of the grapheme baseline, the oracle baseline and our model are used to generate forced-aligned phone transcriptions, which are used to train the triphone models described in Sec. 5.2.2 and Sec. 5.3. Table 5 shows the WERs of the triphone recognition systems. Note that if a more conventional question list, for example, a list that contains rules to classify phones into different broad classes, is used to build the oracle triphone system, the WER can be reduced to 6.5%. However, as mentioned earlier, in order to gain insights into the quality of the induced lexicon and the discovered phonetic set, we compare our model against an oracle triphone system that is built by using a singleton question set.

Unit(%)	Triphone
Our model	13.4
Oracle	10.0
Grapheme	15.7

Table 5: Word error rates of the triphone recognizers. The triphone recognizers are all built by using the phone transcriptions generated by their best monohpone system. For the oracle initialized baseline and for our model, the PMM-refined lexicons are used to build the triphone recognizers.

By comparing Table 2 and Table 5, we can see that the grapheme triphone improves by a large margin compared to its monophone counterpart, which is consistent with the results reported in (Killer et al., 2003). However, even though the grapheme baseline achieves a great performance gain with context-dependent acoustic models, the recognizer trained using the lexicon learned by our model and subsequently refined by PMM still outperforms the grapheme baseline. The consistently better performance our model achieves over the grapheme baseline demonstrates the strength of modeling the pronunciation of each letter with a mixture model that is presented in this paper.

Last but not least, by comparing Table 2 and Table 5, it can be seen that the relative performance gain achieved by our model is similar to that obtained by the oracle baseline. Both Table 2 and Table 5 show that even without exploiting any language-specific knowledge during training, our recognizer is able to perform comparably with the recognizer trained using an expert lexicon. The ability of our model to obtain such similar performance

[6]It is also possible to assign probabilities proportional to the decoding scores of the word tokens.

further supports the effectiveness of the joint learning framework proposed in this paper for discovering the phonetic inventory and the word pronunciations from simply an annotated speech corpus.

7 Conclusion

We present a hierarchical Bayesian model for simultaneously discovering acoustic units and learning word pronunciations from transcribed spoken utterances. Both monophone and triphone recognizers can be built on the discovered acoustic units and the inferred lexicon. The recognizers trained with the proposed unsupervised method consistently outperforms grapheme-based recognizers and approach the performance of recognizers trained with expert-defined lexicons. In the future, we plan to apply this technology to develop ASRs for more languages.

Acknowledgements

The authors would like to thank Ian McGraw and Ekapol Chuangsuwanich for their advice on the PMM and recognition experiments presented in this paper. Thanks to the anonymous reviewers for helpful comments. Finally, the authors would like to thank Stephen Shum for proofreading and editing the early drafts of this paper.

References

Michiel Bacchiani and Mari Ostendorf. 1999. Joint lexicon, acoustic unit inventory and model design. *Speech Communication*, 29:99 – 114.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, May.

Steven B. Davis and Paul Mermelstein. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 28(4):357–366.

Toshiaki Fukada, Michiel Bacchiani, Kuldip Paliwal, and Yoshinori Sagisaka. 1996. Speech recognition based on acoustically derived segment units. In *Proceedings of ICSLP*, pages 1077 – 1080.

Alvin Garcia and Herbert Gish. 2006. Keyword spotting of arbitrary words using minimal speech resources. In *Proceedings of ICASSP*, pages 949–952.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall/CRC, second edition.

James Glass. 2003. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17:137 – 152.

Aren Jansen and Kenneth Church. 2011. Towards unsupervised training of speaker independent acoustic models. In *Proceedings of INTERSPEECH*, pages 1693 – 1696.

Frederick Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64:532 – 556.

Matthew J. Johnson and Alan S. Willsky. 2013. Bayesian nonparametric hidden semi-markov models. *Journal of Machine Learning Research*, 14:673–701, February.

Mirjam Killer, Sebastian Stüker, and Tanja Schultz. 2003. Grapheme based speech recognition. In *Proceeding of the Eurospeech*, pages 3141–3144.

Chia-ying Lee and James Glass. 2012. A nonparametric Bayesian approach to acoustic model discovery. In *Proceedings of ACL*, pages 40–49.

Chin-Hui Lee, Frank Soong, and Biing-Hwang Juang. 1988. A segment model based approach to speech recognition. In *Proceedings of ICASSP*, pages 501–504.

Ian McGraw, Ibrahim Badr, and James Glass. 2013. Learning lexicons from speech using a pronunciation mixture model. *IEEE Trans. on Speech and Audio Processing*, 21(2):357–366.

Kevin P. Murphy. 2002. Hidden semi-Markov models (hsmms). Technical report, University of British Columbia.

Kuldip Paliwal. 1990. Lexicon-building methods for an acoustic sub-word based speech recognizer. In *Proceedings of ICASSP*, pages 729–732.

Man-hung Siu, Herbert Gish, Arthur Chan, William Belfield, and Steve Lowe. 2013. Unsupervised training of an HMM-based self-organizing unit recgonizer with applications to topic classification and keyword discovery. *Computer, Speech, and Language*.

Sebastian Stüker and Tanja Schultz. 2004. A grapheme based speech recognition system for Russian. In *Proceedings of the 9th Conference Speech and Computer*.

Balakrishnan Varadarajan, Sanjeev Khudanpur, and Emmanuel Dupoux. 2008. Unsupervised learning of acoustic sub-word units. In *Proceedings of ACL-08: HLT, Short Papers*, pages 165–168.

Steve J. Young, J.J. Odell, and Philip C. Woodland. 1994. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of HLT*, pages 307–312.

Victor Zue, Stephanie Seneff, James Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington. 2000. Jupiter: A telephone-based conversational interface for weather information. *IEEE Trans. on Speech and Audio Processing*, 8:85–96.

MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text

Matthew Richardson
Microsoft Research
One Microsoft Way
Redmond, WA 98052
mattri@microsoft.com

Christopher J.C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cburges@microsoft.com

Erin Renshaw
Microsoft Research
One Microsoft Way
Redmond, WA 98052
erinren@microsoft.com

Abstract

We present MCTest, a freely available set of stories and associated questions intended for research on the machine comprehension of text. Previous work on machine comprehension (e.g., semantic modeling) has made great strides, but primarily focuses either on limited-domain datasets, or on solving a more restricted goal (e.g., open-domain relation extraction). In contrast, MCTest requires machines to answer multiple-choice reading comprehension questions about fictional stories, directly tackling the high-level goal of open-domain machine comprehension. Reading comprehension can test advanced abilities such as causal reasoning and understanding the world, yet, by being multiple-choice, still provide a clear metric. By being fictional, the answer typically can be found only in the story itself. The stories and questions are also carefully limited to those a young child would understand, reducing the world knowledge that is required for the task. We present the scalable crowd-sourcing methods that allow us to cheaply construct a dataset of 500 stories and 2000 questions. By screening workers (with grammar tests) and stories (with grading), we have ensured that the data is the same quality as another set that we manually edited, but at one tenth the editing cost. By being open-domain, yet carefully restricted, we hope MCTest will serve to encourage research and provide a clear metric for advancement on the machine comprehension of text.

1 Reading Comprehension

A major goal for NLP is for machines to be able to understand text as well as people. Several research disciplines are focused on this problem: for example, information extraction, relation extraction, semantic role labeling, and recognizing textual entailment. Yet these techniques are necessarily evaluated individually, rather than by how much they advance us towards the end goal. On the other hand, the goal of semantic parsing is the machine comprehension of text (MCT), yet its evaluation requires adherence to a specific knowledge representation, and it is currently unclear what the best representation is, for open-domain text.

We believe that it is useful to directly tackle the top-level task of MCT. For this, we need a way to measure progress. One common method for evaluating someone's understanding of text is by giving them a multiple-choice reading comprehension test. This has the advantage that it is objectively gradable (vs. essays) yet may test a range of abilities such as causal or counterfactual reasoning, inference among relations, or just basic understanding of the world in which the passage is set.

Therefore, we propose a multiple-choice reading comprehension task as a way to evaluate progress on MCT. We have built a reading comprehension dataset containing 500 fictional stories, with 4 multiple choice questions per story. It was built using methods which can easily scale to at least 5000 stories, since the stories were created, and the curation was done, using crowd sourcing almost entirely, at a total of $4.00 per story. We plan to periodically update the dataset to ensure that methods are not overfitting to the existing data. The dataset is open-domain, yet restricted to concepts and words that a 7 year old is expected to understand. This task is still beyond the capability of today's computers and algorithms.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 193–203,
Seattle, Washington, USA, 18-21 October 2013.

By restricting the concept space, we gain the difficulty of being an open-domain problem, without the full complexity of the real world (for example, there will be no need for the machine to understand politics, technology, or to have any domain specific expertise). The multiple choice task avoids ambiguities (such as when the task is to find a sentence that best matches a question, as in some early reading comprehension tasks: see Section 2), and also avoids the need for additional grading, such as is needed in some TREC tasks. The stories were chosen to be fictional to focus work on finding the answer in the story itself, rather than in knowledge repositories such as Wikipedia; the goal is to build technology that actually understands stories and paragraphs on a deep level (as opposed to using information retrieval methods and the redundancy of the web to find the answers).

We chose to use crowd sourcing, as opposed to, for example, contracting teachers or paying for existing standardized tests, for three reasons, namely: (1) scalability, both for the sizes of datasets we can provide, and also for the ease of regularly refreshing the data; (2) for the variety in story-telling that having many different authors brings; and (3) for the free availability that can only result from providing non-copyrighted data. The content is freely available at http://research.microsoft.com/mct, and we plan to use that site to track published results and provide other resources, such as labels of various kinds.

2 Previous Work

The research goal of mapping text to meaning representations in order to solve particular tasks has a long history. DARPA introduced the Airline Travel Information System (ATIS) in the early 90's: there the task was to slot-fill flight-related information by modeling the intent of spoken language (see Tur et al., 2010, for a review). This data continues to be a used in the semantic modeling community (see, for example, Zettlemoyer and Collins, 2009). The Geoquery database contains 880 geographical facts about the US and has played a similar role for written (as opposed to spoken) natural language queries against a database (Zelle and Mooney, 1996) and it also continues to spur research (see for example Goldwasser et al., 2011), as does the similar Jobs database, which provides mappings of 640 sentences to a listing of jobs (Tang and Mooney, 2001). More recently, Zweig and Burges (2012) provided a set of 1040 sentences that comprise an SAT-style multiple choice sentence completion task.

The idea of using story-based reading comprehension questions to evaluate methods for machine reading itself goes back over a decade, when Hirschmann et al. (1999) showed that a bag of words approach, together with some heuristic linguistic modeling, could achieve 40% accuracy for the task of picking the sentence that best matches the query for "who / what / when / where / why" questions, on a small reading comprehension dataset from Remedia. This dataset spurred several research efforts, for example using reinforcement learning (Grois and Wilkins, 2005), named entity resolution (Harabagiu et al., 2003) and mapping questions and answers to logical form (Wellner et al., 2006). Work on story understanding itself goes back much further, to 1972, when Charniak proposed using a background model to answer questions about children's stories. Similarly, the TREC (and TAC) Question Answering tracks (e.g., Voorhees and Tice, 1999) aim to evaluate systems on their ability to answer factual questions such as "Where is the Taj Mahal". The QA4MRE task also aims to evaluate machine reading systems through question answering (e.g., Clark et al., 2012). Earlier work has also aimed at controlling the scope by limiting the text to children's stories: Breck et al. (2001) collected 75 stories from the Canadian Broadcasting Corporation's web site for children, and generated 650 questions for them manually, where each question was answered by a sentence in the text. Leidner et al. (2003) both enriched the CBC4kids data by adding several layers of annotation (such as semantic and POS tags), and measured QA performance as a function of question difficulty. For a further compendium of resources related to the story comprehension task, see Mueller (2010).

The task proposed here differs from the above work in several ways. Most importantly, the data collection is scalable: if the dataset proves sufficiently useful to others, it would be straightforward to gather an order of magnitude more. Even the dataset size presented here is an order of magnitude larger than the Remedia or the CBC4kids data and many times larger than QA4MRE. Second, the multiple choice task presents less ambiguity (and is consequently easier to collect data for) than the

James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.
One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.
His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.
After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

1) What is the name of the trouble making turtle?
A) Fries
B) Pudding
C) James
D) Jane

2) What did James pull off of the shelves in the grocery store?
A) pudding
B) fries
C) food
D) splinters

3) Where did James go after he went to the grocery store?
A) his deck
B) his freezer
C) a fast food restaurant
D) his room

4) What did James do after he ordered the fries?
A) went to the grocery store
B) went home without paying
C) ate them
D) made up his mind to be a better turtle

Figure 1. Sample Story and Questions (chosen randomly from MC500 train set).

task of finding the most appropriate sentence, and may be automatically evaluated. Further, our stories are fictional, which means that the information to answer the question is contained only in the story itself (as opposed to being able to directly leverage knowledge repositories such as Wikipedia). This design was chosen to focus the task on the machine understanding of short passages, rather than the ability to match against an existing knowledge base. In addition, while in the CBC4kids data each answer was a sentence from the story, here we required that approximately half of the questions require at least two sentences from the text to answer; being able to control complexity in this way is a further benefit of using multiple choice answers. Finally, as explained in Section 1, the use of free-form input makes the problem open domain (as opposed to the ATIS, Geoquery and Jobs data), leading to the hope that solutions to the task presented here will be easier to apply to novel, unrelated tasks.

3 Generating the Stories and Questions

Our aim was to generate a corpus of fictional story sets[1] that could be scaled with as little expert input as possible. Thus, we designed the process to be gated by cost, and keeping the costs low was a high priority. Crowd-sourcing seemed particularly appropriate, given the nature of the task, so we opted to use Amazon Mechanical Turk[2] (AMT). With over 500,000 workers[3], it provides the work force required to both achieve scalability and, equally importantly, to provide diversity in the stories and types of questions. We restricted our task to AMT workers (*workers*) residing in the United States. The average worker is 36 years old, more educated than the United States population in general (Paolacci et al., 2010), and the majority of workers are female.

3.1 The Story and Questions

Workers were instructed to write a short (150-300 words) fictional story, and to write as if for a child in grade school. The choice of 150-300 was made to keep the task an appropriate size for workers while still allowing for complex stories and questions. The workers were free to write about any topic they desired (as long as it was appropriate for a young child), and so there is a wide range, including vacations, animals, school, cars, eating, gardening, fairy tales, spaceships, and cowboys.

[1] We use the term "story set" to denote the fictional story together with its multiple choice questions, hypothetical answers, and correct answer labels.

[2] http://www.mturk.com

[3] https://requester.mturk.com/tour

Workers were also asked to provide four reading comprehension questions pertaining to their story and, for each, four multiple-choice answers. Coming up with incorrect alternatives (*distractors*) is a difficult task (see, e.g., Agarwal, 2011) but workers were requested to provide "reasonable" incorrect answers that at least include words from the story so that their solution is not trivial. For example, for the question "*What is the name of the dog?*", if only one of the four answers occurs in the story, then that answer must be the correct one.

Finally, workers were asked to design their questions and answers such that at least two of the four questions required multiple sentences from the story to answer them. That is, for those questions it should not be possible to find the answer in any individual sentence. The motivation for this was to ensure that the task could not be fully solved using lexical techniques, such as word matching, alone. Whilst it is still possible that a sophisticated lexical analysis could completely solve the task, requiring that answers be constructed from at least two different sentences in the story makes this much less likely; our hope is that the solution will instead require some inference and some form of limited reasoning. This hope rests in part upon the observation that standardized reading comprehension tests, whose goal after all is to test comprehension, generally avoid questions that can be answered by reading a single sentence.

3.2 Automatic Validation

Besides verifying that the story and all of the questions and answers were provided, we performed the following automatic validation before allowing the worker to complete the task:

Limited vocabulary: The lowercase words in the story, questions, and answers were stemmed and checked against a vocabulary list of approximately 8000 words that a 7-year old is likely to know (Kuperman et al., 2012). Any words not on the list were highlighted in red as the worker typed, and the task could not be submitted unless all of the words satisfied this vocabulary criterion. To allow the use of arbitrary proper nouns, capitalized words were not checked against the vocabulary list.

Multiple-sentence questions: As described earlier, we required that at least two of the questions need multiple sentences to answer. Workers were simply asked to mark whether a question needs one or multiple sentences and we required that at least two are marked as *multiple*.

3.3 The Workers

Workers were required to reside in the United States and to have completed 100 HITs with an over 95% approval rate[4]. The median worker took 22 minutes to complete the task. We paid workers $2.50 per story set and allowed each to do a maximum of 8 tasks (5 in MC500). We did not experiment with paying less, but this rate amounts to $6.82/hour, which is approximately the rate paid by other writing tasks on AMT at the time, though is also significantly higher than the median wage of $1.38 found in 2010 (Horton and Chilton, 2010). Workers could optionally leave feedback on the task, which was overwhelmingly positive – the most frequent non-stopword in the comments was "*fun*" and the most frequent phrase was "*thank you*". The only negative comments (in <1% of submissions) were when the worker felt that a particular word should have been on the allowed vocabulary list. Given the positive feedback, it may be possible to pay less if we collect more data in the future. We did not enforce story length constraints, but some workers interpreted our suggestion that the story be 150-300 words as a hard constraint, and some asked to be able to write a longer story.

The MCTest corpus contains two sets of stories, named MC160 and MC500, and containing 160 and 500 stories respectively. MC160 was gathered first, then some improvements were made before gathering MC500. We give details on the differences between these two sets below.

3.4 MC160: Manually Curated for Quality

In addition to the details described above, MC160 workers were given a target elementary grade school level (1-4) and a sample story matching that grade level[5]. The intent was to produce a set of stories and questions that varied in difficulty so that research work can progress grade-by-grade if needed. However, we found little difference between grades in the corpus..

After gathering the stories, we manually curated the MC160 corpus by reading each story set and

[4] The latter two are the default AMT requirements.

[5] From http://www.englishforeveryone.org/.

1. We went to visit the Smith's at their house.
2. I altered their suits for them.
3. You're car is very old.
4. Jim likes to run, hike, and going kayaking.
5. He should of come to work on time.
6. I think its best to wash lots of apples.
7. Are people who write "ping" thinking of submarines?
8. Smoke filled the room, making it hard to breathe.
9. Alert yet aloof - that's you.
10. They wanted they're money back.
11. Hawks and eagles like to fly high in the sky.
12. Don't let her wear them down.
13. The cat particularly liked the greasy plate.
14. The company is less successful because we have less employees.
15. The hamster belongs to Sam and I.
16. No one landed on the air strip today.
17. He was very effected by her tears.
18. You are a tired piece of toast, metaphorically speaking.
19. Anne plays bass and sings.
20. Him and me met at the park.

Figure 2. Grammar test for qualifying workers.

correcting errors. The most common mistakes were grammatical, though occasionally questions and/or answers needed to be fixed. 66% of the stories have at least one correction. We provide both the curated and original corpuses in order to allow research on reading comprehension in the presence of grammar, spelling, and other mistakes.

3.5 MC500: Adding a Grammar Test

Though the construction of MC160 was successful, it requires a costly curation process which will not scale to larger data sets (although the curation was useful, both for improving the design of MC500, and for assessing the effectiveness of automated curation techniques). To more fully automate the process, we added two more stages: (1) A grammar test that automatically pre-screens workers for writing ability, and (2) a second Mechanical Turk task whereby new workers take the reading comprehension tests and rate their quality. We will discuss stage (2) in the next section.

The grammar test consisted of 20 sentences, half of which had one grammatical error (see Figure 2). The incorrect sentences were written using common errors such as *you're* vs. *your*, using *'s* to indicate plurality, incorrect use of tense, *it's* vs. *its*, *less* vs. *fewer*, *I* vs. *me*, etc. Workers were required to indicate for each sentence whether it was grammatically correct or not, and had to pass with at least 80% accuracy in order to qualify for the task. The 80% threshold was chosen to trade off worker quality with the rate at which the tasks would be completed; initial experiments using a threshold of 90% indicated that collecting 500 stories would take many weeks instead of days. Note that each worker is allowed to write at most 5 stores, so we required at least 100 workers to pass the qualification test.

	Quality (1-5)	About animals
No Grammar Test	3.2	73%
Grammar Test	**4.3**	**30%**

Table 1. Pre-screening workers using a grammar test improves both quality and diversity of stories. Both differences are significant using the two-tailed t-test ($p<0.05$ for quality and $p<0.01$ for animals).

To validate the use of the qualification test, we gathered 30 stories requiring the test (*qual*) and 30 stories without. We selected a random set of 20 stories (10 from each), hid their origin, and then graded the overall quality of the story and questions from 1-5, meaning *do not attempt to fix*, *bad but rescuable*, *has non-minor problems*, *has only minor problems*, and *has no problems*, respectively. Results are shown in Table 1. The difference is statistically significant ($p<0.05$, using the two-tailed t-test). The *qual* stories were also more diverse, with fewer of them about animals (the most common topic).

Additional Modifications: Based on our experience curating MC160, we also made the following modifications to the task. In order to eliminate trivially-answerable questions, we required that each answer be unique, and that either the correct answer did not appear in the story or, if it did appear, that at least two of the incorrect answers also appeared in the story. This is to prevent questions that are trivially answered by checking which answer appears in the story. The condition on whether the correct answer appears is to allow questions such as "*How many candies did Susan eat?*", where the total may never appear in the story, even though the information needed to derive it does. An answer is considered to appear in the story if at least half (rounded down) of its non-stopword

terms appear in the story (ignoring word endings). This check is done automatically and must be satisfied before the worker is able to complete the task. Workers could also bypass the check if they felt it was incorrect, by adding a special term to their answer.

We were also concerned that the sample story might bias the workers when writing the story set, particularly when designing questions that require multiple sentences to answer. So, we removed the sample story and grade level from the task.

Finally, in order to encourage more diversity of stories, we added *creativity terms*, a set of 15 nouns chosen at random from the allowed vocabulary set. Workers were asked to "*please consider*" using one or more of the terms in their story, but use of the words was strictly optional. On average, workers used 3.9 of the creativity terms in their stories.

4 Rating the Stories and Questions

In this section we discuss the crowd-sourced rating of story sets. We wished to ensure story set quality despite the fact that MC500 was only minimally manually curated (see below). Pre-qualifying workers with a grammar test was one step of this process. The second step was to have additional workers on Mechanical Turk both evaluate each story and take its corresponding test. Each story was evaluated in this way by 10 workers, each of whom provided scores for each of age-appropriateness (*yes/maybe/no*), grammaticality (*few/some/many* errors), and story clarity (*excellent/reasonable/poor*). When answering the four reading comprehension questions, workers could also mark a question as "unclear". Each story set was rated by 10 workers who were each paid $0.15 per set.

Since we know the purportedly correct answer, we can estimate worker quality by measuring what fraction of questions that worker got right. Workers with less than 80% accuracy (ignoring those questions marked as unclear) were removed from the set. This constituted just 4.1% of the raters and 4.2% of the judgments (see Figure 3). Only one rater appeared to be an intentional spammer, answering 1056 questions with only 29% accuracy. The others primarily judged only one story. Only one worker fell between, answering 336 questions with just 75% accuracy.

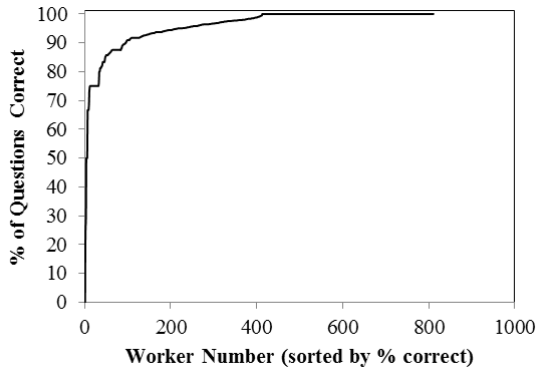


Figure 3. Just 4.1% of raters had an accuracy below 80% (constituting 4.2% of the judgments).

For the remaining workers (those who achieved at least 80% accuracy), we measured median story appropriateness, grammar, and clarity. For each category, stories for which less than half of the ratings were the best possible (e.g., *excellent* story clarity) were inspected and optionally removed from the data set. This required inspecting 40 (<10%) of the stories, only 2 of which were deemed poor enough to be removed (both of which had over half of the ratings all the way at the bottom end of the scale, indicating we could potentially have inspected many fewer stories with the same results). We also inspected questions for which at least 5 workers answered incorrectly, or answered "unclear". In total, 29 questions (<2%) were inspected. 5 were fixed by changing the question, 8 by changing the answers, 2 by changing both, 6 by changing the story, and 8 were left unmodified.

Note that while not fully automated, this process of inspecting stories and repairing questions took one person one day, so is still scalable to at least an order of magnitude more stories.

5 Dataset Analysis

In Table 2, we present results demonstrating the value of the grammar test and curation process. As expected, manually curating MC160 resulted in increased grammar quality and percent of questions answered correctly by raters. The goal of MC500 was to find a more scalable method to achieve the same quality as the curated MC160. As Table 2 shows, the grammar test improved story grammar quality from 1.70 to 1.77 (both uncurated). The rating and one-day curation process in-

Set	AgeAp	Clarity	Grammar	Correct
160	1.88	1.63	1.70	95.3
500	1.92	1.65	1.77	95.3
500 curated	1.94	**1.71**	1.79	**96.9**
160 curated	1.91	1.67	**1.84**‡	**97.7**

Table 2. Average age appropriateness, story clarity, grammar quality (0-2, with 2 being best), and percent of questions answered correctly by raters, for the original and curated versions of the data. Bold indicates statistical significance vs. the original version of the same set, using the two-sample t-test with unequal variance. The ‡ indicates the only statistical difference between 500 curated and 160 curated.

Corpus	Stories	Median writing time	Average Words Per:		
			Story	Question	Answer
MC160	160	26 min	204	8.0	3.4
MC500	500	20 min	212	7.7	3.4

Table 3. Corpus statistics for MC160 and MC500.

creases this to 1.79, whereas a fully manual curation results in a score of 1.84. Curation also improved the percent of questions answered correctly for both MC160 and MC500, but, unlike with grammar, there is no significant difference between the two curated sets. Indeed, the only statistically significant difference between the two is in grammar. So, the MC500 grammar test and curation process is a very scalable method for collecting stories of nearly the quality of the costly manual curation of MC160.

We also computed correlations between these measures of quality and various factors such as story length and time spent writing the story. On MC500, there is a mild correlation between a worker's grammar test score and the judged grammar quality of that worker's story (correlation of 0.24). Interestingly, this relation disappeared once MC500 was curated, likely due to repairing the stories with the worst grammar. On MC160, there is a mild correlation between the clarity and the number of words in the question and answer (0.20 and 0.18). All other correlations were below 0.15. These factors could be integrated into an estimate for age-appropriateness, clarity, and grammar, potentially reducing the need for raters.

Table 3 provides statistics on each corpus. MC160 and MC500 are similar in average number of words per story, question, and answer, as well as the median writing time. The most commonly used

Baseline Algorithms

Require: Passage P, set of passage words PW, i^{th} word in passage P_i, set of words in question Q, set of words in hypothesized answers $A_{1..4}$, and set of stop words U,
Define: $C(w) \coloneqq \sum_i \mathbb{I}(P_i = w)$;
Define: $IC(w) \coloneqq \log\left(1 + \frac{1}{C(w)}\right)$.

Algorithm 1 Sliding Window

for i = 1 to 4 **do**
$S = A_i \cup Q$
$$sw_i = \max_{j=1..|P|} \sum_{w=1..|S|} \begin{cases} IC(P_{j+w}) & if\ P_{j+w} \in S \\ 0 & otherwise \end{cases}$$
end for
return $sw_{1..4}$

Algorithm 2 Distance Based

for i = 1 to 4 **do**
$S_Q = (Q \cap PW) \setminus U$
$S_{A_i} = \left((A_i \cap PW) \setminus Q\right) \setminus U$
if $|S_Q| = 0$ or $|S_{A_i}| = 0$
$d_i = 1$
else
$d_i = \frac{1}{|P|-1} \min_{q \in S_Q, a \in S_{A_i}} d_P(q, a)$,
where $d_P(q, a)$ is the minimum number of words between an occurrence of q and an occurrence of a in P, plus one.
end if
end for
return $d_{1..4}$

Algorithm SW

Return $\arg\max_i sw_{1..4}$

Algorithm SW+D

Return $\arg\max_i sw_{1..4} - d_{1..4}$

Figure 4. The two lexical-based algorithms used for the baselines.

nouns in MC500 are: *day*, *friend*, *time*, *home*, *house*, *mother*, *dog*, *mom*, *school*, *dad*, *cat*, *tree*, and *boy*. The stories vary widely in theme. The first 10 stories of the randomly-ordered MC500 set are about: travelling to Miami to visit friends, waking up and saying hello to pets, a bully on a schoolyard, visiting a farm, collecting insects at Grandpa's house, planning a friend's birthday party, selecting clothes for a school dance, keeping animals from eating your ice cream, animals ordering food, and adventures of a boy and his dog.

MC160	Train and Dev: 400 Q's		Test: 240 Q's	
	SW	SW+D	SW	SW+D
Single	59.46	68.11	64.29	75.89
Multi	59.53	67.44	48.44	57.81
All	*59.50*	*67.75*	*55.83*	*66.25*

Table 4. Percent correct for the multiple choice questions for MC160. SW: sliding window algorithm. SW+D: combined results with sliding window and distance based algorithms. Single/Multi: questions marked by worker as requiring a single/multiple sentence(s) to answer. All differences between SW and SW+D are significant ($p<0.01$ using the two-tailed paired t-test).

MC500	Train and Dev: 1400 Q's		Test: 600 Q's		All
	SW	SW+D	SW	SW+D	SW+D
Single	55.13	61.77	51.10	57.35	60.44
Multi	49.80	55.28	51.83	56.10	55.53
All	*52.21*	*58.21*	*51.50*	*56.67*	*57.75*

Table 5. Percent correct for the multiple choice questions for MC500, notation as above. All differences between SW and SW+D are significant ($p<0.01$, tested as above).

We randomly divided MC160 and MC500 into train, development, and test sets of 70, 30, and 60 stories and 300, 50, and 150 stories, respectively.

6 Baseline System and Results

We wrote two baseline systems, both using only simple lexical features. The first system used a sliding window, matching a bag of words constructed from the question and hypothesized answer to the text. Since this ignored long range dependencies, we added a second, word-distance based algorithm. The distance-based score was simply subtracted from the window-based score to arrive at the final score (we tried scaling the distance score before subtraction but this did not improve results on the MC160 train set). The algorithms are summarized in Figure 4. A coin flip is used to break ties. The use of inverse word counts was inspired by TF-IDF.

Results for MC160 and MC500 are shown in Table 4 and Table 5. The MC160 train and development sets were used for tuning. The baseline algorithm was authored without seeing any portion of MC500, so both the MC160 test set and all of MC500 were used for testing (although we nevertheless report results on the train/test split). Note that adding the distance based algorithm improved accuracy by approximately 10% absolute on MC160 and approximately 6% on MC500. Overall, error rates on MC500 are higher than on MC160, which agrees with human performance (see Table 2), suggesting that MC500's questions are more difficult.

	MC160 Test	MC500 Test
Baseline (SW+D)	66.25	56.67
RTE	59.79‡	53.52
Combined	67.60	60.83‡

Table 6. Percent correct for MC160 and MC500 test sets. The ‡ indicates statistical significance vs. baseline ($p<0.01$ using the two-tailed paired t-test). MC160 combined vs. baseline has p-value 0.063.

7 Recognizing Textual Entailment Results

We also tried using a "recognizing textual entailment" (RTE) system to answer MCTest questions. The goal of RTE (Dagan et al., 2005) is to determine whether a given statement can be inferred from a particular text. We can cast MCTest as an RTE task by converting each question-answer pair into a statement, and then selecting the answer whose statement has the highest likelihood of being entailed by the story. For example, in the sample story given in Figure 1, the second question can be converted into four statements (one for each answer), and the RTE system should select the statement "James pulled pudding off of the shelves in the grocery store" as the most likely one.

For converting question-answer pairs to statements, we used the rules employed in a web-based question answering system (Cucerzan and Agichtein, 2005). For RTE, we used BIUTEE (Stern and Dagan, 2011), which performs better than the median system in the past four RTE competitions. We ran BIUTEE both in its default configuration, as well as with its optional additional data sources (FrameNet, ReVerb, DIRT, and others as found on the BIUTEE home page). The default configuration performed better so we present its results here. The results in Table 6 show that the RTE method performed worse than the baseline.

We also combined the baseline and RTE system by training BIUTEE on the train set and using the development set to optimize a linear combination of BIUTEE with the baseline; the combined system outperforms either component system on MC500.

It is possible that with some tuning, an RTE system will outperform our baseline system. Nevertheless, these RTE results, and the performance of the baseline system, both suggest that the reading comprehension task described here will not be trivially solved by off-the-shelf techniques.

8 Making Data and Results an Ongoing Resource

Our goal in constructing this data is to encourage research and innovation in the machine comprehension of text. Thus, we have made both MC160 and MC500 freely available for download at http://research.microsoft.com/mct. To our knowledge, these are the largest copyright-free reading comprehension data sets publicly available. To further encourage research on these data, we will be continually updating the webpage with the best-known published results to date, along with pointers to those publications.

One of the difficulties in making progress on a particular task is implementing previous work in order to apply improvements to it. To mitigate this difficulty, we are encouraging researchers who use the data to (optionally) provide per-answer scores from their system. Doing so has three benefits: (a) a new system can be measured in the context of the errors made by the previous systems, allowing each research effort to incrementally add useful functionality without needing to also re-implement the current state-of-the-art; (b) it allows system performance to be measured using paired statistical testing, which will substantially increase the ability to determine whether small improvements are significant; and (c) it enables researchers to perform error analysis on any of the existing systems, simplifying the process of identifying and tackling common sources of error. We will also periodically ensemble the known systems using standard machine learning techniques and make those results available as well (unless the existing state-of-the-art already does such ensembling).

The released data contains the stories and questions, as well as the results from workers who rated the stories and took the tests. The latter may be used, for example, to measure machine performance vs. human performance on a per-question basis (i.e., does your algorithm make similar mistakes to humans?), or vs. the judged clarity of each story. The ratings, as well as whether a question needs multiple sentences to answer, should typically only be used in evaluation, since such information is not generally available for most text. We will also provide an anonymized author id for each story, which could allow additional research such as using other works by the same author when understanding a story, or research on authorship attribution (e.g., Stamatatos, 2009).

9 Future Work

We plan to use this dataset to evaluate approaches for machine comprehension, but are making it available now so that others may do the same. If MCTest is used we will collect more story sets and will continue to refine the collection process. One interesting research direction is ensuring that the questions are difficult enough to challenge state-of-the-art techniques as they develop. One idea for this is to apply existing techniques automatically during story set creation to see whether a question is too easily answered by a machine. By requiring authors to create difficult questions, each data set will be made more and more difficult (but still answerable by humans) as the state-of-the-art methods advance. We will also experiment with timing the raters as they answer questions to see if we can find those that are too easy for people to answer. Removing such questions may increase the difficulty for machines as well. Additionally, any divergence between how easily a person answers a question vs. how easily a machine does may point toward new techniques for improving machine comprehension; we plan to conduct research in this direction as well as make any such data available for others.

10 Conclusion

We present the MCTest dataset in the hope that it will help spur research into the machine comprehension of text. The metric (the accuracy on the question sets) is clearly defined, and on that metric, lexical baseline algorithms only attain approximately 58% correct on test data (the MC500 set) as

opposed to the 100% correct that the majority of crowd-sourced judges attain. A key component of MCTest is the scalable design: we have shown that data whose quality approaches that of expertly curated data can be generated using crowd sourcing coupled with expert correction of worker-identified errors. Should MCTest prove useful to the community, we will continue to gather data, both to increase the corpus size, and to keep the test sets fresh. The data is available at http://research.microsoft.com/mct and any submitted results will be posted there too. Because submissions will be requested to include the score for each test item, researchers will easily be able to compare their systems with those of others, and investigation of ensembles comprised of components from several different teams will be straightforward. MCTest also contains supplementary material that researchers may find useful, such as worker accuracies on a grammar test and crowd-sourced measures of the quality of their stories.

Acknowledgments

We would like to thank Silviu Cucerzan and Lucy Vanderwende for their help with converting questions to statements and other useful discussions.

References

M. Agarwal and P. Mannem. 2011. Automatic Gap-fill Question Generation from Text Books. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, 56–64.

E. Breck, M. Light, G.S.Mann, E. Riloff, B. Brown, P. Anand, M. Rooth M. Thelen. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the workshop on Open-domain question answering*, 12, 1-8.

E. Charniak. 1972. Toward a Model of Children's Story Comprehension. *Technical Report*, 266, MIT Artificial Intelligence Laboratory, Cambridge, MA.

P. Clark, P. Harrison, and X. Yao. An Entailment-Based Approach to the QA4MRE Challenge. 2012. In *Proceedings of the Conference and Labs of the Evaluation Forum (CLEF) 2012*.

S. Cucerzan and E. Agichtein. 2005. Factoid Question Answering over Unstructured and Structured Content on the Web. In *Proceedings of the Fourteenth Text Retrieval Conference (TREC)*.

I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, F. d'Alché-Buc (Eds.), Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944, pp. 177-190, Springer.

D. Goldwasser, R. Reichart, J. Clarke, D. Roth. 2011. Confidence Driven Unsupervised Semantic Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 1486-1495.

E. Grois and D.C. Wilkins. 2005. Learning Strategies for Story Comprehension: A Reinforcement Learning Approach. In *Proceedings of the Twenty Second International Conference on Machine Learning*, 257-264.

S.M. Harabagiu, S.J. Maiorano, and M.A. Pasca. 2003. Open-Domain Textual Question Answering Techniques. *Natural Language Engineering*, 9(3):1-38. Cambridge University Press, Cambridge, UK.

L. Hirschman, M. Light, E. Breck, and J.D. Burger. 1999. Deep Read: A Reading Comprehension System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, 325-332.

J. Horton and L. Chilton. 2010. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, 209-218.

V. Kuperman, H. Stadthagen-Gonzalez, M. Brysbaert. 2012. Age-of-acquisition ratings for 30,000 English words. *Behavior Research Methods*, 44(4):978-990.

J.L. Leidner, T. Dalmas, B. Webber, J. Bos, C. Grover. 2003. Automatic Multi-Layer Corpus Annotation for Evaluating Question Answering Methods: CBC4Kids. In *Proceedings of the 3rd International Workshop on Linguistically Interpreted Corpora*.

E.T. Mueller. 2010. Story Understanding Resources. *http://xenia.media.mit.edu/~mueller/storyund/storyres.html*.

G. Paolacci, J. Chandler, and P. Iperirotis. 2010. Running experiments on Amazon Mechanical Turk. *Judgment and Decision Making*. 5(5):411-419.

E. Stamatatos. 2009. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci.*, 60:538–556.

A. Stern and I. Dagan. 2011. A Confidence Model for Syntactically-Motivated Entailment Proofs. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*.

L.R. Tang and R.J. Mooney. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, 466-477.

G. Tur, D. Hakkani-Tur, and L.Heck. 2010. What is left to be understood in ATIS? *Spoken Language Technology Workshop*, 19-24.

E.M. Voorhees and D.M. Tice. 1999. The TREC-8 Question Answering Track Evaluation. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*.

B. Wellner, L. Ferro, W. Greiff, and L. Hirschman. 2005. Reading comprehension tests for computer-based understand evaluation. *Natural Language Engineering*, 12(4):305-334. Cambridge University Press, Cambridge, UK.

J.M. Zelle and R.J. Mooney. 1996. Learning to Parse Database Queries using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI)*, 1050-1055.

L.S. Zettlemoyer and M. Collins. 2009. Learning Context-Dependent Mappings from Sentences to Logical Form. In *Proceedings of the 47th Annual Meeting of the Association for Computation Linguistics (ACL)*, 976-984.

G. Zweig and C.J.C. Burges. 2012. A Challenge Set for Advancing Language Modeling. In *Proceedings of the Workshop on the Future of Language Modeling for HLT, NAACL-HLT*.

Noise-aware Character Alignment for Bootstrapping Statistical Machine Transliteration from Bilingual Corpora

Katsuhito Sudoh[*†] **Shinsuke Mori**[‡] **Masaaki Nagata**[*]
[*]NTT Communication Science Laboratories
[†]Graduate School of Informatics, Kyoto University
[‡]Academic Center for Computing and Media Studies, Kyoto University
sudoh.katsuhito@lab.ntt.co.jp

Abstract

This paper proposes a novel noise-aware character alignment method for bootstrapping statistical machine transliteration from automatically extracted phrase pairs. The model is an extension of a Bayesian many-to-many alignment method for distinguishing non-transliteration (noise) parts in phrase pairs. It worked effectively in the experiments of bootstrapping Japanese-to-English statistical machine transliteration in patent domain using patent bilingual corpora.

1 Introduction

Transliteration is used for providing translations for source language words that have no appropriate counterparts in target language, such as some technical terms and named entities. Statistical machine transliteration (Knight and Graehl, 1998) is a technology to solve it in a statistical manner. Bilingual dictionaries can be used to train its model, but many of their entries are actually *translation* but not *transliteration*. Such non-transliteration pairs hurt the transliteration model and should be eliminated beforehand.

Sajjad et al. (2012) proposed a method to identify such non-transliteration pairs, and applied it successfully to *noisy* word pairs obtained from automatic word alignment on bilingual corpora. It enables the statistical machine transliteration to be bootstrapped from bilingual corpora. This approach is beneficial because it does not require carefully-developed bilingual transliteration dictionaries and it can learn domain-specific transliteration patterns from bilingual corpora in the target domain. However, their transliteration mining approach is sample-wise; that is, it makes a decision whether a bilingual phrase pair is transliteration or not. Suppose that a compound word in a language A is transliterated into two words in another language B. Their correspondence may not be fully identified by automatic word alignment and a wrong alignment between the compound word in A and only one component word in B is found. The sample-wise mining cannot make a correct decision of *partial transliteration* on the aligned candidate, and may introduces noise to the statistical transliteration model.

This paper proposes a novel transliteration mining method for such partial transliterations. The method uses a noise-aware character alignment model that distinguish non-transliteration (noise) parts from transliteration (signal) parts. The model is an extension of a Bayesian alignment model (Finch and Sumita, 2010) and can be trained by a sampling algorithm extended for a constraint on noise. Our experiments of Japanese-to-English transliteration achieved 16% relative error reduction in transliteration accuracy from the sample-wise method. The main contribution of this paper is two-fold:

- we formulate alignment over string pairs with partial noise and present a solution with a noise-aware alignment model;
- we proved its effectiveness by experiments with frequent unknown words in actual Japanese-to-English patent translation data.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 204–209,
Seattle, Washington, USA, 18-21 October 2013.

2 Bayesian many-to-many alignment

We briefly review a Bayesian many-to-many character alignment proposed by Finch and Sumita (2010) on which our model is based. The model is based on a generative process of bilingual substring pairs $\langle \bar{s}, \bar{t} \rangle$ by the following Dirichlet process (DP):

$$\begin{aligned} G|_{\alpha, G_0} &\sim \mathrm{DP}(\alpha, G_0) \\ \langle \bar{s}, \bar{t} \rangle | G &\sim G, \end{aligned}$$

where G is a probability distribution over substring pairs according to a DP prior with base measure G_0 and hyperparameter α. G_0 is modeled as a joint spelling model as follows:

$$G_0\left(\langle \bar{s}, \bar{t} \rangle\right) = \frac{\lambda_s^{|\bar{s}|}}{|\bar{s}|!} e^{-\lambda_s} v_s^{-|\bar{s}|} \times \frac{\lambda_t^{|\bar{t}|}}{|\bar{t}|!} e^{-\lambda_t} v_t^{-|\bar{t}|}. \quad (1)$$

This is a simple joint probability of the spelling models, in which each alphabet appears based on a uniform distribution over the vocabulary (of size v_s and v_t) and each string length follows a Poisson distribution (with the average length λ_s and λ_t).

The model handles infinite number of substring pairs according to the Chinese Restaurant Process (CRP). The probability of a substring pair $\langle \bar{s}_k, \bar{t}_k \rangle$ is based on the counts of all other substring pairs as follows:

$$\begin{aligned} &p\left(\langle \bar{s}_k, \bar{t}_k \rangle | \left\{\langle \bar{s}, \bar{t} \rangle\right\}_{-k}\right) \\ &\qquad = \frac{N\left(\langle \bar{s}_k, \bar{t}_k \rangle\right) + \alpha G_0\left(\langle \bar{s}_k, \bar{t}_k \rangle\right)}{\sum_i N\left(\langle \bar{s}_i, \bar{t}_i \rangle\right) + \alpha}. \quad (2) \end{aligned}$$

Here $\left\{\langle \bar{s}, \bar{t} \rangle\right\}_{-k}$ means a set of substring pairs excluding $\langle \bar{s}_k, \bar{t}_k \rangle$, and $N\left(\langle \bar{s}_k, \bar{t}_k \rangle\right)$ is the number of $\langle \bar{s}_k, \bar{t}_k \rangle$ in the current sample space. This alignment model is suitable for representing very sparse distribution over arbitrary substring pairs, thanks to reasonable CRP-based smoothing for unseen pairs based on the spelling model.

3 Proposed method

We propose an extended many-to-many alignment model that can handle partial noise. We extend the model in the previous section by introducing a noise symbol and state-based probability calculation.

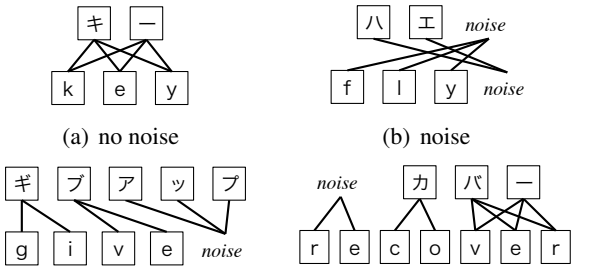


Figure 1: Three types of noise in transliteration data. Solid lines are correct many-to-many alignment links.

3.1 Partial noise in transliteration data

Figure 1 shows transliteration examples with "no noise," "noise," and "partial noise." Solid lines in the figure show correct many-to-many alignment links. The examples (a) and (b) can be distinguished effectively by Sajjad et al. (2012). We aim to do alignment as in the examples (c) and (d) by distinguishing its non-transliteration (noise) part, which cannot be handled by the existing methods.

3.2 Noise-aware alignment model

We introduce a *noise symbol* to handle partial noise in the many-to-many alignment model. Htun et al. (2012) extended the many-to-many alignment for the sample-wise transliteration mining, but its noise model only handles the sample-wise noise and cannot distinguish partial noise. We model partial noise in the CRP-based joint substring model.

Partial noise in transliteration data typically appears in compound words as mentioned earlier, because their counterparts consisting of two or more words may not be fully covered in automatically extracted words and phrases as shown in Figure 1(c). Another type of partial noise is derived from morphological differences due to inflection, which usually appear in the sub-word level as prefixes and suffixes as shown in Figure 1(d). According to this intuition, we assume that partial noise appears in the beginning and/or end of transliteration data (in case of sample-wise noise, we assume the noise is in the beginning). This assumption derives a constraint between signal and noise parts that helps to avoid a welter of transliteration and non-transliteration parts. It also has a shortcoming that it is generally

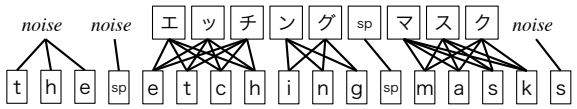


Figure 2: Example of many-to-many alignment with partial noise in the beginning and end. "*noise*" stands for the noise symbol and "sp" stands for a white space.

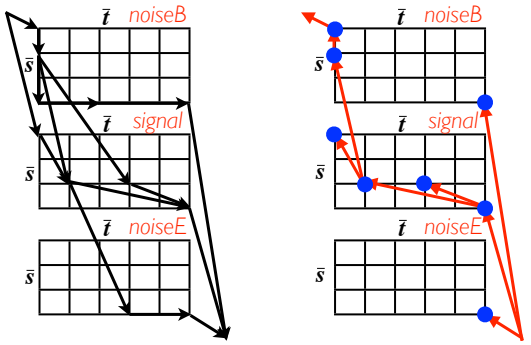
not appropriate for noise in the middle, but handling arbitrary number of noise parts increases computational complexity and sparseness. We rely on this simple assumption in this paper and consider a more complex mid-noise problem as future work.

Figure 2 shows a partial noise example in both the beginning and end. This example is actually correct translation but includes noise in a sense of transliteration; an article "the" is wrongly included in the phrase pair (no articles are used in Japanese) and a plural noun "masks" is transliterated into "マスク"(mask). These non-transliteration parts are aligned to noise symbols in the proposed model. The noise symbols are treated as zero-length substrings in the model, same as other substrings.

3.3 Constrained Gibbs sampling

Finch and Sumita (2010) used a blocked Gibbs sampling algorithm with forward-filtering backward-sampling (FFBS) (Mochihashi et al., 2009). We extend their algorithm for our noise-aware model using a state-based calculation over the three states: non-transliteration part in the beginning (noiseB), transliteration part (signal), non-transliteration part in the end (noiseE).

Figure 3 illustrates our FFBS steps. At first in the forward filtering, we begin with transition to noiseB and signal. The calculation of forward probabilities itself is almost the same as Finch and Sumita (2010) except for state transition constraints: from noiseB to signal, from signal to noiseE. The backward-sampling traverses a path by probability-based sampling with true posteriors, starting from the choice of the ending state among noiseB (means full noise), signal, and noiseE. This algorithm increases the computational cost by three times to consider three different states, compared to that of Finch and Sumita (2010).



(a) Forward filtering (b) Backward sampling

Figure 3: State-based FFBS for the proposed model.

4 Experiments

We conducted experiments comparing the proposed method with the conventional sample-wise method for the use in bootstrapping statistical machine transliteration using Japanese-to-English patent translation dataset (Goto et al., 2013).

4.1 Training data setup

First, we trained a phrase table on the 3.2M parallel sentences by a standard training procedure using Moses, with Japanese tokenization using MeCab[1]. We obtained 591,840 phrase table entries whose Japanese side was written in *katakana* (Japanese phonogram) only[2]. Then, we iteratively ran the method of Sajjad et al. (2012) on these entries and eliminate non-transliteration pairs, until the number of pairs converged. Finally we obtain 104,563 *katakana*-English pairs after 10 iterations; they were our *baseline training set* mined by sample-wise method. We used Sajjad et al.'s method as preprocessing for filtering sample-wise noise while the proposed method could also do that, because the proposed method took much more training time for all phrase table entries.

4.2 Transliteration experiments

The transliteration experiment used a translation-based implementation with Moses, using a

[1]http://code.google.com/p/mecab/

[2]This *katakana*-based filtering is a language dependent heuristic for choosing potential transliteration candidate, because transliterations in Japanese are usually written in *katakana*.

character-based 7-gram language model trained on 300M English patent sentences. We compared three transliteration models below.

The test set was top-1000 unknown (in the Japanese-to-English translation model) *katakana* words appeared in 400M Japanese patent sentences. They covered 15.5% of all unknown *katakana* words and 8.8% of all unknown words (excluding numbers); that is, more than a half of unknown words were *katakana* words.

4.2.1 Sample-wise method (BASELINE)

We used the baseline training set to train statistical machine transliteration model for our baseline. The training procedure was based on Moses: MGIZA++ word alignment, grow-diag-final-and alignment symmetrization and phrase extraction with the maximum phrase length of 7.

4.2.2 Proposed method (PROPOSED)

We applied the proposed method to the baseline training set with 30 sampling iterations and eliminated partial noise. The transliteration model was trained in the same manner as BASELINE after eliminating noise.

The hyperparameters, α, λ_s, and λ_t, were optimized using a held-out set of 2,000 *katakana*-English pairs that were randomly chosen from a general-domain bilingual dictionary. The hyperparameter optimization was based on F-score values on the held-out set with varying α among 0.01, 0.02, 0.05, 0.1, 1.0, and λs among 1, 2, 3, 5.

Table 1 compares the statistics on the training sets of BASELINE and PROPOSED. Note that we applied the proposed method to BASELINE data (the sample-wise method was already applied until convergence). The proposed method eliminated only two transliteration candidates in sample-wise but also eliminated 5,714 (0.64%) *katakana* and 55,737 (4.1%) English characters[3].

4.2.3 Proposed method using aligned joint substrings as phrases (PROPOSED-JOINT)

The many-to-many character alignment actually induces substring pairs, which can be used as phrases in statistical machine transliteration and improved transliteration performance (Finch and Sumita, 2010). We extracted them by: 1) generate many-to-many word alignment, in which all possible word alignment links in many-to-many correspondences (e.g., 0-0 0-1 0-2 1-0 1-1 1-2 for ⟨コ ン, c o m⟩), 2) run phrase extraction and scoring same as a standard Moses training. This procedure extracts longer phrases satisfying the many-to-many alignment constraints than the simple use of extracted joint substring pairs as phrases.

[3]The reason of larger number of partial noise in English side would be a syntactic difference as shown in Figure 2 and the *katakana*-based filtering heuristics.

Table 1: Statistics of the training sets.

Method	#pairs	#Ja chars.	#En chars.
BASELINE	104,563	899,080	1,372,993
PROPOSED	104,561	893,366	1,317,256

4.3 Results

Table 2 shows the results. We used three evaluation metrics: ACC, F-score, and BLEU_c. ACC is a sample-wise accuracy and F-score is a character-wise F-measure-like score (Li et al., 2010). BLEU_c is BLEU (Papineni et al., 2002) in the character level with n=4.

PROPOSED achieved 63% in ACC (16% relative error reduction from BASELINE), and 94.6% in F-score (25% relative error reduction from BASELINE). These improvements clearly showed an advantage of the proposed method over the sample-wise mining. BLEU_c showed a similar improvements. Recall that BASELINE and PROPOSED had a small difference in their training data, actually 0.64% (*katakana*) and 4.1% (English) in the number of characters. The results suggest that the partial noise can hurt transliteration models.

PROPOSED-JOINT showed similar performance as PROPOSED with a slight drop in BLEU_c, although many-to-many substring alignment was expected to improve transliteration as reported by Finch and Sumita (2010). The difference may be due to the difference in coverage of the phrase tables; PROPOSED-JOINT retained relatively long substrings by the many-to-many alignment constraints in contrast to the less-constrained grow-diag-final-and alignments in PROPOSED. Since the training data in our bootstrapping experiments con-

Table 2: Japanese-to-English transliteration results for top-1000 unknown *katakana* words. ACC and F-score stand for the ones used in NEWS workshop, BLEU_c is character-wise BLEU.

Method	ACC	F-score	BLEU_c
BASELINE	0.56	0.929	0.864
PROPOSED	0.63	0.946	0.897
PROPOSED-JOINT	0.63	0.943	0.888

tained many similar phrases unlike dictionary-based data in Finch and Sumita (2010), the phrase table of PROPOSED-JOINT may have a small coverage due to long and sparse substring pairs with large probabilities even if the many-to-many alignment was good. This sparseness problem is beyond the scope of this paper and worth further study.

4.4 Alignment Examples

Figure 4 shows examples of the alignment results in the training data. As expected, partial noise both in Japanese and English was identified correctly in (a), (b), and (c). There were some alignment errors in the signal part in (b), in which characters in boundary positions were aligned incorrectly to adjacent substrings. These alignment errors did not directly degrade the partial noise identification but may cause a negative effect on overall alignment performance in the sampling-based optimization. (d) is a negative example in which partial noise was incorrectly aligned. (c) and (d) have similar partial noise in their English word endings, but it could not be identified in (d). One possible reason for that is the sparseness problem mentioned above, as shown in erroneous long character alignments in (d).

5 Conclusion

This paper proposed a noise-aware many-to-many alignment model that can distinguish partial noise in transliteration pairs for bootstrapping statistical machine transliteration model from automatically extracted phrase pairs. The model and training algorithm are straightforward extension of those by Finch and Sumita (2010). The proposed method was proved to be effective in Japanese-to-English transliteration experiments in patent domain.

Future work will investigate the proposed method

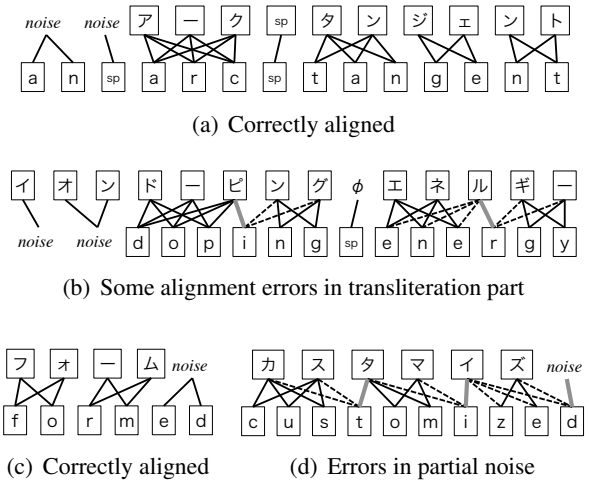


Figure 4: Examples of noise-aware many-to-many alignment in the training data. ϕ stands for a zero-length substring. Dashed lines show incorrect alignments, and bold grey lines mean their corrections.

in other domains and language pairs. The partial noise would appear in other language pairs, typically between agglutinative and non-agglutinative languages. It is also worth extending the approach into word alignment in statistical machine translation.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments and suggestions.

References

Andrew Finch and Eiichiro Sumita. 2010. A Bayesian Model of Bilingual Segmentation for Transliteration. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 259–266.

Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K. Tsou. 2013. Overview of the Patent Machine Translation Task at the NTCIR-10 Workshop. In *The 10th NTCIR Conference*, June.

Ohnmar Htun, Andrew Finch, Eiichiro Sumita, and Yoshiki Mikami. 2012. Improving Transliteration Mining by Integrating Expert Knowledge with Statistical Approaches. *International Journal of Computer Applications*, 58(17):12–22, November.

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

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2010. Whitepaper of NEWS 2010 Shared Task on Transliteration Generation. In *Proceedings of the 2010 Named Entities Workshop*, pages 12–20, Uppsala, Sweden, July. Association for Computational Linguistics.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore, August. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2012. A statistical model for unsupervised and semi-supervised transliteration mining. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 469–477, Jeju Island, Korea, July. Association for Computational Linguistics.

Optimal Beam Search for Machine Translation

Alexander M. Rush **Yin-Wen Chang**
MIT CSAIL,
Cambridge, MA 02139, USA
{srush, yinwen}@csail.mit.edu

Michael Collins
Department of Computer Science,
Columbia University,
New York, NY 10027, USA
mcollins@cs.columbia.edu

Abstract

Beam search is a fast and empirically effective method for translation decoding, but it lacks formal guarantees about search error. We develop a new decoding algorithm that combines the speed of beam search with the optimal certificate property of Lagrangian relaxation, and apply it to phrase- and syntax-based translation decoding. The new method is efficient, utilizes standard MT algorithms, and returns an exact solution on the majority of translation examples in our test data. The algorithm is 3.5 times faster than an optimized incremental constraint-based decoder for phrase-based translation and 4 times faster for syntax-based translation.

1 Introduction

Beam search (Koehn et al., 2003) and cube pruning (Chiang, 2007) have become the *de facto* decoding algorithms for phrase- and syntax-based translation. The algorithms are central to large-scale machine translation systems due to their efficiency and tendency to produce high-quality translations (Koehn, 2004; Koehn et al., 2007; Dyer et al., 2010). However despite practical effectiveness, neither algorithm provides any bound on possible decoding error.

In this work we present a variant of beam search decoding for phrase- and syntax-based translation. The motivation is to exploit the effectiveness and efficiency of beam search, but still maintain formal guarantees. The algorithm has the following benefits:

- In theory, it can provide a certificate of optimality; in practice, we show that it produces optimal hypotheses, with certificates of optimality, on the vast majority of examples.
- It utilizes well-studied algorithms and extends off-the-shelf beam search decoders.
- Empirically it is very fast, results show that it is 3.5 times faster than an optimized incremental constraint-based solver.

While our focus is on fast decoding for machine translation, the algorithm we present can be applied to a variety of dynamic programming-based decoding problems. The method only relies on having a constrained beam search algorithm and a fast unconstrained search algorithm. Similar algorithms exist for many NLP tasks.

We begin in Section 2 by describing constrained hypergraph search and showing how it generalizes translation decoding. Section 3 introduces a variant of beam search that is, in theory, able to produce a certificate of optimality. Section 4 shows how to improve the effectiveness of beam search by using weights derived from Lagrangian relaxation. Section 5 puts everything together to derive a fast beam search algorithm that is often optimal in practice.

Experiments compare the new algorithm with several variants of beam search, cube pruning, A* search, and relaxation-based decoders on two translation tasks. The optimal beam search algorithm is able to find exact solutions with certificates of optimality on 99% of translation examples, significantly more than other baselines. Additionally the optimal

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 210–221,
Seattle, Washington, USA, 18-21 October 2013.

beam search algorithm is much faster than other exact methods.

2 Background

The focus of this work is decoding for statistical machine translation. Given a source sentence, the goal is to find the target sentence that maximizes a combination of translation model and language model scores. In order to analyze this decoding problem, we first abstract away from the specifics of translation into a general form, known as a hypergraph. In this section, we describe the hypergraph formalism and its relation to machine translation.

2.1 Notation

Throughout the paper, scalars and vectors are written in lowercase, matrices are written in uppercase, and sets are written in script-case, e.g. $\mathcal{X}$. All vectors are assumed to be column vectors. The function $\delta(j)$ yields an indicator vector with $\delta(j)_j = 1$ and $\delta(j)_i = 0$ for all $i \neq j$.

2.2 Hypergraphs and Search

A directed hypergraph is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1 \dots |\mathcal{V}|\}$ is a set of vertices, and $\mathcal{E}$ is a set of directed hyperedges. Each hyperedge $e \in \mathcal{E}$ is a tuple $\left\langle \langle v_2, \dots, v_{|v|} \rangle, v_1 \right\rangle$ where $v_i \in \mathcal{V}$ for $i \in \{1 \dots |v|\}$. The *head* of the hyperedge is $h(e) = v_1$. The *tail* of the hyperedge is the ordered sequence $t(e) = \langle v_2, \dots, v_{|v|} \rangle$. The size of the tail $|t(e)|$ may vary across different hyperedges, but $|t(e)| \geq 1$ for all edges and is bounded by a constant. A directed graph is a directed hypergraph with $|t(e)| = 1$ for all edges $e \in \mathcal{E}$.

Each vertex $v \in \mathcal{V}$ is either a *non-terminal* or a *terminal* in the hypergraph. The set of non-terminals is $\mathcal{N} = \{v \in \mathcal{V} : h(e) = v \text{ for some } e \in \mathcal{E}\}$. Conversely, the set of terminals is defined as $\mathcal{T} = \mathcal{V} \setminus \mathcal{N}$.

All directed hypergraphs used in this work are acyclic: informally this implies that no hyperpath (as defined below) contains the same vertex more than once (see Martin et al. (1990) for a full definition). Acyclicity implies a partial topological ordering of the vertices. We also assume there is a distinguished *root* vertex 1 where for all $e \in \mathcal{E}$, $1 \notin t(e)$.

Next we define a hyperpath as $x \in \{0, 1\}^{|\mathcal{E}|}$ where $x(e) = 1$ if hyperedge e is used in the hyperpath, $x(e) = 0$ otherwise. The set of valid hyperpaths is defined as

$$\mathcal{X} = \left\{ x : \begin{array}{l} \displaystyle\sum_{e \in \mathcal{E}: h(e) = 1} x(e) = 1, \\ \\ \displaystyle\sum_{e: h(e) = v} x(e) = \sum_{e: v \in t(e)} x(e) \quad \forall\, v \in \mathcal{N} \setminus \{1\} \end{array} \right\}$$

The first problem we consider is *unconstrained* hypergraph search. Let $\theta \in R^{|\mathcal{E}|}$ be the weight vector for the hypergraph and let $\tau \in \mathbb{R}$ be a weight offset.[1] The unconstrained search problem is to find

$$\max_{x \in \mathcal{X}} \sum_{e \in \mathcal{E}} \theta(e) x(e) + \tau = \max_{x \in \mathcal{X}} \theta^\top x + \tau$$

This maximization can be computed for any weights and directed acyclic hypergraph in time $O(|\mathcal{E}|)$ using dynamic programming. Figure 1 shows this algorithm which is simply a version of the CKY algorithm.

procedure BESTPATHSCORE(θ, τ)
  $\pi[v] \leftarrow 0$ for all $v \in \mathcal{T}$
  for $e \in \mathcal{E}$ in topological order **do**
    $\langle \langle v_2, \dots, v_{|v|} \rangle, v_1 \rangle \leftarrow e$
    $s \leftarrow \theta(e) + \sum_{i=2}^{|v|} \pi[v_i]$
    if $s > \pi[v_1]$ **then** $\pi[v_1] \leftarrow s$
  return $\pi[1] + \tau$

Figure 1: Dynamic programming algorithm for unconstrained hypergraph search. Note that this version only returns the highest score: $\max_{x \in \mathcal{X}} \theta^\top x + \tau$. The optimal hyperpath can be found by including back-pointers.

Next consider a variant of this problem: *constrained* hypergraph search. Constraints will be necessary for both phrase- and syntax-based decoding. In phrase-based models, the constraints will ensure that each source word is translated exactly once. In syntax-based models, the constraints will be used to intersect a translation forest with a language model.

In the constrained hypergraph problem, hyperpaths must fulfill additional linear hyperedge constraints. Define the set of constrained hyperpaths as

$$\mathcal{X}' = \{x \in \mathcal{X} : Ax = b\}$$

[1] The purpose of the offset will be clear in later sections. For this section, the value of τ can be taken as 0.

where we have a constraint matrix $A \in \mathbb{R}^{|b| \times |\mathcal{E}|}$ and vector $b \in R^{|b|}$ encoding $|b|$ constraints. The optimal constrained hyperpath is $x^* = \arg\max_{x \in \mathcal{X}'} \theta^\top x + \tau$.

Note that the constrained hypergraph search problem may be NP-Hard. Crucially this is true even when the corresponding unconstrained search problem is solvable in polynomial time. For instance, phrase-based decoding is known to be NP-Hard (Knight, 1999), but we will see that it can be expressed as a polynomial-sized hypergraph with constraints.

Example: Phrase-Based Machine Translation Consider translating a source sentence $w_1 \dots w_{|w|}$ to a target sentence in a language with vocabulary Σ. A simple phrase-based translation model consists of a tuple $(\mathcal{P}, \omega, \sigma)$ with

- $\mathcal{P}$; a set of pairs (q, r) where $q_1 \dots q_{|q|}$ is a sequence of source-language words and $r_1 \dots r_{|r|}$ is a sequence of target-language words drawn from the target vocabulary Σ.
- $\omega : \mathbb{R}^{|\mathcal{P}|}$; parameters for the translation model mapping each pair in $\mathcal{P}$ to a real-valued score.
- $\sigma : \mathbb{R}^{|\Sigma \times \Sigma|}$; parameters of the language model mapping a bigram of target-language words to a real-valued score.

The translation decoding problem is to find the best derivation for a given source sentence. A derivation consists of a sequence of *phrases* $p = p_1 \dots p_n$. Define a phrase as a tuple (q, r, j, k) consisting of a span in the source sentence $q = w_j \dots w_k$ and a sequence of target words $r_1 \dots r_{|r|}$, with $(q, r) \in \mathcal{P}$. We say the source words $w_j \dots w_k$ are *translated* to r.

The score of a derivation, $f(p)$, is the sum of the translation score of each phrase plus the language model score of the target sentence

$$f(p) = \sum_{i=1}^{n} \omega(q(p_i), r(p_i)) + \sum_{i=0}^{|u|+1} \sigma(u_{i-1}, u_i)$$

where u is the sequence of words in Σ formed by concatenating the phrases $r(p_1) \dots r(p_n)$, with boundary cases $u_0 = \texttt{<s>}$ and $u_{|u|+1} = \texttt{</s>}$.

Crucially for a derivation to be valid it must satisfy an additional condition: it must translate every source word *exactly* once. The decoding problem for phrase-based translation is to find the highest-scoring derivation satisfying this property.

We can represent this decoding problem as a constrained hypergraph using the construction of Chang and Collins (2011). The hypergraph weights encode the translation and language model scores, and its structure ensures that the count of source words translated is $|w|$, i.e. the length of the source sentence. Each vertex will remember the preceding target-language word and the count of source words translated so far.

The hypergraph, which for this problem is also a directed graph, takes the following form.

- Vertices $v \in \mathcal{V}$ are labeled (c, u) where $c \in \{1 \dots |w|\}$ is the count of source words translated and $u \in \Sigma$ is the last target-language word produced by a partial hypothesis at this vertex. Additionally there is an initial terminal vertex labeled $(0, \texttt{<s>})$.
- There is a hyperedge $e \in \mathcal{E}$ with head (c', u') and tail $\langle (c, u) \rangle$ if there is a valid corresponding phrase (q, r, j, k) such that $c' = c + |q|$ and $u' = r_{|r|}$, i.e. c' is the count of words translated and u' is the last word of target phrase r. We call this phrase $p(e)$.

  The weight of this hyperedge, $\theta(e)$, is the translation model score of the pair plus its language model score

$$\theta(e) = \omega(q, r) + \left(\sum_{i=2}^{|r|} \sigma(r_{i-1}, r_i) \right) + \sigma(u, r_1)$$

- To handle the end boundary, there are hyperedges with head 1 and tail $\langle (|w|, u) \rangle$ for all $u \in \Sigma$. The weight of these edges is the cost of the stop bigram following u, i.e. $\sigma(u, \texttt{</s>})$.

While any valid derivation corresponds to a hyperpath in this graph, a hyperpath may not correspond to a valid derivation. For instance, a hyperpath may translate some source words more than once or not at all.

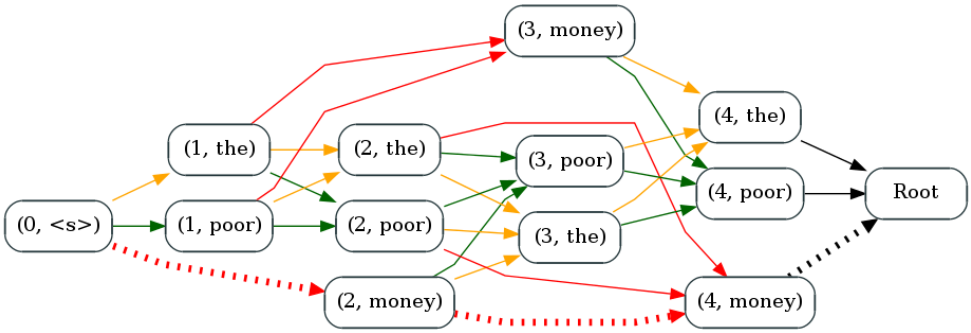


Figure 2: Hypergraph for translating the sentence $w = \texttt{les}_1\ \texttt{pauvres}_2\ \texttt{sont}_3\ \texttt{demunis}_4$ with set of pairs $\mathcal{P} = \{(\texttt{les}, \texttt{the}), (\texttt{pauvres}, \texttt{poor}), (\texttt{sont demunis}, \texttt{don't have any money})\}$. Hyperedges are color-coded by source words translated: orange for $\texttt{les}_1$, green for $\texttt{pauvres}_2$, and red for $\texttt{sont}_3\ \texttt{demunis}_4$. The dotted lines show an invalid hyperpath x that has signature $Ax = \langle 0, 0, 2, 2 \rangle \neq \langle 1, 1, 1, 1 \rangle$.

We handle this problem by adding additional constraints. For all source words $i \in \{1 \dots |w|\}$, define ρ as the set of hyperedges that translate w_i

$$\rho(i) = \{e \in \mathcal{E} : j(p(e)) \leq i \leq k(p(e))\}$$

Next define $|w|$ constraints enforcing that each word in the source sentence is translated exactly once

$$\sum_{e \in \rho(i)} x(e) = 1 \quad \forall\, i \in \{1 \dots |w|\}$$

These linear constraints can be represented with a matrix $A \in \{0, 1\}^{|w| \times |\mathcal{E}|}$ where the rows correspond to source indices and the columns correspond to edges. We call the product Ax the *signature*, where in this case $(Ax)_i$ is the number of times word i has been translated. The full set of constrained hyperpaths is $\mathcal{X}' = \{x \in \mathcal{X} : Ax = \mathbf{1}\}$, and the best derivation under this phrase-based translation model has score $\max_{x \in \mathcal{X}'} \theta^\top x + \tau$.

Figure 2.2 shows an example hypergraph with constraints for translating the sentence `les pauvres sont demunis` into English using a simple set of phrases. Even in this small example, many of the possible hyperpaths violate the constraints and correspond to invalid derivations.

Example: Syntax-Based Machine Translation Syntax-based machine translation with a language model can also be expressed as a constrained hypergraph problem. For the sake of space, we omit the definition. See Rush and Collins (2011) for an in-depth description of the constraint matrix used for syntax-based translation.

3 A Variant of Beam Search

This section describes a variant of the beam search algorithm for finding the highest-scoring constrained hyperpath. The algorithm uses three main techniques: (1) dynamic programming with additional signature information to satisfy the constraints, (2) beam pruning where some, possibly optimal, hypotheses are discarded, and (3) branch-and-bound-style application of upper and lower bounds to discard provably non-optimal hypotheses.

Any solution returned by the algorithm will be a valid constrained hyperpath and a member of $\mathcal{X}'$. Additionally the algorithm returns a certificate flag opt that, if true, indicates that no beam pruning was used, implying the solution returned is optimal. Generally it will be hard to produce a certificate even by reducing the amount of beam pruning; however in the next section we will introduce a method based on Lagrangian relaxation to tighten the upper bounds. These bounds will help eliminate most solutions before they trigger pruning.

3.1 Algorithm

Figure 3 shows the complete beam search algorithm. At its core it is a dynamic programming algorithm filling in the chart π. The beam search chart indexes hypotheses by vertex $v \in \mathcal{V}$ as well as a signature $sig \in \mathbb{R}^{|b|}$ where $|b|$ is the number of constraints. A new hypothesis is constructed from each hyperedge and all possible signatures of tail nodes. We define the function SIGS to take the tail of an edge and re-

turn the set of possible signature combinations

$$\textsc{Sigs}(v_2, \dots v_{|v|}) = \prod_{i=2}^{|v|} \{sig : \pi[v_i, sig] \neq -\infty\}$$

where the product is the Cartesian product over sets. Line 8 loops over this entire set.[2] For hypothesis x, the algorithm ensures that its signature sig is equal to Ax. This property is updated on line 9.

The signature provides proof that a hypothesis is still valid. Let the function $\textsc{Check}(sig)$ return true if the hypothesis can still fulfill the constraints. For example, in phrase-based decoding, we will define $\textsc{Check}(sig) = (sig \leq \mathbf{1})$; this ensures that each word has been translated 0 or 1 times. This check is applied on line 11.

Unfortunately maintaining all signatures is inefficient. For example we will see that in phrase-based decoding the signature is a bit-string recording which source words have been translated; the number of possible bit-strings is exponential in the length of the sentence. The algorithm includes two methods for removing hypotheses, bounding and pruning.

Bounding allows us to discard provably non-optimal solutions. The algorithm takes as arguments a lower bound on the optimal score $\mathrm{lb} \leq \theta^\top x^* + \tau$, and computes upper bounds on the outside score for all vertices v: $\mathrm{ubs}[v]$, i.e. an overestimate of the score for completing the hyperpath from v. If a hypothesis has score s, it can only be optimal if $s + \mathrm{ubs}[v] \geq \mathrm{lb}$. This bound check is performed on line 11.

Pruning removes weak partial solutions based on problem-specific checks. The algorithm invokes the black-box function, PRUNE, on line 13, passing it a pruning parameter β and a vertex-signature pair. The parameter β controls a threshold for pruning. For instance for phrase-based translation, it specifies a hard-limit on the number of hypotheses to retain. The function returns true if it prunes from the chart. Note that pruning may remove optimal hypotheses, so we set the certificate flag opt to false if the chart is modified.

[2]For simplicity we write this loop over the entire set. In practice it is important to use data structures to optimize lookup. See Tillmann (2006) and Huang and Chiang (2005).

1: **procedure** BEAMSEARCH($\theta, \tau, \mathrm{lb}, \beta$)
2: $\mathrm{ubs} \leftarrow \textsc{Outside}(\theta, \tau)$
3: $\mathrm{opt} \leftarrow \mathrm{true}$
4: $\pi[v, sig] \leftarrow -\infty$ for all $v \in \mathcal{V}, sig \in \mathbb{R}^{|b|}$
5: $\pi[v, 0] \leftarrow 0$ for all $v \in \mathcal{T}$
6: **for** $e \in \mathcal{E}$ in topological order **do**
7: $\langle \langle v_2, \dots, v_{|v|} \rangle, v_1 \rangle \leftarrow e$
8: **for** $sig^{(2)} \dots sig^{(|v|)} \in \textsc{Sigs}(v_2, \dots, v_{|v|})$ **do**
9: $sig \leftarrow A\delta(e) + \sum_{i=2}^{|v|} sig^{(i)}$
10: $s \leftarrow \theta(e) + \sum_{i=2}^{|v|} \pi[v_i, sig^{(i)}]$
11: **if** $\begin{pmatrix} s > \pi[v_1, sig] \wedge \\ \textsc{Check}(sig) \wedge \\ s + \mathrm{ubs}[v_1] \geq \mathrm{lb} \end{pmatrix}$ **then**
12: $\pi[v_1, sig] \leftarrow s$
13: **if** $\textsc{Prune}(\pi, v_1, sig, \beta)$ **then** $\mathrm{opt} \leftarrow \mathrm{false}$
14: $\mathrm{lb}' \leftarrow \pi[1, c] + \tau$
15: **return** $\mathrm{lb}', \mathrm{opt}$

Input:

$(\mathcal{V}, \mathcal{E}, \theta, \tau)$	hypergraph with weights
(A, b)	matrix and vector for constraints
$\mathrm{lb} \in \mathbb{R}$	lower bound
β	a pruning parameter

Output:

lb'	resulting lower bound score
opt	certificate of optimality

Figure 3: A variant of the beam search algorithm. Uses dynamic programming to produce a lower bound on the optimal constrained solution and, possibly, a certificate of optimality. Function OUTSIDE computes upper bounds on outside scores. Function SIGS enumerates all possible tail signatures. Function CHECK identifies signatures that do not violate constraints. Bounds lb and ubs are used to remove provably non-optimal solutions. Function PRUNE, taking parameter β, returns true if it prunes hypotheses from π that could be optimal.

This variant on beam search satisfies the following two properties (recall x^* is the optimal constrained solution)

Property 3.1 (Primal Feasibility)**.** *The returned score* lb' *lower bounds the optimal constrained score, that is* $\mathrm{lb}' \leq \theta^\top x^* + \tau$.

Property 3.2 (Dual Certificate)**.** *If beam search returns with* $\mathrm{opt} = \mathrm{true}$*, then the returned score is optimal, i.e.* $\mathrm{lb}' = \theta^\top x^* + \tau$.

An immediate consequence of Property 3.1 is that the output of beam search, lb', can be used as the input lb for future runs of the algorithm. Furthermore,

procedure PRUNE(π, v, sig, β)
  $\mathcal{C} \leftarrow \{(v', sig') : ||sig'||_1 = ||sig||_1, \pi[v', sig'] \neq -\infty\}$
  $\mathcal{D} \leftarrow \mathcal{C} \setminus \text{mBEST}(\beta, \mathcal{C}, \pi)$
  $\pi[v', sig'] \leftarrow -\infty$ for all $v', sig' \in \mathcal{D}$
  if $\mathcal{D} = \emptyset$ **then return** true
  else return false

Input: (v, sig) the last hypothesis added to the chart
$\beta \in \mathbb{Z}$ # of hypotheses to retain
Output: true, if π is modified

Figure 4: Pruning function for phrase-based translation. Set $\mathcal{C}$ contains all hypotheses with $||sig||_1$ source words translated. The function prunes all but the top-β scoring hypotheses in this set.

if we loosen the amount of beam pruning by adjusting the pruning parameter β we can produce tighter lower bounds and discard more hypotheses. We can then iteratively apply this idea with a sequence of parameters $\beta_1 \dots \beta_K$ producing lower bounds $\text{lb}^{(1)}$ through $\text{lb}^{(K)}$. We return to this idea in Section 5.

Example: Phrase-based Beam Search. Recall that the constraints for phrase-based translation consist of a binary matrix $A \in \{0,1\}^{|w| \times |\mathcal{E}|}$ and vector $b = \mathbf{1}$. The value sig_i is therefore the number of times source word i has been translated in the hypothesis. We define the predicate CHECK as CHECK$(sig) = (sig \leq \mathbf{1})$ in order to remove hypotheses that already translate a source word more than once, and are therefore invalid. For this reason, phrase-based signatures are called *bit-strings*.

A common beam pruning strategy is to group together items into a set $\mathcal{C}$ and retain a (possibly complete) subset. An example phrase-based beam pruner is given in Figure 4. It groups together hypotheses based on $||sig_i||_1$, i.e. the number of source words translated, and applies a hard pruning filter that retains only the β highest-scoring items $(v, sig) \in \mathcal{C}$ based on $\pi[v, sig]$.

3.2 Computing Upper Bounds

Define the set $\mathcal{O}(v, x)$ to contain all outside edges of vertex v in hyperpath x (informally, hyperedges that do not have v as an ancestor). For all $v \in \mathcal{V}$, we set the upper bounds, ubs, to be the best unconstrained outside score

$$\text{ubs}[v] = \max_{x \in \mathcal{X}: v \in x} \sum_{e \in \mathcal{O}(v,x)} \theta(e) + \tau$$

This upper bound can be efficiently computed for all vertices using the standard outside dynamic programming algorithm. We will refer to this algorithm as OUTSIDE(θ, τ).

Unfortunately, as we will see, these upper bounds are often quite loose. The issue is that unconstrained outside paths are able to violate the constraints without being penalized, and therefore greatly overestimate the score.

4 Finding Tighter Bounds with Lagrangian Relaxation

Beam search produces a certificate only if beam pruning is never used. In the case of phrase-based translation, the certificate is dependent on all groups $\mathcal{C}$ having β or less hypotheses. The only way to ensure this is to bound out enough hypotheses to avoid pruning. The effectiveness of the bounding inequality, $s + \text{ubs}[v] < \text{lb}$, in removing hypotheses is directly dependent on the tightness of the bounds.

In this section we propose using Lagrangian relaxation to improve these bounds. We first give a brief overview of the method and then apply it to computing bounds. Our experiments show that this approach is very effective at finding certificates.

4.1 Algorithm

In Lagrangian relaxation, instead of solving the constrained search problem, we relax the constraints and solve an unconstrained hypergraph problem with modified weights. Recall the constrained hypergraph problem: $\max_{x \in \mathcal{X}: Ax = b} \theta^\top x + \tau$. The Lagrangian dual of this optimization problem is

$$\begin{aligned} L(\lambda) &= \max_{x \in \mathcal{X}} \theta^\top x + \tau - \lambda^\top (Ax - b) \\ &= \left(\max_{x \in \mathcal{X}} (\theta - A^\top \lambda)^\top x \right) + \tau + \lambda^\top b \\ &= \max_{x \in \mathcal{X}} \theta'^\top x + \tau' \end{aligned}$$

where $\lambda \in R^{|b|}$ is a vector of dual variables and define $\theta' = \theta - A^\top \lambda$ and $\tau' = \tau + \lambda^\top b$. This maximization is over $\mathcal{X}$, so for any value of λ, $L(\lambda)$ can be calculated as `BestPathScore`(θ', τ').

Note that for all valid constrained hyperpaths $x \in \mathcal{X}'$ the term $Ax - b$ equals 0, which implies that these hyperpaths have the same score under the modified weights as under the original weights, $\theta^\top x + \tau = \theta'^\top x + \tau'$. This leads to the following two properties,

procedure LRROUND(α_k, λ)
$x \leftarrow \arg\max_{x \in \mathcal{X}} \theta^\top x + \tau - \lambda^\top (Ax - b)$
$\lambda' \leftarrow \lambda - \alpha_k (Ax - b)$
$\text{opt} \leftarrow Ax = b$
$\text{ub} \leftarrow \theta^\top x + \tau$
return $\lambda', \text{ub}, \text{opt}$
procedure LAGRANGIANRELAXATION(α)
$\lambda^{(0)} \leftarrow 0$
for k in $1 \dots K$ **do**
$\lambda^{(k)}, \text{ub}, \text{opt} \leftarrow \text{LRROUND}(\alpha_k, \lambda^{(k-1)})$
if opt **then return** $\lambda^{(k)}, \text{ub}, \text{opt}$
return $\lambda^{(K)}, \text{ub}, \text{opt}$

Input: $\alpha_1 \dots \alpha_K$ sequence of subgradient rates

Output:
- λ final dual vector
- ub upper bound on optimal constrained solution
- opt certificate of optimality

Figure 5: Lagrangian relaxation algorithm. The algorithm repeatedly calls LRROUND to compute the subgradient, update the dual vector, and check for a certificate.

where $x \in \mathcal{X}$ is the hyperpath computed within the max,

Property 4.1 (Dual Feasibility). *The value $L(\lambda)$ upper bounds the optimal solution, that is $L(\lambda) \geq \theta^\top x^* + \tau$*

Property 4.2 (Primal Certificate). *If the hyperpath x is a member of $\mathcal{X}'$, i.e. $Ax = b$, then $L(\lambda) = \theta^\top x^* + \tau$.*

Property 4.1 states that $L(\lambda)$ always produces some upper bound; however, to help beam search, we want as tight a bound as possible: $\min_\lambda L(\lambda)$.

The Lagrangian relaxation algorithm, shown in Figure 5, uses subgradient descent to find this minimum. The subgradient of $L(\lambda)$ is $Ax - b$ where x is the argmax of the modified objective $x = \arg\max_{x \in \mathcal{X}} \theta'^\top x + \tau'$. Subgradient descent iteratively solves unconstrained hypergraph search problems to compute these subgradients and updates λ. See Rush and Collins (2012) for an extensive discussion of this style of optimization in natural language processing.

Example: Phrase-based Relaxation. For phrase-based translation, we expand out the Lagrangian to

$$L(\lambda) = \max_{x \in \mathcal{X}} \theta^\top x + \tau - \lambda^\top (Ax - b) =$$
$$\max_{x \in \mathcal{X}} \sum_{e \in \mathcal{E}} \left(\theta(e) - \sum_{i=j(p(e))}^{k(p(e))} \lambda_i \right) x(e) + \tau + \sum_{i=1}^{|s|} \lambda_i$$

The weight of each edge $\theta(e)$ is modified by the dual variables λ_i for each source word translated by the edge, i.e. if $(q, r, j, k) = p(e)$, then the score is modified by $\sum_{i=j}^{k} \lambda_i$. A solution under these weights may use source words multiple times or not at all. However if the solution uses each source word exactly once ($Ax = \mathbf{1}$), then we have a certificate and the solution is optimal.

4.2 Utilizing Upper Bounds in Beam Search

For many problems, it may not be possible to satisfy Property 4.2 by running the subgradient algorithm alone. Yet even for these problems, applying subgradient descent will produce an improved estimate of the upper bound, $\min_\lambda L(\lambda)$.

To utilize these improved bounds, we simply replace the weights in beam search and the outside algorithm with the modified weights from Lagrangian relaxation, θ' and τ'. Since the result of beam search must be a valid constrained hyperpath $x \in \mathcal{X}'$, and for all $x \in \mathcal{X}'$, $\theta^\top x + \tau = \theta'^\top x + \tau'$, this substitution does not alter the necessary properties of the algorithm; i.e. if the algorithm returns with opt equal to true, then the solution is optimal.

Additionally the computation of upper bounds now becomes

$$\text{ubs}[v] = \max_{x \in \mathcal{X}: v \in x} \sum_{e \in \mathcal{O}(v,x)} \theta'(e) + \tau'$$

These outside paths may still violate constraints, but the modified weights now include penalty terms to discourage common violations.

5 Optimal Beam Search

The optimality of the beam search algorithm is dependent on the tightness of the upper and lower bounds. We can produce better lower bounds by varying the pruning parameter β; we can produce better upper bounds by running Lagrangian relaxation. In this section we combine these two ideas and present a complete optimal beam search algorithm.

Our general strategy will be to use Lagrangian relaxation to compute modified weights and to use beam search over these modified weights to attempt to find an optimal solution. One simple method for doing this, shown at the top of Figure 6, is to run

in stages. The algorithm first runs Lagrangian relaxation to compute the best λ vector. The algorithm then iteratively runs beam search using the parameter sequence β_k. These parameters allow the algorithm to loosen the amount of beam pruning. For example in phrase based pruning, we would raise the number of hypotheses stored per group until no beam pruning occurs.

A clear disadvantage of the staged approach is that it needs to wait until Lagrangian relaxation is completed before even running beam search. Often beam search will be able to quickly find an optimal solution even with good but non-optimal λ. In other cases, beam search may still improve the lower bound lb.

This motivates the alternating algorithm OPTBEAM shown Figure 6. In each round, the algorithm alternates between computing subgradients to tighten ubs and running beam search to maximize lb. In early rounds we set β for aggressive beam pruning, and as the upper bounds get tighter, we loosen pruning to try to get a certificate. If at any point either a primal or dual certificate is found, the algorithm returns the optimal solution.

procedure OPTBEAMSTAGED(α, β)
  $\lambda, \mathrm{ub}, \mathrm{opt} \leftarrow$LAGRANGIANRELAXATION($\alpha$)
  if opt **then return** ub
  $\theta' \leftarrow \theta - A^\top \lambda$
  $\tau' \leftarrow \tau + \lambda^\top b$
  $\mathrm{lb}^{(0)} \leftarrow -\infty$
  for k in $1 \dots K$ **do**
    $\mathrm{lb}^{(k)}, \mathrm{opt} \leftarrow$ BEAMSEARCH($\theta', \tau', \mathrm{lb}^{(k-1)}, \beta_k$)
    if opt **then return** $\mathrm{lb}^{(k)}$
  return $\max_{k \in \{1 \dots K\}} \mathrm{lb}^{(k)}$

procedure OPTBEAM(α, β)
  $\lambda^{(0)} \leftarrow 0$
  $\mathrm{lb}^{(0)} \leftarrow -\infty$
  for k in $1 \dots K$ **do**
    $\lambda^{(k)}, \mathrm{ub}^{(k)}, \mathrm{opt} \leftarrow$ LRROUND($\alpha_k, \lambda^{(k-1)}$)
    if opt **then return** $\mathrm{ub}^{(k)}$
    $\theta' \leftarrow \theta - A^\top \lambda^{(k)}$
    $\tau' \leftarrow \tau + \lambda^{(k)\top} b$
    $\mathrm{lb}^{(k)}, \mathrm{opt} \leftarrow$ BEAMSEARCH($\theta', \tau', \mathrm{lb}^{(k-1)}, \beta_k$)
    if opt **then return** $\mathrm{lb}^{(k)}$
  return $\max_{k \in \{1 \dots K\}} \mathrm{lb}^{(k)}$

Input: $\begin{bmatrix} \alpha_1 \dots \alpha_K & \text{sequence of subgradient rates} \\ \beta_1 \dots \beta_K & \text{sequence of pruning parameters} \end{bmatrix}$
Output: optimal constrained score or lower bound

Figure 6: Two versions of optimal beam search: staged and alternating. Staged runs Lagrangian relaxation to find the optimal λ, uses λ to compute upper bounds, and then repeatedly runs beam search with pruning sequence $\beta_1 \dots \beta_k$. Alternating switches between running a round of Lagrangian relaxation and a round of beam search with the updated λ. If either produces a certificate it returns the result.

6 Related Work

Approximate methods based on beam search and cube-pruning have been widely studied for phrase-based (Koehn et al., 2003; Tillmann and Ney, 2003; Tillmann, 2006) and syntax-based translation models (Chiang, 2007; Huang and Chiang, 2007; Watanabe et al., 2006; Huang and Mi, 2010).

There is a line of work proposing exact algorithms for machine translation decoding. Exact decoders are often slow in practice, but help quantify the errors made by other methods. Exact algorithms proposed for IBM model 4 include ILP (Germann et al., 2001), cutting plane (Riedel and Clarke, 2009), and multi-pass A* search (Och et al., 2001). Zaslavskiy et al. (2009) formulate phrase-based decoding as a traveling salesman problem (TSP) and use a TSP decoder. Exact decoding algorithms based on finite state transducers (FST) (Iglesias et al., 2009) have been studied on phrase-based models with limited reordering (Kumar and Byrne, 2005). Exact decoding based on FST is also feasible for certain hierarchical grammars (de Gispert et al., 2010). Chang and Collins (2011) and Rush and Collins (2011) develop Lagrangian relaxation-based approaches for exact machine translation.

Apart from translation decoding, this paper is closely related to work on column generation for NLP. Riedel et al. (2012) and Belanger et al. (2012) relate column generation to beam search and produce exact solutions for parsing and tagging problems. The latter work also gives conditions for when beam search-style decoding is optimal.

7 Results

To evaluate the effectiveness of optimal beam search for translation decoding, we implemented decoders for phrase- and syntax-based models. In this section we compare the speed and optimality of these

decoders to several baseline methods.

7.1 Setup and Implementation

For phrase-based translation we used a German-to-English data set taken from Europarl (Koehn, 2005). We tested on 1,824 sentences of length at most 50 words. For experiments the phrase-based systems uses a trigram language model and includes standard distortion penalties. Additionally the unconstrained hypergraph includes further derivation information similar to the graph described in Chang and Collins (2011).

For syntax-based translation we used a Chinese-to-English data set. The model and hypergraphs come from the work of Huang and Mi (2010). We tested on 691 sentences from the newswire portion of the 2008 NIST MT evaluation test set. For experiments, the syntax-based model uses a trigram language model. The translation model is tree-to-string syntax-based model with a standard context-free translation forest. The constraint matrix A is based on the constraints described by Rush and Collins (2011).

Our decoders use a two-pass architecture. The first pass sets up the hypergraph in memory, and the second pass runs search. When possible the baselines share optimized construction and search code.

The performance of optimal beam search is dependent on the sequences α and β. For the step-size α we used a variant of Polyak's rule (Polyak, 1987; Boyd and Mutapcic, 2007), substituting the unknown optimal score for the last computed lower bound: $\alpha_k \leftarrow \frac{\text{ub}^{(k)} - \text{lb}^{(k)}}{||Ax^{(k)} - b||_2^2}$. We adjust the order of the pruning parameter β based on a function μ of the current gap: $\beta_k \leftarrow 10^{\mu(\text{ub}^{(k)} - \text{lb}^{(k)})}$.

Previous work on these data sets has shown that exact algorithms do not result in a significant increase in translation accuracy. We focus on the efficiency and model score of the algorithms.

7.2 Baseline Methods

The experiments compare optimal beam search (OPTBEAM) to several different decoding methods. For both systems we compare to: BEAM, the beam search decoder from Figure 3 using the original weights θ and τ, and $\beta \in \{100, 1000\}$; LR-TIGHT, Lagrangian relaxation followed by incremental tightening constraints, which is a reimplementation of Chang and Collins (2011) and Rush and Collins (2011).

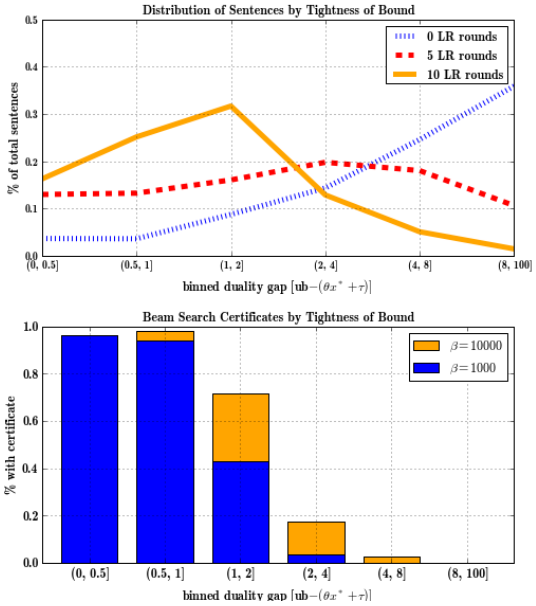


Figure 7: Two graphs from phrase-based decoding. Graph (a) shows the duality gap distribution for 1,824 sentences after 0, 5, and 10 rounds of LR. Graph (b) shows the % of certificates found for sentences with differing gap sizes and beam search parameters β. Duality gap is defined as, ub - $(\theta^\top x^* + \tau)$.

For phrase-based translation we compare with: MOSES-GC, the standard Moses beam search decoder with $\beta \in \{100, 1000\}$ (Koehn et al., 2007); MOSES, a version of Moses without gap constraints more similar to BEAM (see Chang and Collins (2011)); ASTAR, an implementation of A* search using original outside scores, i.e. OUTSIDE(θ, τ), and capped at 20,000,000 queue pops.

For syntax-based translation we compare with: ILP, a general-purpose integer linear programming solver (Gurobi Optimization, 2013) and CUBEPRUNING, an approximate decoding method similar to beam search (Chiang, 2007), tested with $\beta \in \{100, 1000\}$.

7.3 Experiments

Table 1 shows the main results. For phrase-based translation, OPTBEAM decodes the optimal translation with certificate in 99% of sentences with an average time of 17.27 seconds per sentence. This

	11-20 (558)			21-30 (566)			31-40 (347)			41-50 (168)			all (1824)		
Phrase-Based	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact
BEAM (100)	2.33	19.5	38.0	8.37	1.6	7.2	24.12	0.3	1.4	71.35	0.0	0.0	14.50	15.3	23.2
BEAM (1000)	2.33	37.8	66.3	8.42	3.4	18.9	21.60	0.6	3.2	53.99	0.6	1.2	12.44	22.6	36.9
BEAM (100000)	3.34	83.9	96.2	18.53	22.4	60.4	46.65	2.0	18.1	83.53	1.2	6.5	23.39	43.2	62.4
MOSES (100)	0.18	0.0	81.0	0.36	0.0	45.6	0.53	0.0	14.1	0.74	0.0	6.0	0.34	0.0	52.3
MOSES (1000)	2.29	0.0	97.8	4.39	0.0	78.8	6.52	0.0	43.5	9.00	0.0	19.6	4.20	0.0	74.6
ASTAR (cap)	11.11	99.3	99.3	91.39	53.9	53.9	122.67	7.8	7.8	139.61	1.2	1.2	67.99	58.8	58.8
LR-TIGHT	4.20	100.0	100.0	23.25	100.0	100.0	88.16	99.7	99.7	377.9	97.0	97.0	60.11	99.7	99.7
OPTBEAM	2.85	100.0	100.0	10.33	100.0	100.0	28.29	100.0	100.0	84.34	97.0	97.0	17.27	99.7	99.7
ChangCollins	10.90	100.0	100.0	57.20	100.0	100.0	203.4	99.7	99.7	679.9	97.0	97.0	120.9	99.7	99.7
MOSES-GC (100)	0.14	0.0	89.4	0.27	0.0	84.1	0.41	0.0	75.8	0.58	0.0	78.6	0.26	0.0	84.9
MOSES-GC (1000)	1.33	0.0	89.4	2.62	0.0	84.3	4.15	0.0	75.8	6.19	0.0	79.2	2.61	0.0	85.0

	11-20 (192)			21-30 (159)			31-40 (136)			41-100 (123)			all (691)		
Syntax-Based	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact
BEAM (100)	0.40	4.7	75.9	0.40	0.0	66.0	0.75	0.0	43.4	1.66	0.0	25.8	0.68	5.72	58.7
BEAM (1000)	0.78	16.9	79.4	2.65	0.6	67.1	6.20	0.0	47.5	15.5	0.0	36.4	4.16	12.5	65.5
CUBE (100)	0.08	0.0	77.6	0.16	0.0	66.7	0.23	0.0	43.9	0.41	0.0	26.3	0.19	0.0	59.0
CUBE (1000)	1.76	0.0	91.7	4.06	0.0	95.0	5.71	0.0	82.9	10.69	0.0	60.9	4.66	0.0	85.0
LR-TIGHT	0.37	100.0	100.0	1.76	100.0	100.0	4.79	100.0	100.0	30.85	94.5	94.5	7.25	99.0	99.0
OPTBEAM	0.23	100.0	100.0	0.50	100.0	100.0	1.42	100.0	100.0	7.14	93.6	93.6	1.75	98.8	98.8
ILP	9.15	100.0	100.0	32.35	100.0	100.0	49.6	100.0	100.0	108.6	100.0	100.0	40.1	100.0	100.0

Table 1: Experimental results for translation experiments. Column *time* is the mean time per sentence in seconds, *cert* is the percentage of sentences solved with a certificate of optimality, *exact* is the percentage of sentences solved exactly, i.e. $\theta^\top x + \tau = \theta^\top x^* + \tau$. Results are grouped by sentence length (group 1-10 is omitted for space).

is seven times faster than the decoder of Chang and Collins (2011) and 3.5 times faster then our reimplementation, LR-TIGHT. ASTAR performs poorly, taking lots of time on difficult sentences. BEAM runs quickly, but rarely finds an exact solution. MOSES without gap constraints is also fast, but less exact than OPTBEAM and unable to produce certificates.

For syntax-based translation. OPTBEAM finds a certificate on 98.8% of solutions with an average time of 1.75 seconds per sentence, and is four times faster than LR-TIGHT. CUBE (100) is an order of magnitude faster, but is rarely exact on longer sentences. CUBE (1000) finds more exact solutions, but is comparable in speed to optimal beam search. BEAM performs better than in the phrase-based model, but is not much faster than OPTBEAM.

Figure 7.2 shows the relationship between beam search optimality and duality gap. Graph (a) shows how a handful of LR rounds can significantly tighten the upper bound score of many sentences. Graph (b) shows how beam search is more likely to find optimal solutions with tighter bounds. BEAM effectively uses 0 rounds of LR, which may explain why it finds so few optimal solutions compared to OPTBEAM.

Table 2 breaks down the time spent in each part of the algorithm. For both methods, beam search has the most time variance and uses more time on longer sentences. For phrase-based sentences, Lagrangian relaxation is fast, and hypergraph construction dominates. If not for this cost, OPTBEAM might be comparable in speed to MOSES (1000).

		≥ 30		all	
		mean	median	mean	median
PB	Hypergraph	56.6%	69.8%	59.6%	69.6%
	Lag. Relaxation	10.0%	5.5%	9.4%	7.6%
	Beam Search	33.4%	24.6%	30.9%	22.8%
SB	Hypergraph	0.5%	1.6%	0.8%	2.4%
	Lag. Relaxation	15.0%	35.2%	17.3%	41.4%
	Beam Search	84.4%	63.1%	81.9 %	56.1%

Table 2: Distribution of time within optimal beam search, including: hypergraph construction, Lagrangian relaxation, and beam search. *Mean* is the percentage of total time. *Median* is the distribution over the median values for each row.

8 Conclusion

In this work we develop an optimal variant of beam search and apply it to machine translation decoding. The algorithm uses beam search to produce constrained solutions and bounds from Lagrangian relaxation to eliminate non-optimal solutions. Results show that this method can efficiently find exact solutions for two important styles of machine translation.

Acknowledgments Alexander Rush, Yin-Wen Chang and Michael Collins were all supported by NSF grant IIS-1161814. Alexander Rush was partially supported by an NSF Graduate Research Fellowship.

References

David Belanger, Alexandre Passos, Sebastian Riedel, and Andrew McCallum. 2012. Map inference in chains using column generation. In *NIPS*, pages 1853–1861.

Stephen Boyd and Almir Mutapcic. 2007. Subgradient methods.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 26–37. Association for Computational Linguistics.

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

Adria de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical Phrase-Based Translation with Weighted Finite-State Transducers and Shallow-n Grammars. In *Computational linguistics*, volume 36, pages 505–533.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathen Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vlad Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 228–235.

Inc. Gurobi Optimization. 2013. Gurobi optimizer reference manual.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 380–388, Athens, Greece, March. Association for Computational Linguistics.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL '03, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. *Machine translation: From real users to research*, pages 115–124.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

R. Kipp Martin, Rardin L. Rardin, and Brian A. Campbell. 1990. Polyhedral characterization of discrete dynamic programming. *Operations research*, 38(1):127–138.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation - Volume 14*, DMMT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Boris Polyak. 1987. *Introduction to Optimization*. Optimization Software, Inc.

Sebastian Riedel and James Clarke. 2009. Revisiting optimal decoding for machine translation IBM model 4. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 5–8. Association for Computational Linguistics.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut: delayed column and row generation for graph based parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational*

Natural Language Learning, pages 732–743. Association for Computational Linguistics.

Alexander M Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 72–82.

Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Christoph Tillmann. 2006. Efficient dynamic programming search algorithms for phrase-based SMT. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, CHSLP '06, pages 9–16.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 777–784, Morristown, NJ, USA. Association for Computational Linguistics.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 333–341, Stroudsburg, PA, USA. Association for Computational Linguistics.

An Efficient Language Model Using Double-Array Structures

Makoto Yasuhara **Toru Tanaka** †**Jun-ya Norimatsu** **Mikio Yamamoto**
Department of Computer Science
University of Tsukuba, Japan
†norimatsu@mibel.cs.tsukuba.ac.jp

Abstract

Ngram language models tend to increase in size with inflating the corpus size, and consume considerable resources. In this paper, we propose an efficient method for implementing $n$gram models based on double-array structures. First, we propose a method for representing backwards suffix trees using double-array structures and demonstrate its efficiency. Next, we propose two optimization methods for improving the efficiency of data representation in the double-array structures. Embedding probabilities into unused spaces in double-array structures reduces the model size. Moreover, tuning the word IDs in the language model makes the model smaller and faster. We also show that our method can be used for building large language models using the division method. Lastly, we show that our method outperforms methods based on recent related works from the viewpoints of model size and query speed when both optimization methods are used.

1 Introduction

Ngram language models (F. Jelinek, 1990) are widely used as probabilistic models of sentence in natural language processing. The wide use of the Internet has entailed a dramatic increase in size of the available corpora, which can be harnessed to obtain a significant improvement in model quality. In particular, Brants et al. (2007) have shown that the performance of statistical machine translation systems is monotonically improved with the increasing size of training corpora for the language model. However, models using larger corpora also consume more resources. In recent years, many methods for improving the efficiency of language models have been proposed to tackle this problem (Pauls and Klein, 2011; Kenneth Heafield, 2011). Such methods not only reduce the required memory size but also raise query speed.

In this paper, we propose the double-array language model (DALM) which uses double-array structures (Aoe, 1989). Double-array structures are widely used in text processing, especially for Japanese. They are known to provide a compact representation of tries (Fredkin, 1960) and fast transitions between trie nodes. The ability to store and manipulate tries efficiently is expected to increase the performance of language models (i.e., improving query speed and reducing the model size in terms of memory) because tries are one of the most common representations of data structures in language models. We use double-array structures to implement a language model since we can utilize their speed and compactness when querying the model about an $n$gram.

In order to utilize of double-array structures as language models, we modify them to be able to store probabilities and backoff weights. We also propose two optimization methods: `embedding` and `ordering`. These methods reduce model size and increase query speed. `Embedding` is an efficient method for storing $n$gram probabilities and backoff weights, whereby we find vacant spaces in the double-array language model structure and populate them with language model information, such as probabilities and backoff weights. `Ordering` is

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 222–232,
Seattle, Washington, USA, 18-21 October 2013.

a method for compacting the double-array structure. DALM uses word IDs for all words of the $n$gram, and `ordering` assigns a word ID to each word to reduce the model size. These two optimization methods can be used simultaneously and are also expected to work well.

In our experiments, we use a language model based on corpora of the NTCIR patent retrieval task (Atsushi Fujii et al., 2007; Atsushi Fujii et al., 2005; Atsushi Fujii et al., 2004; Makoto Iwayama et al., 2003). The model size is 31 GB in the ARPA file format. We conducted experiments focusing on query speed and model size. The results indicate that when the abovementioned optimization methods are used together, DALM outperforms state-of-the-art methods on those points.

2 Related Work

2.1 Tries and Backwards Suffix Trees

Tries (Fredkin, 1960) are one of the most widely used tree structures in $n$gram language models since they can reduce memory requirements by sharing common prefix. Moreover, since the query speed for tries depends only on the number of input words, the query speed remains constant even if the $n$gram model increases in size.

Backwards suffix trees (Bell et al., 1990; Stolcke, 2002; Germann et al., 2009) are among the most efficient representations of tries for language models. They contain $n$grams in reverse order of history words.

Figure 1 shows an example of a backwards suffix tree representation. In this paper, we denote an $n$gram: by the form $w_1, w_2, \cdots, w_n$ as $\boldsymbol{w}_1^n$. In this example, word lists (represented as rectangular tables) contain target words (here, w_n) of $n$grams, and circled words in the tree denote history words (here, $\boldsymbol{w}_1^{n-1}$) associated with target words. The history words "I eat," "you eat", and "do you eat" are stored in reverse order. Querying this trie about an $n$gram is simple: just trace history words in reverse and then find the target word in a list. For example, consider querying about the trigram "I eat fish". First, simply trace the history in the trie in reverse order ("eat" → "I"); then, find "fish" in list <1>. Similarly, querying a backwards suffix tree about unknown $n$grams is also efficient, because the backwards suffix tree

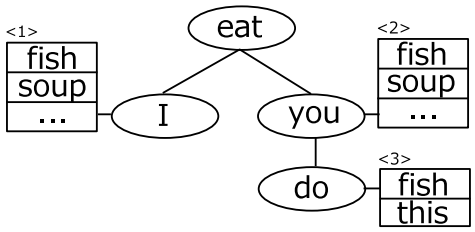


Figure 1: Example of a backwards suffix tree. There are two branch types in a backwards suffix tree: history words and target words. History words are shown in circles and target words are stored in word lists.

representation is highly suitable for the backoff calculation. For example, in querying about the 4gram "do you eat soup", we first trace "eat" → "you" → "do" in a manner similar to above. However, searching for the word "soup" in list <3> fails because list <3> does not contain the word "soup". In this case, we return to the node "you" to search the list <2>, where we find "soup". This means that the trigram "you eat soup" is contained in the tree while the 4gram "do you eat soup" is not. This behavior can be efficiently used for backoff calculation.

SRILM (Stolcke, 2002) is a widely used language model toolkit. It utilizes backwards suffix trees for its data structures. In SRILM, tries are implemented as 64-bit pointer links, which wastes a lot of memory. On the other hand, the access speed for $n$gram probabilities is relatively high.

2.2 Efficient Language Models

In recent years, several methods have been proposed for storing language models efficiently in memory.

Talbot and Osborne (2007) have proposed an efficient method based on bloom filters. This method modifies bloom filters to store count information about training sets. In prior work, bloom filters have been used for checking whether certain data are contained in a set. To store the count information, pairs from <$n$gram,1> to <$n$gram,count> are all added to the set for each $n$gram. To query this language model about the probability of an $n$gram, probabilities are calculated during querying by using these counts. Talbot and Brants (2008) have proposed a method based on perfect hash functions and bloomier filters. This method uses perfect hash functions to store $n$grams and encode values (for exam-

ple, probabilities or counts of $n$grams in the training corpus) to a large array. Guthrie and Hepple (2010) have proposed a language model called ShefLM that uses minimal perfect hash functions (Belazzougui et al., 2009), which can store $n$grams without vacant spaces. Furthermore, values are compressed by *simple dense coding* (Fredriksson and Nikitin, 2007). ShefLM achieves a high compression ratio when it stores counts of $n$grams in the training corpus. However, when this method stores probabilities of $n$grams, the advantage of using compression is limited because floating-point numbers are difficult to compress. Generally, compression is performed by combining the same values but, two floating-point numbers are rarely the same, especially in the case of probability values[1]. These methods implement lossy language models, meaning that, we can reduce the model size at the expense of model quality. These methods also reduce the model performance (perplexity).

Pauls and Klein (2011) have proposed BerkeleyLM which is based on an *implicit encoding* structure, where $n$grams are separated according to their order, and are sorted by word ID. The sorted $n$grams are linked to each other like a trie structure. BerkeleyLM provides rather efficient methods. Variable-length coding and block compression are used if small model size is more important than query speed. In addition, Heafield (2011) has proposed an efficient language model toolkit called KenLM that has been recently used in machine translation systems, for which large language models are often needed. KenLM has two different main structure types: `trie` and `probing`. The `trie` structure is compact but relatively slower to query, whereas the `probing` structure is relatively larger but faster to query.

In this paper, we propose a language model structure based on double-array structures. As we describe in Section 3, double-array structures can be used as fast and compact representations of tries. We propose several techniques for maximizing the performance of double-array structures from the perspective of query speed and model size.

[1]In our experience, it is considerably easier to compress backoff weights than to compress probabilities, although both are represented with floating-point numbers. We use this knowledge in our methods.

3 Double-Array

3.1 Double-Array Structure

In DALM, we use a double-array structure (Aoe, 1989) to represent the trie of a language model. Double-array structures are trie representations consisting of two parallel arrays: $BASE$ and $CHECK$. They are not only fast to query, but also provide a compact way to store tries. In the structure, nodes in the trie are represented by slots with the same index in both arrays. Before proposing several efficient language model representation techniques in Section 4, we introduce double-array themselves. In addition, the construction algorithms for double-arrays are described in Section 3.2 and Section 3.3.

The most naive implementation of a trie will have a two-dimensional array $NEXT$. Let WORDID (w) be a function that returns a word ID as a number corresponding to its argument word w; then $NEXT[n]$[WORDID (w)] (that presents the $WORDID(w)$-th slot of the nth row in the $NEXT$ array) stores the node number which can be transit from the node number n by the word w, and we can traverse the trie efficiently and easily to serialize the array in memory. This idea is simple but wastes the most of the used memory because almost all of the slots are unused and this results in occupying memory space. The double-array structures solve this problem by taking advantage of the sparseness of the $NEXT$ array. The two-dimensional array $NEXT$ is merged into a one-dimensional array $BASE$ by shifting the entries of each row of the $NEXT$ array and combining the set of resulting arrays. We can store this result in much less memory than the serialization of the naive implementation above. Additionally, a $CHECK$ array is introduced to check whether the transition is valid or not because we cannot distinguish which node the information in a particular slot comes from. Using a $CHECK$ array, we can avoid transition errors and move safely to the child node of any chosen node.

As a definition, a node link from a node n_s with a word w to the next node n_{next}in the trie is defined as follows:

$$\begin{aligned} next \quad &\leftarrow \quad BASE[s] + \text{WORDID}\,(w) \\ &if \quad CHECK[next] == s \end{aligned}$$

where s denotes the index of the slot in the double-

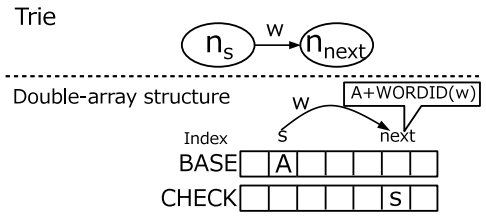


Figure 2: A trie and a corresponding double-array structure. Node n_s is represented by the slots $BASE[s]$ and $CHECK[s]$. A link from a node n_s with a word w is indicated by $CHECK[next] == s$.

array structure which represents n_s. The trie transition from a node n_s with a word w is applied according to the following steps:

Step 1 Calculating the "$next$" destination and

Step 2 Checking whether the transition is correct.

Step 2 specifically means the following:

1. If $CHECK[next] == s$, then we can "move" to the node n_{next};

2. otherwise, we can detect that the transition from the node n_s with the word w is not contained in the trie.

Figure 2 shows an example of a transition from a parent node n_s with a word w.

Next, we describe how the existence of an $n$gram history can be determined (Aoe, 1989). We can iterate over the nodes by the transitions shown above and may find the node representing an $n$gram history. But we should check that it is valid because nodes except for leaf nodes possiblly represent a fragment of some total $n$gram history. We can use *endmarker* symbols to determine whether an $n$gram history is in the trie. We add nodes meaning the *endmarker* symbol after the last node of each $n$gram history. When querying about $\boldsymbol{w}_1^{n-1}$, we transit repeatedly; in other words, we set $s = 0$ and start by applying Step 1 and 2 repeatedly for each word. When we reach the node w_{n-1}, we continue searching for an *endmarker* symbol. If the symbol is found, we know that the $n$gram history $\boldsymbol{w}_1^{n-1}$ is in the trie.

The double-array structure consumes 8 bytes per node because the $BASE$ and $CHECK$ arrays are 4 byte array variables. Therefore, the structure can store nodes compactly in case of a high filling rate. Moreover, node transitions are very fast because they require only one addition and one comparison per transition. We use a double-array structure in DALM, which can maximize its potential.

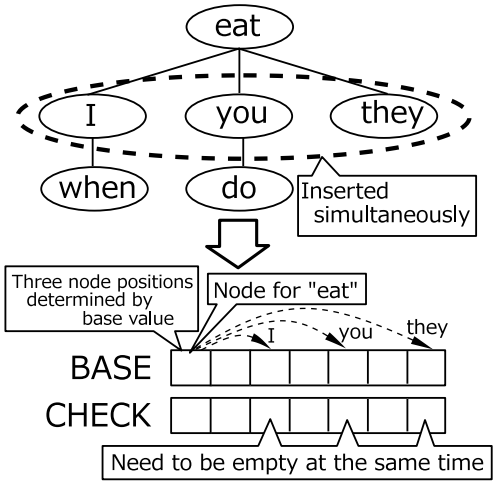


Figure 3: Greedy insertion of trie elements. The children of a node are collectively inserted into the double-array when the $BASE$ value of the node is fixed.

3.2 Greedy Construction

Greedy algorithms are widely used for constructing static double-array structures[2]. The construction steps are as follows:

1. Define the root node of a trie to correspond to index 0 of the double-array structure and

2. Find the $BASE$ value greedily (i.e., in order $1, 2, 3, \cdots$) for all nodes which have fixed their indices in the double-array structure.

In practice, once the $BASE$ value of a node is fixed, the positions of its children are fixed at the same time, and we can find the $BASE$ values for each child recursively.

Figure 3 shows an example of such construction. In this example, three nodes ("I", "you" and "they") are inserted at the same time. This is because the above three node positions are fixed by the $BASE$ value of the node "eat". To insert nodes

[2]We were unable to find an original source for this technique. However, this method is commonly used in double-array implementations.

"I", "you" and "they", the following three slots must be empty (i.e., the slots must not be used by other nodes.):

- $BASE[s]$ + WORDID("I")
- $BASE[s]$ + WORDID("you")
- $BASE[s]$ + WORDID("they")

where s is the index of the node "eat". At the construction step, we need to find $BASE[s]$ which satisfies the above conditions.

3.3 Efficient Construction Algorithm

The construction time for a double-array structure poses the greatest challenge. We use a more efficient method (Nakamura and Mochizuki, 2006) instead of the naive method for constructing a double-array structure because the naive method requires a long time. We call the method "empty doubly-linked list". This algorithm is one of the most efficient construction methods devised to date. Figure 4 shows an example of an empty doubly-linked list. We can efficiently define the $BASE$ value of each node by using the $CHECK$ array to store the next empty slot. In this example, in searching the $BASE$ value of a node, the first child node can be set to position 1, and if that fails, we can successively try positions $3, 4, 6, 8, \cdots$ by tracing the list instead of searching all $BASE$ values $0, 1, 2, 3, 4, 5, \cdots$.

As analyzed by Nakamura and Mochizuki(2006), the computational cost of a node insertion is less than in the naive method. The original naive method requires $O(NM)$ time for a node insertion, where M is a number of unique word types and N is a number of nodes of the trie; the algorithm using an empty double-linked list requires $O(UM)$, where U is the number of unused slots.

As described in Section 5, we divide the trie into several smaller tries and apply the efficient method for constructing our largest models. This is because it is not feasible to wait several weeks for the large language model structure to be built. The dividing method is currently the only method allowing us to build them.

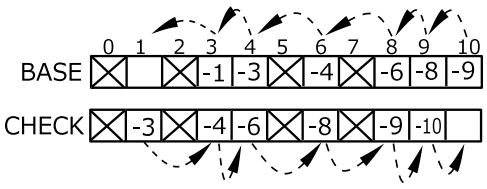


Figure 4: Empty doubly-linked list. Unused $CHECK$ slots are used to indicate the next unused slots, and unused $BASE$ slots are used to indicate previous unused slots. Thus, the $BASE$ and $CHECK$ arrays are used as a doubly-linked list which can reduce the number of ineffective trials.

4 Proposed Methods

4.1 DALM

In this section, we present the application of the double-array structure to backwards suffix trees. As this is the most basic structure based on double-array structures, we refer to it as the `simple` structure and improve its performance as described in the following sections.

To represent a backwards suffix tree as a double-array structure, we should modify the tree because it has two types of branches (target words and history nodes), which must be distinguished in the double-array structure. Instead, we should distinguish the branch type which indicates whether the node is a target word or a history word. We use the *endmarker* symbol (<#>) for branch discrimination. In prior work, the *endmarker* symbol has been used to indicate whether an $n$gram is in the trie. However, there is no need to distinguish whether the node of the tree is included in the language model because all nodes of a backwards suffix tree which represents $n$grams surely exist in the model. We use the *endmarker* symbol to indicate nodes which are end-of-history words. Therefore, target words of $n$grams are children of the *endmarker* symbols that they follow.

By using the *endmarker* symbol, target words can be treated the same as ordinary nodes because all target words are positioned after <#>. Figure 5 shows an example of such construction. We can clearly distinguish target words and history words in the backwards suffix tree.

Querying in the tree is rather simple. For example, consider the case of a query trigram "I eat fish" in the trie of Figure 5. We can trace this trigram in

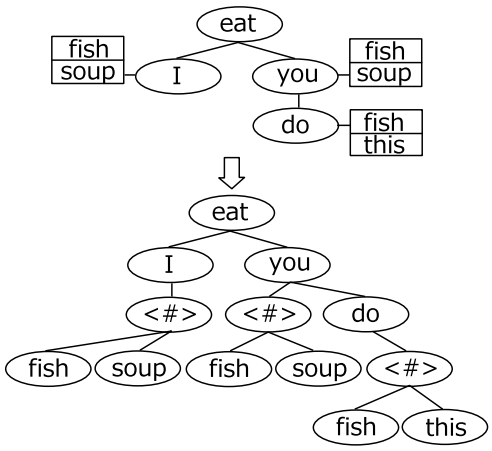


Figure 5: An example of converting a backwards suffix tree. We introduce *endmarker* symbols to distinguish the two branch types. We can treat the tree as an ordinary trie that can be represented by a double-array structure while retaining the advantages of the tree structure.

the same way as the original backwards suffix tree. First, we trace "eat" → "I", then trace that to the *endmarker* symbol <#> and finally find the word "fish".

Next, we describe the procedure for storing probabilities and backoff weights. We prepare a $VALUE$ array to store the probabilities and backoff weights of $n$grams. Figure 6 shows the `simple` DALM structure. The backwards suffix tree stores a backoff weight for each node and a probability for each target word. In `simple` DALM, each value is stored for the respective position of the corresponding node.

4.2 `Embedding`

`Embedding` is a method for reducing model size. In the `simple` DALM structure, there are many vacant spaces in the $BASE$ and $CHECK$ arrays. We use these vacant spaces to store backoff weights and probabilities. Figure 7 shows vacant spaces in the `simple` DALM structure.

First, the $BASE$ array slots of target word nodes are unused because target words are always in leaf positions in the backwards suffix tree and do not have any children nodes. In the example of Figure 7, $BASE[9]$ is not used, and therefore can be used for storing a probability value. This method can reduce the model size because all probabilities are stored into the $BASE$ array. As a result, the $VALUE$ array

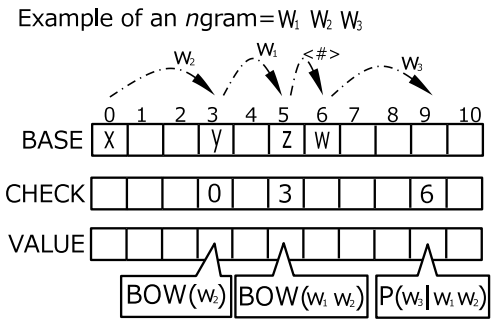


Figure 6: The `simple` DALM data structure. The $BASE$ and $CHECK$ arrays are used in the same way as in a double-array structure. To return probabilities and backoff weights, a $VALUE$ array is introduced.

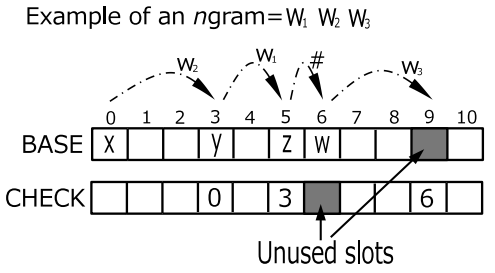


Figure 7: Unused slots in the `simple` DALM structure used for other types of information, such as probabilities.

contains only backoff weights.

Next, the $CHECK$ array slots of *endmarker* symbols are also vacant. We do not need to check for *endmarker* symbol transition because the *endmarker* symbol <#> is seen for all nodes except target word nodes. This means that all *endmarker* symbol transitions are ensured to be correct and the $CHECK$ array slots of *endmarker* symbols do not need to be used. We use this space to store backoff weights.

In order to avoid false positives, we cannot store backoff weights directly. Instead, we store the positions of the backoff weights in the $VALUE$ array as negative numbers. When a query for an unknown $n$gram encounters an *endmarker* symbol node, the value of the $CHECK$ array is never matched because the corresponding value stored there is negative. The same values in the $VALUE$ array can be unified to reduce the memory requirements. Figure 8 illustrates an example of the `embedding` method.

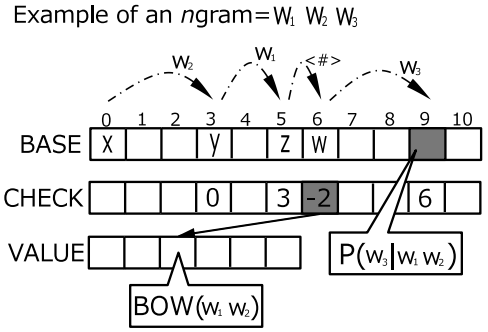


Figure 8: Implementation of the `embedding` method. We use vacant spaces in the $VALUE$ array to store the probabilities and indices of backoff weights. The indices of backoff weights are taken with a negative sign to avoid false positives. Backoff weights are stored in the $VALUE$ array, and the same values in the $VALUE$ array can be unified.

4.3 `Ordering`

`Ordering` is a method for shortening the double-array structure and increasing the query speed. In `ordering`, word IDs are assigned in order of unigram probability. This is done at a preprocessing stage, before the DALM is built.

Before explaining the reasons why this method is effective, we present an *interpretation* of double-array construction in Figure 9 which corresponds to the case presented in Figure 3. In the previous section, we pointed out that the insertion problem is equivalent to the problem of finding the $BASE$ value of the parent node. Here, we expand this further into the idea that finding the $BASE$ value is equivalent to the problem of finding the shift length of an *insertion array*. We can create an *insertion array* which is an array of flag bits set to 1 at the positions of word IDs of children nodes' words. Moreover, we prepare a *used array* which is also a flag bit array denoting whether the original slots in the double-array structure are occupied. In this situation, finding the shift length is equivalent to the problem of finding the $BASE$ value of the slot for the node "eat", and the combined *used array* denotes the size of the double-array structure after insertion.

Figure 10 shows an intuitive example illustrating the efficiency of the `ordering` method. When word IDs are assigned in order of unigram probability, 1s in the *insertion array* are gathered toward

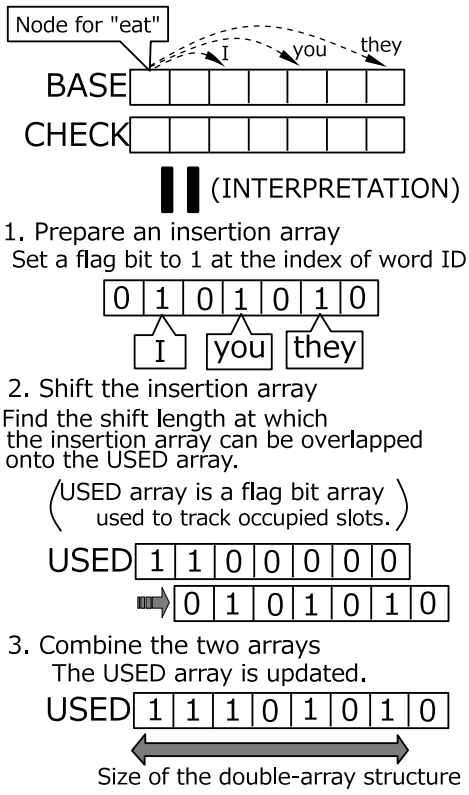


Figure 9: Interpretation of a double-array construction. The insertion problem for the double-array structure is interpreted as a finding problem of a shift length of the insertion array. We can measure the size of the double-array structure in the *used array*.

the beginning of the array. This means that 1s in the *insertion array* form clusters, which makes insertion easier than for unordered *insertion arrays*. This shortens the shift lengths for each insertion array: a shorter double-array structure results.

5 Experiment

5.1 Experimental Setup

To compare the performance of DALM with other methods, we conduct experiments on two $n$gram models built from small and large training corpora. Table 1 shows the specifications of the model.

Training data are extracted from the *Publication of unexamined Japanese patent applications*, which is distributed with the NTCIR 3,4,5,6 patent retrieval task (Atsushi Fujii et al., 2007; Atsushi Fujii et al., 2005; Atsushi Fujii et al., 2004; Makoto Iwayama et al., 2003). We used data for the period from

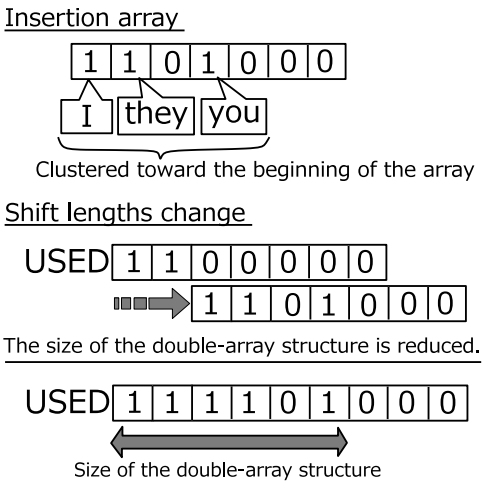


Figure 10: An example of word ID `ordering` efficiency. Word IDs in the insertion array are packed to the front in advance. Therefore, shift lengths for ordered arrays are often shorter than unordered ones. The resulting size of the double-array structure is expected to be smaller than that of an unordered array.

Table 1: Corpus and model specifications.

Model	Corpus Size (words)	Unique Type (words)	Ngram Type (1-5gram)
100 Mwords	100 M	195 K	31 M
5 Gwords	5 G	2,140 K	936 M
Test set	100 M	198 K	-

1,993 to 2,002 and extracted paragraphs containing "background" and "example". This method is similar to the NTCIR 7 Patent Translation Task(Fujii et al., 2008). The small and large training data sets contained 100 Mwords and 5 Gwords, respectively. Furthermore, we sampled another 100 Mwords as a test set to measure the access speed for extracting $n$gram probabilities. We used an Intel® Xeon® X5675 (3.07 GHz) 24-core server with 142 GB of RAM.

Our experiments were performed from the viewpoints of speed and model size. We executed each program twice, and the results of the second run were taken as the final performance.

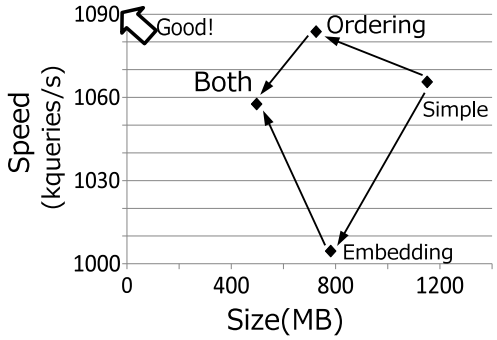


Figure 11: Comparison between tuned and non-tuned double-array structures.

Table 2: Comparison between tuned and non-tuned double-array structures.

Method	Size (MB)	Speed (queries/s)
`Simple`	1,152	1,065,536
`Embedding`	782	1,004,555
`Ordering`	726	1,083,703
`Both`	498	1,057,607

5.2 Optimization Methods

We compared the performance of the DALMs proposed here, namely `simple`, `embedding`, `ordering` and `both`, where `both` indicates that the language model uses both `embedding` and `ordering`. We conducted experiments examining how these methods affect the size of the double-array structures and the query speeds. We used the 100 Mwords model in the comparison because it was difficult to build a DALM using the 5 Gwords model.

The results are shown in Figure 11 and Table 2. While both `ordering` and `embedding` decreased the model size, the query speed was increased by the former and decreased by the latter. `Both` was the smallest and most balanced method.

5.3 Divided Double-Array Structure

Building a double-array structure requires a long time, which can sometimes be impractical. In fact, as mentioned above, waiting on construction of the double-array structure of the 5 Gwords model is infeasible.

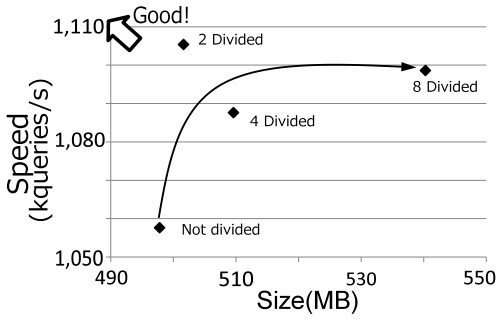


Figure 12: Comparison between divided and original double-array structures.

As described in Section 3.3, the efficient algorithm requires $O(UM)$ time to insert one node and the insertion is iterated N (the total number of insertions) times. If we assume that the number of unused slots at the ith insertion, U_i, is proportional to i, or that $U_i = c \times i$ where c is a proportionality constant, we can calculate the building time as follows: $\sum_{i=1}^{N} U_i M = O\left(MN^2\right)$.

To shorten the build time, we divided the original trie into several parts. Building parts of the original trie is possible because N is reduced. Moreover, these double-array structures can be built in parallel. Note that query results for both original and divided tries are completely equivalent because divided tries hold all the $n$gram statistics of the original trie. This method is similar to that used in *randomized language models* (Talbot and Brants, 2008).

We compared the differences between the methods using the original and divided double-array structures. In the comparison, we also used the 100 Mwords model with the `both` optimization method described in the previous section (Figure 12 and Table 3).

Although dividing the trie increased the size of the DALM slightly, the model size was still smaller than that without optimization. Query speed increased as the number of parts was increased. We attributed this to the divided DALM consisting of several double-array structures, each smaller than the undivided structure which results in an increase. Figure 12 shows that there is a trade-off relation between model size and query speed.

Below, we use the 5 Gwords model in our experiments. In our environment, building a 5 Gwords double-array structure required about 4 days when the double-array structures were divided into 8 parts, even though we used the more efficient algorithm described in Section 3.3. The time required for building the model when the original structure was divided into less than 8 parts was too long. Thus, a more efficient building algorithm is essential for advancing this research further.

Table 3: Comparison between divided and original double-array structures.

Number of parts	Size (MB)	Speed (queries/s)
1	498	1,057,607
2	502	1,105,358
4	510	1,087,619
8	540	1,098,594

5.4 Comparison with Other Methods

Using the 100 Mwords and 5 Gwords models, we compared DALM with other methods (KenLM (Kenneth Heafield, 2011) and SRILM (Stolcke, 2002)). In this experiment, we used the `both` method (which is mentioned above) for DALM and divided the original trie into 8 parts and built double-array structures.

The results are shown in Figure 13 and Table 4; the group on the left shows the results for the 100 Mwords model and the group on the right shows the results for the 5 Gwords model.

The experimental results clearly indicate that DALM is the fastest of all the compared methods and that the model size is nearly the same or slightly smaller than that of KenLM (`Probing`). Whereas KenLM (`Trie`) is the smallest model, it is slower than DALM.

The differences between the 5 Gwords versions of DALM and KenLM (`Probing`) are smaller in comparison with the 100 Mwords models. This is because hash-based language models have an advantage when storing higher-order $n$grams. Large language models have more 5grams, which leads to shorter backoff times. On the other hand, trie-based language models have to trace higher-order $n$grams for every query, which requires more time.

Finally, we discuss practical situations. We con-

Table 4: Comparison between DALM and other methods.

LM	100 Mwords Model		5 Gwords Model	
	Size (MB)	Speed (queries/s)	Size (MB)	Speed (queries/s)
SRILM	1,194	894,138	31,747	729,447
KenLM (`Probing`)	665	1,002,489	18,685	913,208
KenLM (`Trie`)	340	804,513	9,606	635,300
DALM (8 parts)	540	1,098,594	15,345	953,186

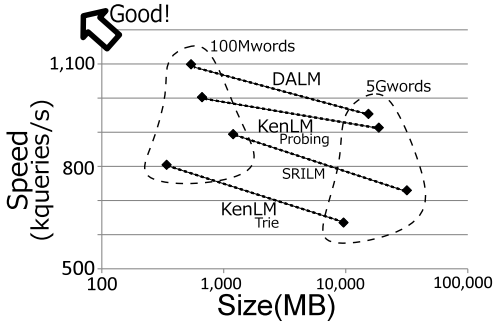


Figure 13: Comparison between DALM and other language model systems.

ducted this study's experiments using test set text written by humans. In some applications such as statistical machine translations, language model systems should compute probabilities of many unnatural $n$grams which will be unknown. This may affect query speed because querying unknown and unnatural $n$grams generate many backoffs. They may results in trie-based LM being slightly faster, because traversing the trie can stop immediately when it detects that a queried $n$gram history is not contained in the trie. On the other hand, hash-based LM such as KenLM probing would repeat queries until finding truncated $n$gram histories in the trie.

6 Conclusion

We proposed a method for implementing language models based on double-array structures. We call this method DALM. Moreover, we proposed two methods for optimizing DALM: `embedding` and `ordering`. `Embedding` is a method whereby empty spaces in arrays are used to store $n$gram probabilities and backoff weights, and `ordering` is a method for numbering word IDs; these methods reduce model size and increase query speed. These two optimization methods work well independently, but even better performance can be achieved if they are combined.

We also used a division method to build the model structure in several parts in order to speed up the construction of double-array structures. Although this procedure results in a slight increase in model size, the divided double-array structures mostly retained the compactness and speed of the original structure. The time required for building double-array structures is the bottleneck of DALM as it is sometimes too long to be practical, even though the model structure itself achieves high performance. In future work, we will develop a faster algorithm for building double-array structures.

While DALM has outperformed state-of-the-art language model implementations methods in our experiments, we should continue to consider ways to optimize the method for higher-order $n$grams.

Acknowledgments

We thank the anonymous reviewers for many valuable comments. This work is supported by JSPS KAKENHI Grant Number 24650063.

References

J.-I. Aoe. 1989. An Efficient Digital Search Algorithm by Using a Double-Array Structure. *IEEE Transactions on Software Engineering*, 15(9):1066–1077.

Atsushi Fujii, Makoto Iwayama, and Noriko Kando. 2004. Overview of the Patent Retrieval Task at NTCIR-4.

Atsushi Fujii, Makoto Iwayama, and Noriko Kando. 2005. Overview of Patent Retrieval Task at NTCIR-5.

Atsushi Fujii, Makoto Iwayama, and Noriko Kando. 2007. Overview of the Patent Retrieval Task at the NTCIR-6 Workshop. pages 359–365.

Djamal Belazzougui, Fabiano C. Botelho, and Martin Dietzfelbinger. 2009. Hash, displace, and compress. In *ESA*, pages 682–693.

Timothy C. Bell, John G. Cleary, and Ian H. Witten. 1990. Text compression.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on EMNLP-CoNLL*, pages 858–867. ACL.

B. Merialdo F. Jelinek. 1990. Self-organized language modeling for speech recognition.

Edward Fredkin. 1960. Trie memory. *Communications of the ACM*, 3(9):490–499.

Kimmo Fredriksson and Fedor Nikitin. 2007. Simple Compression Code Supporting Random Access and Fast String Matching. In *Proceedings of the 6th international conference on Experimental algorithms*, WEA'07, pages 203–216. Springer-Verlag.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2008. Overview of the Patent Translation Task at the NTCIR-7 Workshop.

Ulrich Germann, Eric Joanis, and Samuel Larkin. 2009. Tightly Packed Tries: How to Fit Large Models into Memory, and Make them Load Fast, Too. In *Proceedings of the Workshop on SETQA-NLP*, pages 31–39. ACL.

David Guthrie and Mark Hepple. 2010. Storing the Web in Memory: Space Efficient Language Models with Constant Time Retrieval. In *Proceedings of the 2010 Conference on EMNLP*, pages 262–272. ACL.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. ACL.

Makoto Iwayama, Atsushi Fujii, Noriko Kando, and Akihiko Takano. 2003. Overview of Patent Retrieval Task at NTCIR-3.

Yasumasa Nakamura and Hisatoshi Mochizuki. 2006. Fast Computation of Updating Method of a Dictionary for Compression Digital Search Tree. *Transactions of Information Processing Society of Japan. Data*, 47(13):16–27.

Adam Pauls and Dan Klein. 2011. Faster and Smaller N-Gram Language Models. In *Proceedings of the 49th Annual Meeting of the ACL-HLT*, pages 258–267. ACL.

A. Stolcke. 2002. SRILM-an Extensible Language Modeling Toolkit. *Seventh International Conference on Spoken Language Processing*.

David Talbot and Thorsten Brants. 2008. Randomized Language Models via Perfect Hash Functions. In *Proceedings of ACL-08: HLT*.

David Talbot and Miles Osborne. 2007. Smoothed Bloom Filter Language Models: Tera-Scale LMs on the Cheap. In *Proceedings of the 2007 Joint Conference on EMNLP-CoNLL*, pages 468–476.

Structured Penalties for Log-linear Language Models

Anil Nelakanti,[*‡] **Cédric Archambeau,**[*] **Julien Mairal,**[†] **Francis Bach,**[‡] **Guillaume Bouchard**[*]
[*]Xerox Research Centre Europe, Grenoble, France
[†]INRIA-LEAR Project-Team, Grenoble, France
[‡]INRIA-SIERRA Project-Team, Paris, France
firstname.lastname@xrce.xerox.com firstname.lastname@inria.fr

Abstract

Language models can be formalized as log-linear regression models where the input features represent previously observed contexts up to a certain length m. The complexity of existing algorithms to learn the parameters by maximum likelihood scale linearly in nd, where n is the length of the training corpus and d is the number of observed features. We present a model that grows logarithmically in d, making it possible to efficiently leverage longer contexts. We account for the sequential structure of natural language using tree-structured penalized objectives to avoid overfitting and achieve better generalization.

1 Introduction

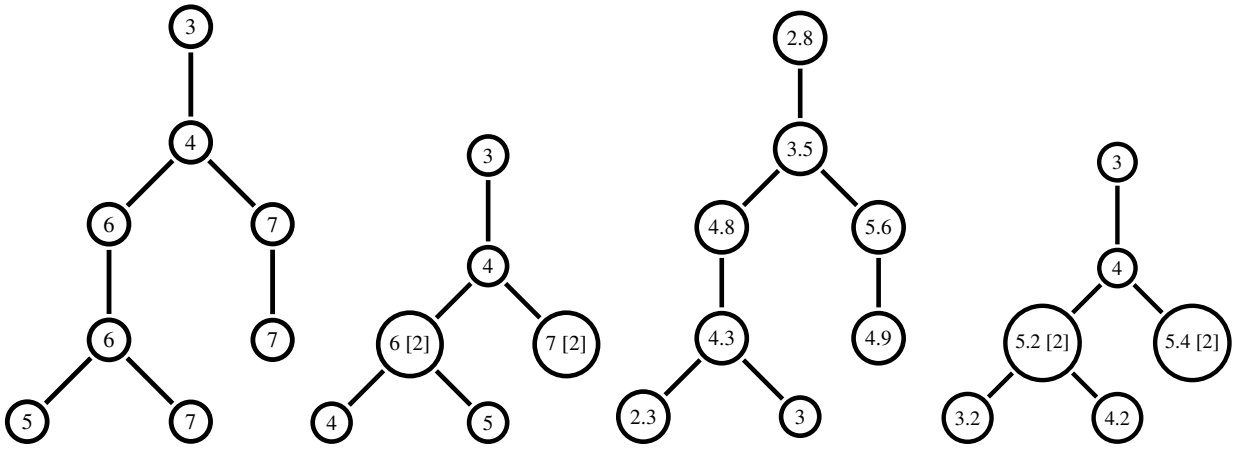
Language models are crucial parts of advanced natural language processing pipelines, such as speech recognition (Burget et al., 2007), machine translation (Chang and Collins, 2011), or information retrieval (Vargas et al., 2012). When a sequence of symbols is observed, a language model predicts the probability of occurrence of the next symbol in the sequence. Models based on so-called back-off smoothing have shown good predictive power (Goodman, 2001). In particular, Kneser-Ney (KN) and its variants (Kneser and Ney, 1995) are still achieving state-of-the-art results for more than a decade after they were originally proposed. Smoothing methods are in fact clever heuristics that require tuning parameters in an ad-hoc fashion. Hence, more principled ways of learning language models have been proposed based on maximum entropy (Chen and Rosenfeld, 2000) or conditional random fields (Roark et al., 2004), or by adopting a Bayesian approach (Wood et al., 2009).

In this paper, we focus on penalized maximum likelihood estimation in log-linear models. In contrast to language models based on *unstructured* norms such as ℓ_2 (quadratic penalties) or ℓ_1 (absolute discounting), we use *tree-structured* norms (Zhao et al., 2009; Jenatton et al., 2011). Structured penalties have been successfully applied to various NLP tasks, including chunking and named entity recognition (Martins et al., 2011), but not language modelling. Such penalties are particularly well-suited to this problem as they mimic the nested nature of word contexts. However, existing optimizing techniques are not scalable for large contexts m.

In this work, we show that structured tree norms provide an efficient framework for language modelling. For a special case of these tree norms, we obtain an memory-efficient learning algorithm for log-linear language models. Furthermore, we aslo give the first efficient learning algorithm for structured ℓ_∞ tree norms with a complexity nearly linear in the number of training samples. This leads to a memory-efficient *and* time-efficient learning algorithm for generalized linear language models.

The paper is organized as follows. The model and other preliminary material is introduced in Section 2. In Section 3, we review unstructured penalties that were proposed earlier. Next, we propose structured penalties and compare their memory and time requirements. We summarize the characteristics of the proposed algorithms in Section 5 and experimentally validate our findings in Section 6.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 233–243,
Seattle, Washington, USA, 18-21 October 2013.



(a) Trie-structured vector. $w = [\,3\,4\,6\,6\,4\,5\,7\,7\,]^\top$.

(b) Tree-structured vector. $w = [\,3\,4\,6\,6\,4\,5\,7\,7\,]^\top$.

(c) ℓ_2^T-proximal $\Pi_{\ell_2^T}(w, 0.8) =$ $[\,2.8\,3.5\,4.8\,4.3\,2.3\,3\,5.6\,4.9\,]^\top$.

(d) ℓ_∞^T-proximal $\Pi_{\ell_\infty^T}(w, 0.8) =$ $[\,3\,4\,5.2\,5.2\,3.2\,4.2\,5.4\,5.4\,]^\top$.

Figure 1: Example of uncollapsed (trie) and corresponding collapsed (tree) structured vectors and proximal operators applied to them. Weight values are written inside the node. Subfigure (a) shows the complete trie S and Subfigure (b) shows the corresponding collapsed tree T. The number in the brackets shows the number of nodes collapsed. Subfigure (c) shows vector after proximal projection for ℓ_2^T-norm (which cannot be collapsed), and Subfigure (d) that of ℓ_∞^T-norm proximal projection which can be collapsed.

2 Log-linear language models

Multinomial logistic regression and Poisson regression are examples of log-linear models (McCullagh and Nelder, 1989), where the likelihood belongs to an exponential family and the predictor is linear. The application of log-linear models to language modelling was proposed more than a decade ago (Della Pietra et al., 1997) and it was shown to be competitive with state-of-the-art language modelling such as Knesser-Ney smoothing (Chen and Rosenfeld, 2000).

2.1 Model definition

Let V be a set of words or more generally a set of symbols, which we call vocabulary. Further, let xy be a sequence of $n+1$ symbols of V, where $x \in V^n$ and $y \in V$. We model the probability that symbol y succeeds x as

$$P(y = v|x) = \frac{e^{w_v^\top \phi_m(x)}}{\sum_{u \in V} e^{w_u^\top \phi_m(x)}}, \qquad (1)$$

where $W = \{w_v\}_{v \in V}$ is the set of parameters, and $\phi_m(x)$ is the vector of features extracted from x, the sequence preceding y. We will describe the features shortly.

Let $x_{1:i}$ denote the subsequence of x starting at the first position up to the i^{th} position and y_i the next symbol in the sequence. Parameters are estimated by minimizing the penalized log-loss:

$$W^* \in \operatorname*{argmin}_{W \in \mathcal{K}} f(W) + \lambda \Omega(W), \qquad (2)$$

where $f(W) := -\sum_{i=1}^n \ln p(y_i|x_{1:i}; W)$ and $\mathcal{K}$ is a convex set representing the constraints applied on the parameters. Overfitting is avoided by adjusting the regularization parameter λ, e.g., by cross-validation.

2.2 Suffix tree encoding

Suffix trees provide an efficient way to store and manipulate discrete sequences and can be constructed in linear time when the vocabulary is fixed (Giegerich and Kurtz, 1997). Recent examples include language models based on a variable-length Markovian assumption (Kennington et al., 2012) and the sequence memoizer (Wood et al., 2011). The suffix tree data structure encodes all the unique suffixes observed in a sequence up to a maximum given length. It exploits the fact that the set of observed contexts is a small subset of all possible contexts. When a series of suffixes of increasing lengths are

Algorithm 1 $W^* := \operatorname{argmin} \{f(X, Y; W) + \lambda\Omega(W)\}$ Stochastic optimization algorithm (Hu et al., 2009)

Input: λ regularization parameter , L Lipschitz constant of ∇f , μ coefficient of strong-convexity of $f + \lambda\Omega$, X design matrix, Y label set
Initialize: $\overline{W} = Z = 0$, $\tau = \delta = 1$, $\rho = L + \mu$
repeat until maximum iterations
  #estimate point for gradient update
  $\overline{W} = (1 - \tau)W + \tau Z$
  #use mini-batch $\{X_\vartheta, Y_\vartheta\}$ for update
  W = ParamUpdate$(X_\vartheta, Y_\vartheta, \overline{W}, \lambda, \rho)$
  #weighted combination of estimates
  $Z = \frac{1}{\rho\tau + \mu}\big((1 - \mu)Z + (\mu - \rho)\overline{W} + \rho W\big)$
  #update constants
  $\rho = L + \mu/\delta$, $\tau = \frac{\sqrt{4\delta + \delta^2} - \delta}{2}$, $\delta = (1 - \tau)\delta$

Procedure: $W :=$ ParamUpdate$(X_\vartheta, Y_\vartheta, \overline{W}, \lambda, \rho)$
  $W' = \overline{W} - \frac{1}{\rho}\nabla f(X_\vartheta, Y_\vartheta, \overline{W})$ #gradient step
  $W = [W]_+$ #projection to non-negative orthant
  $W = \Pi_\Omega(w, \kappa)$ #proximal step

always observed in the same context, the successive suffixes are collapsed into a single node. The uncollapsed version of the suffix tree T is called a suffix *trie*, which we denote S. A suffix trie also has a tree structure, but it potentially has much larger number of nodes. An example of a suffix trie S and the associated suffix tree T are shown in Figures 1(a) and 1(b) respectively. We use $|S|$ to denote the number of nodes in the trie S and $|T|$ for the number of nodes in the tree T.

Suffix tree encoding is particularly helpful in applications where the resulting hierarchical structures are thin and tall with numerous non-branching paths. In the case of text, it has been observed that the number of nodes in the tree grows slower than that of the trie with the length of the sequence (Wood et al., 2011; Kennington et al., 2012). This is a significant gain in the memory requirements and, as we will show in Section 4, can also lead to important computational gains when this structure is exploited.

The feature vector $\phi_m(x)$ encodes suffixes (or contexts) of increasing length up to a maximum length m. Hence, the model defined in (1) is similar to m-gram language models. Naively, the feature vector $\phi_m(x)$ corresponds to one path of length m starting at the root of the suffix trie S. The entries in W correspond to weights for each suffix. We thus have a trie structure S on W (see Figure 1(a)) constraining the number of free parameters. In other words, there is one weight parameter per node in the trie S and the matrix of parameters W is of size $|S|$.

In this work, however, we consider models where the number of parameters is equal to the size of the suffix tree T, which has much fewer nodes than S. This is achieved by ensuring that all parameters corresponding to suffixes at a node share the same parameter value (see Figure 1(b)). These parameters correspond to paths in the suffix trie that do not branch *i.e.* sequence of words that always appear together in the same order.

2.3 Proximal gradient algorithm

The objective function (2) involves a smooth convex loss f and a possibly non-smooth penalty Ω. Subgradient descent methods for non-smooth Ω could be used, but they are unfortunately very slow to converge. Instead, we choose proximal methods (Nesterov, 2007), which have fast convergence rates and can deal with a large number of penalties Ω, see (Bach et al., 2012).

Proximal methods iteratively update the current estimate by making a generalized gradient update at each iteration. Formally, they are based on a linearization of the smooth function f around a parameter estimate $\overline{W}$, adding a quadratic penalty term to keep the updated estimate in the neighborhood of $\overline{W}$. At iteration t, the update of the parameter W is given by

$$W^{t+1} = \operatorname*{argmin}_{W \in \mathcal{K}} \Big\{ f(\overline{W}) + (W - \overline{W})^\top \nabla f(\overline{W}) + \Omega(W) + \frac{L}{2}\|W - \overline{W}\|_2^2 \Big\}, \quad (3)$$

where $L > 0$ is an upper-bound on the Lipschitz constant of the gradient ∇f. The matrix $\overline{W}$ could either be the current estimate W^t or its weighted combination with the previous estimate for accelerated convergence depending on the specific algorithm used (Beck and Teboulle, 2009). Equation (3) can be rewritten to be solved in two independent steps: a gradient update from the smooth part followed by a projection depending only on the non-smooth penalty:

$$W' = \overline{W} - \frac{1}{L}\nabla f(\overline{W}), \quad (4)$$

$$W^{t+1} = \operatorname*{argmin}_{W \in \mathcal{K}} \frac{1}{2} \left\| W - W' \right\|_2^2 + \frac{\lambda \Omega(W)}{L}. \quad (5)$$

Update (5) is called the proximal operator of W' with parameter $\frac{\lambda}{L}$ that we denote $\Pi_\Omega \left(W', \frac{\lambda}{L} \right)$. Efficiently computing the proximal step is crucial to maintain the fast convergence rate of these methods.

2.4 Stochastic proximal gradient algorithm

In language modelling applications, the number of training samples n is typically in the range of 10^5 or larger. Stochastic version of the proximal methods (Hu et al., 2009) have been known to be well adapted when n is large. At every update, the stochastic algorithm estimates the gradient on a mini-batch, that is, a subset of the samples. The size of the mini-batches controls the trade-off between the variance in the estimate of gradient and the time required for compute it. In our experiments we use mini-batches of size 400. The training algorithm is summarized in Algorithm 1. The acceleration is obtained by making the gradient update at a specific weighted combination of the current and the previous estimates of the parameters. The weighting is shown in step 6 of the Algorithm 1.

2.5 Positivity constraints

Without constraining the parameters, the memory required by a model scales linearly with the vocabulary size $|V|$. Any symbol in V observed in a given context is a positive example, while any symbols in V that does not appear in this context is a negative example. When adopting a log-linear language model, the negative examples are associated with a small negative gradient step in (4), so that the solution is not sparse accross multiple categories in general. By constraining the parameters to be positive (i.e., the set of feasible solutions $\mathcal{K}$ is the positive orthant), the projection step 2 in Algorithm 1 can be done with the same complexity, while maintaining sparse parameters accross multiple categories. More precisely, the weights for the category k associated to a given context x, is always zeros if the category k never occured after context x. A significant gain in memory (nearly $|V|$-fold for large context lengths) was obtained without loss of accuracy in our experiments.

3 Unstructured penalties

Standard choices for the penalty function $\Omega(W)$ include the ℓ_1-norm and the squared ℓ_2-norm. The former typically leads to a solution that is sparse and easily interpretable, while the latter leads to a non-sparse, generally more stable one. In particular, the squared ℓ_2 and ℓ_1 penalties were used in the context of log-linear language models (Chen and Rosenfeld, 2000; Goodman, 2004), reporting performances competitive with bi-gram and tri-gram interpolated Kneser-Ney smoothing.

3.1 Proximal step on the suffix trie

For squared ℓ_2 penalties, the proximal step $\Pi_{\ell_2^2}(w^t, \frac{\kappa}{2})$ is the element-wise rescaling operation:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} (1 + \kappa)^{-1} \quad (6)$$

For ℓ_1 penalties, the proximal step $\Pi_{\ell_1}(w^t, \kappa)]$ is the soft-thresholding operator:

$$w_i^{(t+1)} \leftarrow \max(0, w_i^{(t)} - \kappa). \quad (7)$$

These projections have linear complexity in the number of features.

3.2 Proximal step on the suffix tree

When feature values are identical, the corresponding proximal (and gradient) steps are identical. This can be seen from the proximal steps (7) and (6), which apply to single weight entries. This property can be used to group together parameters for which the feature values are equal. Hence, we can collapse successive nodes that always have the same values in a suffix trie (as in Figure 1(b)), that is to say we can directly work on the suffix tree. This leads to a proximal step with complexity that scales linearly with the number of symbols seen in the corpus (Ukkonen, 1995) and logarithmically with context length.

4 Structured penalties

The ℓ_1 and squared ℓ_2 penalties do not account for the sequential dependencies in the data, treating suffixes of different lengths equally. This is inappropriate considering that longer suffixes are typically observed less frequently than shorter ones. Moreover, the fact that suffixes might be nested is disregarded. Hence, we propose to use the tree-structured

Algorithm 2 $w := \Pi_{\ell_2^T}(w, \kappa)$ Proximal projection step for ℓ_2^T on grouping $\mathcal{G}$.

Input: T suffix tree, w trie-structured vector, κ threshold
Initialize: $\{\gamma_i\} = 0$, $\{\eta_i\} = 1$
η = UpwardPass(η, γ, κ, w)
w = DownwardPass(η, w)

Procedure: η := UpwardPass(η, γ, κ, w)
for $x \in$ DepthFirstSuffixTraversal(T, *PostOrder*)
  $\gamma_x = w_x^2 + \sum_{h \in children(x)} \gamma_h$
  $\eta_x = [1 - \kappa/\sqrt{\gamma_x}]_+$
  $\gamma_x = \eta_x^2 \gamma_x$

Procedure: w := DownwardPass(η, w)
for $x \in$ DepthFirstSuffixTraversal(T, *PreOrder*)
  $w_x = \eta_x w_x$
  for $h \in children(x)$
    $\eta_h = \eta_x \eta_h$

a DepthFirstSuffixTraversal(T,*Order*) returns observed suffixes from the suffix tree T by depth-first traversal in the order prescribed by *Order*.
b w_x is the weights corresponding to the suffix x from the weight vector w and $children(x)$ returns all the immediate children to suffix x in the tree.

Algorithm 3 $w := \Pi_{\ell_\infty^T}(w, \kappa)$ Proximal projection step for ℓ_∞^T on grouping $\mathcal{G}$.

Input: T suffix tree, w=[v c] tree-structured vector v with corresponding number of suffixes collapsed at each node in c, κ threshold
for $x \in$ DepthFirstNodeTraversal(T, *PostOrder*)
  $g(v, x) := \pi_{\ell_\infty^T}(\, g(v, x), c_x \kappa\,)$

Procedure: $q := \pi_{\ell_\infty}(q, \kappa)$
Input: $q = [v\ c],\ q_i = [v_i\ c_i],\ i = 1, \cdots, |q|$
Initialize: $U = \{\}, L = \{\}, I = \{1, \cdots, |q|\}$
while $I \neq \varnothing$
  pick random $\rho \in I$ #choose pivot
  $U = \{j | v_j \geq v_\rho\}$ #larger than v_ρ
  $L = \{j | v_j < v_\rho\}$ #smaller than v_ρ
  $\delta S = \sum_{i \in U} v_i \cdot c_i,\ \delta C = \sum_{i \in U} c_i$
  if $(S + \delta S) - (C + \delta C)\rho < \kappa$
    $S := (S + \delta S),\ C := (C + \delta C),\ I := L$
  else $I := U \backslash \{\rho\}$
$r = \frac{S - \kappa}{C}$, $v_i := v_i - \max(0, v_i - r)$ #take residuals

a DepthFirstNodeTraversal(T,*Order*) returns nodes x from the suffix tree T by depth-first traversal in the order prescribed by *Order*.

norms (Zhao et al., 2009; Jenatton et al., 2011), which are based on the suffix trie or tree, where subtrees correspond to contexts of increasing lengths. As will be shown in the experiments, this prevents the model to overfit unlike the ℓ_1- or squared ℓ_2-norm.

4.1 Definition of tree-structured ℓ_p^T norms

Definition 1. *Let x be a training sequence. Group $g(w, j)$ is the subvector of w associated with the subtree rooted at the node j of the suffix trie $S(x)$.*

Definition 2. *Let $\mathcal{G}$ denote the ordered set of nodes of the tree $T(x)$ such that for $r < s$, $g(w, r) \cap g(w, s) = \emptyset$ or $g(w, r) \subset g(w, s)$. The tree-structured ℓ_p-norm is defined as follows:*

$$\ell_p^T(w) = \sum_{j \in \mathcal{G}} \|g(w, j)\|_p \,. \tag{8}$$

We specifically consider the cases $p = 2, \infty$ for which efficient optimization algorithms are available. The ℓ_p^T-norms can be viewed as a group sparsity-inducing norms, where the groups are organized in a tree. This means that when the weight associated with a parent in the tree is driven to zero, the weights associated to all its descendants should also be driven to zero.

For structured ℓ_p^T-norm, the proximal step amounts to residuals of recursive projections on the ℓ_q-ball in the order defined by $\mathcal{G}$ (Jenatton et al., 2011), where ℓ_q-norm is the dual norm of ℓ_p-norm[1]. In the case ℓ_2^T-norm this comes to a series of projections on the ℓ_2-ball. For ℓ_∞^T-norm it is instead projections on the ℓ_1-ball. The order of projections defined by $\mathcal{G}$ is generated by an upward pass of the suffix trie. At each node through the upward pass, the subtree below is projected on the dual norm ball of size κ, the parameter of proximal step. We detail the projections on the norm ball below.

4.2 Projections on ℓ_q-ball for $q = 1, 2$

Each of the above projections on the dual norm ball takes one of the following forms depending on the choice of the norm. Projection of vector w on the ℓ_2-ball is equivalent to thresholding the magnitude of w by κ units while retaining its direction:

$$w \leftarrow [||w||_2 - \kappa]_+ \frac{w}{||w||_2}. \tag{9}$$

This can be performed in time linear in size of w, $O(|w|)$. Projection of a non-negative vector w on the ℓ_1-ball is more involved and requires thresholding

[1] ℓ_p-norm and ℓ_q-norm are dual to each other if $\frac{1}{p} + \frac{1}{q} = 1$. ℓ_2-norm is self-dual while the dual of ℓ_∞-norm is the ℓ_1-norm.

by a value such that the entries in the resulting vector add up to κ, otherwise w remains the same:

$$w \leftarrow [w - \tau]_+ \text{ s.t. } ||w||_1 = \kappa \text{ or } \tau = 0. \quad (10)$$

$\tau = 0$ is the case where w lies inside the ℓ_1-ball of size κ with $||w||_1 < \kappa$, leaving w intact. In the other case, the threshold τ is to be computed such that after thresholding, the resulting vector has an ℓ_1-norm of κ. The simplest way to achieve this is to sort by descending order the entries $\overline{w} = \text{sort}(w)$ and pick the k largest values such that the $(k+1)^{th}$ largest entry is smaller than τ:

$$\sum_{i=1}^{k} \overline{w}_i - \tau = \kappa \text{ and } \tau > \overline{w}_{k+1}. \quad (11)$$

We refer to $\overline{w}_k$ as the pivot and are only interested in entries larger than the pivot. Given a sorted vector, it requires looking up to exactly k entries, however, sorting itself take $O(|w| \log |w|)$.

4.3 Proximal step

Naively employing the projection on the ℓ_2-ball described above leads to an $O(d^2)$ algorithm for ℓ_2^T proximal step. This could be improved to a linear algorithm by aggregating all necessary scaling factors while making an upward pass of the trie S and applying them in a single downward pass as described in (Jenatton et al., 2011). In Algorithm 2, we detail this procedure for trie-structured vectors.

The complexity of ℓ_∞^T-norm proximal step depends directly on that of the pivot finding algorithm used within its ℓ_1-projection method. Naively sorting vectors to find the pivot leads to an $O(d^2 \log d)$ algorithm. Pivot finding can be improved by randomly choosing candidates for the pivot and the best known algorithm due to (Bruckner, 1984) has amortized linear time complexity in the size of the vector. This leaves us with $O(d^2)$ complexity for ℓ_∞^T-norm proximal step. (Duchi et al., 2008) proposes a method that scales linearly with the number of non-zero entries in the gradient update (s) but logarithmically in d. But recursive calls to ℓ_1-projection over subtrees will fail the sparsity assumption (with $s \approx d$) making proximal step quadratic. Procedure for $\Pi_{\ell_\infty^T}$ on trie-structured vectors using randomized pivoting method is described in Algorithm 3.

We next explain how the number of ℓ_1-projections can be reduced by switching to the tree T instead of trie S which is possible due to the good properties of ℓ_∞^T-norm. Then we present a pivot finding method that is logarithmic in the feature size for our application.

4.4 ℓ_∞^T-norm with suffix trees

We consider the case where all parameters are initialized with the same value for the optimization procedure, typically with zeros. The condition that the parameters at any given node continue to share the same value requires that both the gradient update (4) and proximal step (5) have this property. We modify the tree structure to ensure that after gradient updates parameters at a given node continue to share a single value. Nodes that do not share a value after gradient update are split into multiple nodes where each node has a single value. We formally define this property as follows:

Definition 3. *A **constant value non-branching path** is a set of nodes $P \in \mathcal{P}(T, w)$ of a tree structure T w.r.t. vector w if P has $|P|$ nodes with $|P| - 1$ edges between them and each node has at most one child and all nodes $i, j \in P$ have the same value in vector w as $w_i = w_j$.*

The nodes of Figure 1(b) correspond to constant value non-branching paths when the values for all parameters at each of the nodes are the same. Next we show that this tree structure is retained after proximal steps of ℓ_∞^T-norm.

Proposition 1. *Constant value non-branching paths $\mathcal{P}(T, w)$ of T structured vector w are preserved under the proximal projection step $\Pi_{\ell_\infty^T}(w, \kappa)$.*

Figure 1(d) illustrates this idea showing ℓ_∞^T projection applied on the collapsed tree. This makes it memory efficient but the time required for the proximal step remains the same since we must project each subtree of S on the ℓ_1-ball. The sequence of projections at nodes of S in a non-branching path can be rewritten into a single projection step using the following technique bringing the number of projections from $|S|$ to $|T|$.

Proposition 2. *Successive projection steps for subtrees with root in a constant value non-branching path $P = \{g_1, \cdots, g_{|P|}\} \in \mathcal{P}(T, w)$ for $\Pi_{\ell_\infty^T}(w, \kappa)$*

is $\pi_{g_{|P|}} \circ \cdots \circ \pi_{g_1}(w, \kappa)$ applied in bottom-up order defined by $\mathcal{G}$. The composition of projections can be rewritten into a single projection step with κ scaled by the number of projections $|P|$ as,

$$\pi_{g_{|P|}}(w, \kappa|P|) \equiv \pi_{g_{|P|}} \circ \cdots \circ \pi_{g_1}(w, \kappa).$$

The above propositions show that ℓ_∞^T-norm can be used with the suffix tree with fewer projection steps. We now propose a method to further improve each of these projection steps.

4.5 Fast proximal step for ℓ_∞^T-norm

Let k be the cardinality of the set of values larger than the pivot in a vector to compute the threshold for ℓ_1-projection as referred in (11). This value varies from one application to another, but for language applications, our experiments on 100K english words (APNews dataset) showed that k is generally small: its value is on average 2.5, and its maximum is around 10 and 20, depending on the regularization level. We propose using a max-heap data structure (Cormen et al., 1990) to fetch the k-largest values necessary to compute the threshold. Given the heap of the entries the cost of finding the pivot is $O(k \log(d))$ if the pivot is the k^{th} largest entry and there are d features. This operation is performed d times for ℓ_∞^T-norm as we traverse the tree bottom-up. The heap itself is built on the fly during this upward pass. At each subtree, the heap is built by merging those of their children in constant time by using Fibonacci heaps. This leaves us with a $O(dk \log(d))$ complexity for the proximal step. This procedure is detailed in Algorithm 4.

Algorithm 4 $w := \Pi_{\ell_\infty^T}(w, \kappa)$ Proximal projection step for ℓ_∞^T on grouping $\mathcal{G}$ using heap data structure.

Input: T suffix tree, w=[v c] tree-structured vector v with corresponding number of suffixes collapsed at each node in c, κ threshold
Initialize $\mathcal{H}$ = {} # empty set of heaps
1 **for** $x \in$ DepthFirstNodeTraversal(T, *PostOrder*)
  $g(v, x) := \pi_{\ell_\infty^T}(w, x, c_x\kappa, \mathcal{H})$

Procedure: $q := \pi_{\ell_\infty}(w, x, \kappa, \mathcal{H})$
1 $\mathcal{H}_x = \text{NewHeap}(v_x, c_x, v_x)$
2 **for** $j \in \text{children}(x)$ # merge with child heaps
  $\tau_x = \tau_x + \tau_j$ # update ℓ_1-norm
  $\mathcal{H}_x = \text{Merge}(\mathcal{H}_x, \mathcal{H}_j), \mathcal{H} = \mathcal{H} \backslash \mathcal{H}_j$
3 $\mathcal{H} = \mathcal{H} \cup H_x, S = 0, C = 0, J = \{\}$
4 **if** $\mathcal{H}_x(\tau) < \kappa$, **set** $\mathcal{H}_x = 0$ **return**
5 **for** $j \in \text{OrderedIterator}(\mathcal{H}_x)$ # get max values
  if $v_j > \frac{S+(v_j c_j)-\kappa}{C+c_j}$
    $S = S + (v_j \cdot c_j), C = C + c_j, J = J \cup \{j\}$
  else break
6 $r = \frac{S-\kappa}{C}, \delta = 0$ # compute threshold
7 **for** $j \in J$ # apply threshold
  $\nu = \min(v_j, r), \delta = \delta + (v_j - \nu)$
  $\mathcal{H}_j(v) = \nu$
8 $\mathcal{H}_x(\tau) = \mathcal{H}_j(\tau) - \delta$ # update ℓ_1-norm

a. Heap structure on vector w holds three values (v, c, τ) at each node. v, c being value and its count, τ is the ℓ_1-norm of the sub-vector below. Tuples are ordered by decreasing value of v and $\mathcal{H}_j$ refers to heap with values in sub-tree rooted at j. Merge operation merges the heaps passed. OrderedIterator returns values from the heap in decreasing order of v.

5 Summary of the algorithms

Table 1 summarizes the characteristics of the algorithms associated to the different penalties:

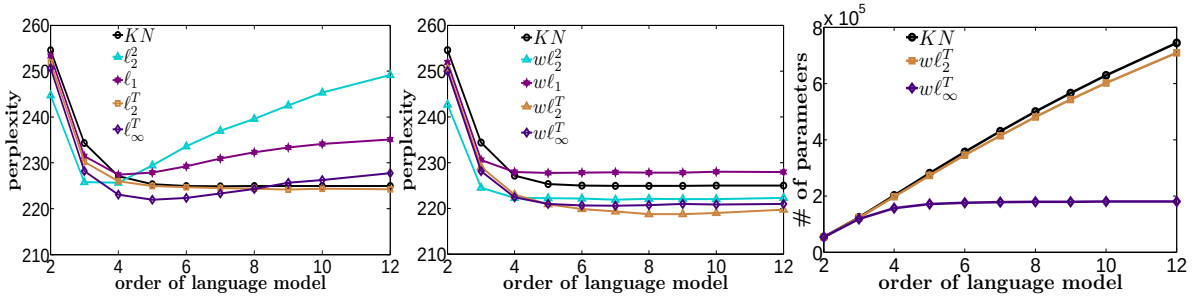
1. The unstructured norms ℓ_p do not take into account the varying sparsity level with context length. For p=1, this leads to a sparse solution and for p=2, we obtain the classical quadratic penalty. The suffix tree representation leads to an efficient memory usage. Furthermore, to make the training algorithm time efficient, the parameters corresponding to contexts which always occur in the same larger context are grouped. We will illustrate in the experiments that these penalties do not lead to good predictive performances.

2. The ℓ_2^T-norm nicely groups features by subtrees which concurs with the sequential structure of sequences. This leads to a powerful algorithm in terms of generalization. But it can only be applied on the uncollapsed tree since there is no closure property of the *constant value non-branching path* for its proximal step making it less amenable for larger tree depths.

3. The ℓ_∞^T-norm groups features like the ℓ_2^T-norm while additionally encouraging numerous feature groups to share a single value, leading to a substantial reduction in memory usage. The generalization properties of this algorithm is as good as the generalization obtained with the ℓ_2^T penalty, if not better. However, it has the *constant value non-branching path* property, which

Penalty			good generalization	memory efficient	time efficient
unstructured ℓ_1 and ℓ_2^2			no	yes $_{O(\|T\|)}$	yes $_{O(\|T\|)}$
struct.	ℓ_2^T		yes	no $_{O(\|S\|)}$	no $_{O(\|S\|)}$
	ℓ_∞^T	rand. pivot	yes	yes $_{O(\|T\|)}$	no $_{O(\|T\|^2)}$
	ℓ_∞^T	heap	yes	yes $_{O(\|T\|)}$	yes $_{O(\|T\|\log\|T\|)}$

Table 1: Properties of the algorithms proposed in this paper. Generalization properties are as compared by their performance with increasing context length. Memory efficiency is measured by the number of free parameters of W in the optimization. Note that the suffix tree is much smaller than the trie (uncollapsed tree): $|T| << |S|$. Time complexities reported are that of one proximal projection step.



(a) Unweighted penalties. (b) Weighted penalties. (c) Model complexity for structured penalties.

Figure 2: (a) compares average perplexity (lower is better) of different methods from 2-gram through 12-gram on four different 100K-20K train-test splits. (b) plot compares the same with appropriate feature weighting. (c) compares model complexity for weighted structured penalties $w\ell_2^T$ and $w\ell_\infty^T$ measure by then number of parameters.

means that the proximal step can be applied directly to the suffix tree. There is thus also a significant gain of performances.

6 Experiments

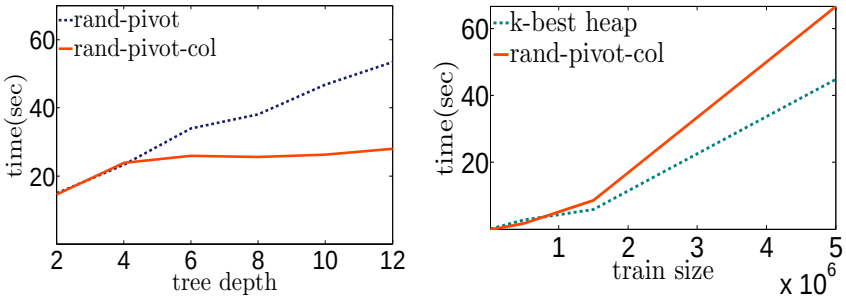
In this section, we demonstrate empirically the properties of the algorithms summarized in Table 1. We consider four distinct subsets of the Associated Press News (AP-news) text corpus with train-test sizes of 100K-20K for our experiments. The corpus was preprocessed as described in (Bengio et al., 2003) by replacing proper nouns, numbers and rare words with special symbols "⟨proper_noun⟩", "#n" and "⟨unknown⟩" respectively. Punctuation marks are retained which are treated like other normal words. Vocabulary size for each of the training subsets was around 8,500 words. The model was reset at the start of each sentence, meaning that a word in any given sentence does not depend on any word in the previous sentence. The regularization parameter λ is chosen for each model by cross-validation on a smaller subset of data. Models are fitted to training sequence of 30K words for different values of λ and validated against a sequence of 10K words to choose λ.

We quantitatively evaluate the proposed model using perplexity, which is computed as follows:

$$P(\{x_i, y_i\}, W) = 10^{\left\{\frac{-1}{n_V}\sum_{i=1}^{n}\mathbb{I}(y_i \in V)\log p(y_i|x_{1:i};W)\right\}},$$

where $n_V = \sum_i \mathbb{I}(y_i \in V)$. Performance is measured for varying depth of the suffix trie with different penalties. Interpolated Kneser-Ney results were computed using the openly available SRILM toolkit (Stolcke, 2002).

Figure 2(a) shows perplexity values averaged over four data subsets as a function of the language model order. It can be observed that performance of unstructured ℓ_1 and squared ℓ_2 penalties improve until a relatively low order and then degrade, while ℓ_2^T penalty does not show such degradation, indicating



(a) Iteration time of random-pivoting on the collapsed and uncollapsed trees.

(b) Iteration time of random-pivoting and k-best heap on the collapsed tree.

Figure 3: Comparison of different methods for performing ℓ_∞^T proximal projection. The `rand-pivot` is the random pivoting method of (Bruckner, 1984) and `rand-pivot-col` is the same applied with the nodes collapsed. The `k-best heap` is the method described in Algorithm 4.

that taking the tree-structure into account is beneficial. Moreover, the log-linear language model with ℓ_2^T penalty performs similar to interpolated Kneser-Ney. The ℓ_∞^T-norm outperforms all other models at order 5, but taking the structure into account does not prevent a degradation of the performance at higher orders, unlike ℓ_2^T. This means that a single regularization for all model orders is still inappropriate.

To investigate this further, we adjust the penalties by choosing an exponential decrease of weights varying as α^m for a feature at depth m in the suffix tree. Parameter α was tuned on a smaller validation set. The best performing values for these weighted models $w\ell_2^2$, $w\ell_1$, $w\ell_2^T$ and $w\ell_\infty^T$ are 0.5, 0.7, 1.1 and 0.85 respectively. The weighting scheme further appropriates the regularization at various levels to suit the problem's structure. Perplexity plots for weighted models are shown in Figure 2(b). While $w\ell_1$ improves at larger depths, it fails to compare to others showing that the problem does not admit sparse solutions. Weighted ℓ_2^2 improves considerably and performs comparably to the unweighted tree-structured norms. However, the introduction of weighted features prevents us from using the suffix tree representation, making these models inefficient in terms of memory. Weighted ℓ_∞^T is corrected for overfitting at larger depths and $w\ell_2^T$ gains more than others. Optimal values for α are fractional for all norms except $w\ell_2^T$-norm showing that the unweighted model ℓ_2^T-norm was over-penalizing features at larger depths, while that of others were under-penalizing them. Interestingly, perplexity improves up to about 9-grams with $w\ell_2^T$ penalty for the data set we considered, indicating that there is more to gain from longer dependencies in natural language sentences than what is currently believed.

Figure 2(c) compares model complexity measured by the number of parameters for weighted models using structured penalties. The ℓ_2^T penalty is applied on trie-structured vectors, which grows roughly at a linear rate with increasing model order. This is similar to Kneser-Ney. However, the number of parameters for the $w\ell_\infty^T$ penalty grows logarithmically with the model order. This is due to the fact that it operates on the suffix tree-structured vectors instead of the suffix trie-structured vectors. These results are valid for, both, weighted and unweighted penalties.

Next, we compare the average time taken per iteration for different implementations of the ℓ_∞^T proximal step. Figure 3(a) shows this time against increasing depth of the language model order for random pivoting method with and without the collapsing of parameters at different constant value non-branching paths. The trend in this plot resembles that of the number of parameters in Figure 2(c). This shows that the complexity of the full proximal step is sublinear when accounting for the suffix tree data structure. Figure 3(b) plots time per iteration random pivoting and k-best heap against the varying size of training sequence. The two algorithms are operating directly on the suffix tree. It can be observed that the heap-based method are superior with

increasing size of training data.

7 Conclusion

In this paper, we proposed several log-linear language models. We showed that with an efficient data structure and structurally appropriate convex regularization schemes, they were able to outperform standard Kneser-Ney smoothing. We also developed a proximal projection algorithm for the tree-structured ℓ_∞^T-norm suitable for large trees.

Further, we showed that these models can be trained online, that they accurately learn the m-gram weights and that they are able to better take advantage of long contexts. The time required to run the optimization is still a concern. It takes 7583 minutes on a standard desktop computer for one pass of the of the complete AP-news dataset with 13 million words which is little more than time reported for (Mnih and Hinton, 2007). The most time consuming part is computing the normalization factor for the log-loss. A hierarchical model in the flavour of (Mnih and Hinton, 2008) should lead to significant improvements to this end. Currently, the computational bottleneck is due to the normalization factor in (1) as it appears in every gradient step computation. Significant savings would be obtained by computing it as described in (Wu and Khundanpur, 2000).

Acknowledgements

The authors would like to thank anonymous reviewers for their comments. This work was partially supported by the CIFRE grant 1178/2010 from the French ANRT.

References

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, pages 1–106.

A. Beck and M. Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2(1):183–202.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

P. Bruckner. 1984. An $o(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, 3:163–166.

L. Burget, P. Matejka, P. Schwarz, O. Glembek, and J.H. Cernocky. 2007. Analysis of feature extraction and channel compensation in a GMM speaker recognition system. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):1979–1986, September.

Y-W. Chang and M. Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proc. Conf. Empirical Methods for Natural Language Processing*, pages 26–37.

S. F. Chen and R. Rosenfeld. 2000. A survey of smoothing techniques for maximum entropy models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. *An Introduction to Algorithms*. MIT Press.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. 2008. Efficient projections onto the ℓ_1-ball for learning in high dimensions. *Proc. 25th Int. Conf. Machine Learning*.

R. Giegerich and S. Kurtz. 1997. From ukkonen to McCreight and weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*.

J. Goodman. 2001. A bit of progress in language modelling. *Computer Speech and Language*, pages 403–434, October.

J. Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. North American Chapter of the Association of Computational Linguistics*.

C. Hu, J.T. Kwok, and W. Pan. 2009. Accelerated gradient methods for stochastic optimization and online learning. *Advances in Neural Information Processing Systems*.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. 2011. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334.

C. R. Kennington, M. Kay, and A. Friedrich. 2012. Sufx trees as language models. *Language Resources and Evaluation Conference*.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, volume 1.

A. F. T. Martins, N. A. Smith, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2011. Structured sparsity in structured prediction. In *Proc. Conf. Empirical Methods for Natural Language Processing*, pages 1500–1511.

P. McCullagh and J. Nelder. 1989. *Generalized linear models*. Chapman and Hall. 2nd edition.

A. Mnih and G. Hinton. 2007. Three new graphical models for statistical language modelling. *Proc. 24th Int. Conference on Machine Learning*.

A. Mnih and G. Hinton. 2008. A scalable hierarchical distributed language model. *Advances in Neural Information Processing Systems*.

Y. Nesterov. 2007. Gradient methods for minimizing composite objective function. *CORE Discussion Paper*.

B. Roark, M. Saraclar, M. Collins, and M. Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. *Proc. Association for Computation Linguistics*.

A. Stolcke. 2002. Srilm- an extensible language modeling toolkit. *Proc. Int. Conf. Spoken Language Processing*, 2:901–904.

E. Ukkonen. 1995. Online construction of suffix trees. *Algorithmica*.

S. Vargas, P. Castells, and D. Vallet. 2012. Explicit relevance models in intent-oriented information retrieval diversification. In *Proc. 35th Int. ACM SIGIR Conf. Research and development in information retrieval*, SIGIR '12, pages 75–84. ACM.

F. Wood, C. Archambeau, J. Gasthaus, J. Lancelot, and Y.-W. Teh. 2009. A stochastic memoizer for sequence data. In *Proc. 26th Intl. Conf. on Machine Learning*.

F. Wood, J. Gasthaus, C. Archambeau, L. James, and Y. W. Teh. 2011. The sequence memoizer. In *Communications of the ACM*, volume 54, pages 91–98.

J. Wu and S. Khundanpur. 2000. Efficient training methods for maximum entropy language modeling. *Proc. 6th Inter. Conf. Spoken Language Technologies*, pages 114–117.

P. Zhao, G. Rocha, and B. Yu. 2009. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497.

Interactive Machine Translation using Hierarchical Translation Models

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedí, Francisco Casacuberta
D. de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera s/n, 46021 Valencia (Spain)
{jegonzalez,dortiz,jbenedi,fcn}@dsic.upv.es

Abstract

Current automatic machine translation systems are not able to generate error-free translations and human intervention is often required to correct their output. Alternatively, an interactive framework that integrates the human knowledge into the translation process has been presented in previous works. Here, we describe a new interactive machine translation approach that is able to work with phrase-based and hierarchical translation models, and integrates error-correction all in a unified statistical framework. In our experiments, our approach outperforms previous interactive translation systems, and achieves estimated effort reductions of as much as 48% relative over a traditional post-edition system.

1 Introduction

Research in the field of *machine translation* (MT) aims to develop computer systems which are able to translate between languages automatically, without human intervention. However, the quality of the translations produced by any automatic MT system still remain below than that of human translation. Typical solutions to reach human-level quality require a subsequent manual *post-editing* process. Such decoupled post-edition solution is rather inefficient and tedious for the human translator. Moreover, it prevents the MT system from taking advantage of the knowledge of the human translator and, reciprocal, the human translator cannot take advantage of the adapting ability of MT technology.

An alternative way to take advantage of the existing MT technology is to use them in *collaboration* with human translators within a *computer-assisted translation* (CAT) or *interactive* framework (Isabelle and Church, 1998). The *TransType* and *TransType2* projects (Foster et al., 1998; Langlais and Lapalme, 2002; Barrachina et al., 2009) entailed an interesting focus shift in CAT technology by aiming interaction directly at the production of the target text. These research projects proposed to embed an MT system within an interactive translation environment. This way, the human translator can ensure a high-quality output while the MT system ensures a significant gain of productivity. Particularly interesting is the *interactive machine translation* (IMT) approach proposed in (Barrachina et al., 2009). In this scenario, a statistical MT (SMT) system uses the source sentence and a previously validated part (prefix[1]) of its translation to propose a suitable continuation. Then the user finds and corrects the next system error, thereby providing a longer prefix which the system uses to suggests a new, hopefully better continuation. The reported results showed that IMT can save a significant amount of human effort.

Barrachina et al,. (2009) provide a thorough description of the IMT approach and describe algorithms for its practical implementation. Nevertheless, we identify two basic problems for which we think there is room for improvement. The first problem arises when the system cannot generate the prefix validated by the user. To solve this problem, the authors simply provide an ad-hoc heuristic error-correction technique. The second problem is how the system deals with word reordering. Particularly, the models used by the system were either mono-

[1]We use the terms prefix and suffix to denote any sub-string at the beginning and end respectively of a string of characters (including spaces and punctuation). These terms do not imply any morphological significance as they usually do in linguistics.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 244–254,
Seattle, Washington, USA, 18-21 October 2013.

tonic by nature or non-monotonic but heuristically defined (not estimated from training data).

We work on the foundations of Barrachina et al., (2009) and provide formal solutions to these two challenges. On the one hand, we adopt the statistical formalization of the IMT framework described in (Ortiz-Martínez, 2011), which includes a stochastic error-correction model in its formalization to address prefix coverage problems. Moreover, we refine this formalization proposing an alternative error-correction formalization for the IMT framework (Section 2). Additionally, we also propose a specific error-correction model based on a statistical interpretation of the Levenshtein distance (Levenshtein, 1966). These formalizations provide a unified statistical framework for the IMT model in comparison to the ad-hoc heuristic error-correction methods previously used.

In order to address the problem of properly deal with reordering in IMT, we introduce the use of hierarchical MT models (Chiang, 2005; Zollmann and Venugopal, 2006). These methods provide a natural approach to handle long range dependencies and allow the incorporation of reordering information into a consistent statistical framework. Here, we also describe how state-of-the-art hierarchical MT models can be extended to handle IMT (Sections 3 and 4).

We evaluate the proposed IMT approach on two different translation task. The comparative results against the IMT approach described by Barrachina et al., (2009) and a conventional post-edition approach show that our IMT formalization for hierarchical SMT models indeed outperform other approaches (Sections 5 and 6). Moreover, it leads to large reductions in the human effort required to generate error-free translations.

2 Statistical Framework

2.1 Statistical Machine Translation

Assuming that we are given a sentence $\mathbf{s}$ in a source language, the *translation* problem can be stated as finding its translation $\mathbf{t}$ in a target language of maximum probability (Brown et al., 1993):

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \Pr(\mathbf{t} \mid \mathbf{s}) \quad (1)$$

$$= \arg\max_{\mathbf{t}} \Pr(\mathbf{t}) \cdot \Pr(\mathbf{s} \mid \mathbf{t}) \quad (2)$$

source ($\mathbf{s}$): Para ver la lista de recursos
desired translation ($\hat{\mathbf{t}}$): To view a listing of resources

IT-0	$\mathbf{p}$	
	$\mathbf{t}_s$	*To view the resources list*
IT-1	$\mathbf{p}$	To view
	k	a
	$\mathbf{t}_s$	*list of resources*
IT-2	$\mathbf{p}$	To view a list
	k	i
	$\mathbf{t}_s$	*ng resources*
IT-3	$\mathbf{p}$	To view a listing
	k	o
	$\mathbf{t}_s$	*f resources*
END	$\mathbf{p}$	To view a listing of resources

Figure 1: IMT session to translate a Spanish sentence into English. The desired translation is the translation the human user wants to obtain. At IT-0, the system suggests a translation ($\mathbf{t}_s$). At IT-1, the user moves the mouse to accept the first eight characters "To view " and presses the a key (k), then the system suggests completing the sentence with "*list of resources*" (a new $\mathbf{t}_s$). Iterations 2 and 3 are similar. In the final iteration, the user accepts the current translation.

The terms in the latter equation are the *language model* probability $\Pr(\mathbf{t})$ that represents the well-formedness of $\mathbf{t}$ (*n-gram* models are usually adopted), and the *(inverted) translation model* $\Pr(\mathbf{s} \mid \mathbf{t})$ that represents the relationship between the source sentence and its translation.

In practice all of these models (and possibly others) are often combined into a *log-linear model* for $\Pr(\mathbf{t} \mid \mathbf{s})$ (Och and Ney, 2002):

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \left\{ \sum_{n=1}^{N} \lambda_n \cdot \log(f_n(\mathbf{t}, \mathbf{s})) \right\} \quad (3)$$

where $f_n(\mathbf{t}, \mathbf{s})$ can be any model that represents an important feature for the translation, N is the number of models (or features), and λ_n are the weights of the log-linear combination.

2.2 Statistical Interactive Machine Translation

Unfortunately, current MT technology is still far from perfect. This implies that, in order to achieve good translations, manual post-editing is needed. An alternative to this decoupled approach (first MT, then manual correction) is given by the IMT

paradigm (Barrachina et al., 2009). Under this paradigm, translation is considered as an iterative left-to-right process where the human and the computer collaborate to generate the final translation.

Figure 1 shows an example of the IMT approach. There, a source Spanish sentence $\mathbf{s}$ ="Para ver la lista de recursos" is to be translated into a target English sentence $\hat{\mathbf{t}}$. Initially, with no user feedback, the system suggests a complete translation $\mathbf{t}_s$ ="To view the resources list". From this translation, the user marks a prefix $\mathbf{p}$ ="To view" as correct and begins to type the rest of the target sentence. Depending on the system or the user's preferences, the user might type the full next word, or only some letters of it (in our example, the user types the single next character "a"). Then, the MT system suggests a new suffix $\mathbf{t}_s$ ="list of resources" that completes the validated prefix and the input the user has just typed ($\mathbf{p}$ ="To view a"). The interaction continues with a new prefix validation followed, if necessary, by new input from the user, and so on, until the user considers the translation to be complete and satisfactory.

The crucial step of the process is the production of the suffix. Again decision theory tells us to maximize the probability of the suffix given the available information. Formally, the best suffix of a given length will be:

$$\hat{\mathbf{t}}_s = \arg\max_{\mathbf{t}_s} \Pr(\mathbf{t}_s \mid \mathbf{s}, \mathbf{p}) \tag{4}$$

which can be straightforwardly rewritten as:

$$\hat{\mathbf{t}}_s = \arg\max_{\mathbf{t}_s} \Pr(\mathbf{p}, \mathbf{t}_s \mid \mathbf{s}) \tag{5}$$

$$= \arg\max_{\mathbf{t}_s} \Pr(\mathbf{p}, \mathbf{t}_s) \cdot \Pr(\mathbf{s} \mid \mathbf{p}, \mathbf{t}_s) \tag{6}$$

Note that, since $\mathbf{p}\,\mathbf{t}_s = \mathbf{t}$, this equation is very similar to Equation (2). The main difference is that now the search process is restricted to those target sentences $\mathbf{t}$ that contains $\mathbf{p}$ as prefix. This implies that we can use the same MT models (including the log-linear approach) if the search procedures are adequately modified (Och et al., 2003). Finally, it should be noted that the statistical models are usually defined at word level, while the IMT process described in this section works at character level. To deal with this problem, during the search process it is necessary to verify the compatibility between $\mathbf{t}$ and $\mathbf{p}$ at character level.

2.3 IMT with Stochastic Error-Correction

A common problem in IMT arises when the user sets a prefix which cannot be explained by the statistical models. To solve this problem, IMT systems typically include ad-hoc error-correction techniques to guarantee that the suffixes can be generated (Barrachina et al., 2009). As an alternative to this heuristic approach, Ortiz-Martínez (2011) proposed a formalization of the IMT framework that does include stochastic error-correction models in its statistical formalization. The starting point of this alternative IMT formalization accounts for the problem of finding the translation $\mathbf{t}$ that, at the same time, better explains the source sentence $\mathbf{s}$ and the prefix given by the user $\mathbf{p}$:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \Pr(\mathbf{t} \mid \mathbf{s}, \mathbf{p}) \tag{7}$$

$$= \arg\max_{\mathbf{t}} \Pr(\mathbf{t}) \cdot \Pr(\mathbf{s}, \mathbf{p} \mid \mathbf{t}) \tag{8}$$

The following naïve Bayes' assumption is now made: the source sentence $\mathbf{s}$ and the user prefix $\mathbf{p}$ are statistically independent variables given the translation $\mathbf{t}$, obtaining:

$$\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \Pr(\mathbf{t}) \cdot \Pr(\mathbf{s} \mid \mathbf{t}) \cdot \Pr(\mathbf{p} \mid \mathbf{t}) \tag{9}$$

where $\Pr(\mathbf{t})$ can be approximated with a language model, $\Pr(\mathbf{s} \mid \mathbf{t})$ can be approximated with a translation model, and $\Pr(\mathbf{p} \mid \mathbf{t})$ can be approximated by an error correction model that measures the compatibility between the user-defined prefix $\mathbf{p}$ and the hypothesized translation $\mathbf{t}$.

Note that the translation result, $\hat{\mathbf{t}}$, given by Equation (9) may not contain $\mathbf{p}$ as prefix because every translation is compatible with $\mathbf{p}$ with a certain probability. Thus, despite being close, Equation (9) is not equivalent to the IMT formalization in Equation (6).

To solve this problem, we define an alignment, $\mathbf{a}$, between the user-defined prefix $\mathbf{p}$ and the hypothesized translation $\mathbf{t}$, so that the unaligned words of $\mathbf{t}$, in an appropriate order, constitute the suffix searched in IMT. This allows us to rewrite the error correction probability as follows:

$$\Pr(\mathbf{p} \mid \mathbf{t}) = \sum_{\mathbf{a}} \Pr(\mathbf{p}, \mathbf{a} \mid \mathbf{t}) \tag{10}$$

To simplify things, we assume that $\mathbf{p}$ is monotonically aligned to $\mathbf{t}$, leaving the potential word-reordering to the language and translation models.

Under this assumption, $\mathbf{a}$ determines an alignment for $\mathbf{t}$, such that $\mathbf{t} = \mathbf{t}_p\mathbf{t}_s$, where $\mathbf{t}_p$ is fully-aligned to $\mathbf{p}$ and $\mathbf{t}_s$ remains unaligned. Taking all these things into consideration, and following a maximum approximation, we finally arrive to the expression:

$$(\hat{\mathbf{t}}, \hat{\mathbf{a}}) = \arg\max_{\mathbf{t},\mathbf{a}} \Pr(\mathbf{t}) \cdot \Pr(\mathbf{s} \mid \mathbf{t}) \cdot \Pr(\mathbf{p}, \mathbf{a} \mid \mathbf{t}) \quad (11)$$

where the suffix required in IMT is obtained as the portion of $\hat{\mathbf{t}}$ that is not aligned with the user prefix.

In practice, we combine the models in Equation (11) in a log-linear fashion as it is typically done in SMT (see Equation (3)).

2.4 Alternative Formalization for IMT with Stochastic Error-Correction

Alternatively to Equation (11), we can operate from Equation (9) and reach a different formalization for IMT with error-correction. We can re-write the first and last terms of Equation (9) as:

$$\Pr(\mathbf{t}) \cdot \Pr(\mathbf{p} \mid \mathbf{t}) = \Pr(\mathbf{p}) \cdot \Pr(\mathbf{t} \mid \mathbf{p}) \quad (12)$$

As in the previous section, we introduce an alignment variable, $\mathbf{a}$, between $\mathbf{t}$ and $\mathbf{p}$, giving:

$$\Pr(\mathbf{t} \mid \mathbf{p}) = \sum_{\mathbf{a}} \Pr(\mathbf{t}, \mathbf{a} \mid \mathbf{p}) \quad (13)$$

$$= \sum_{\mathbf{a}} \Pr(\mathbf{a} \mid \mathbf{p}) \cdot \Pr(\mathbf{t} \mid \mathbf{p}, \mathbf{a}) \quad (14)$$

If we consider monotonic alignments, $\mathbf{a}$ defines again an alignment between a prefix of the system translation ($\mathbf{t}_p$) and the user prefix, producing the suffix required in IMT ($\mathbf{t}_s$) as the unaligned part. Thus, we can re-write $\Pr(\mathbf{t} \mid \mathbf{p}, \mathbf{a})$ as:

$$\Pr(\mathbf{t} \mid \mathbf{p}, \mathbf{a}) = \Pr(\mathbf{t}_p, \mathbf{t}_s \mid \mathbf{p}, \mathbf{a}) \quad (15)$$

$$\approx \Pr(\mathbf{t}_p \mid \mathbf{p}, \mathbf{a}) \cdot \Pr(\mathbf{t}_s \mid \mathbf{p}, \mathbf{a}) \quad (16)$$

where Equation (16) has been obtained following a naïve Bayes' decomposition.

Combining equations (12), (14), and (16) into Equation (9), and following a maximum approximation for the summation of the alignment variable $\mathbf{a}$, we arrive to the following expression:

$$(\hat{\mathbf{t}}, \hat{\mathbf{a}}) = \arg\max_{\mathbf{t},\mathbf{a}} \Pr(\mathbf{s}|\mathbf{t}) \cdot \Pr(\mathbf{t}_p|\mathbf{p}, \mathbf{a}) \cdot \Pr(\mathbf{t}_s|\mathbf{p}, \mathbf{a}) \quad (17)$$

where $\Pr(\mathbf{p})$ and $\Pr(\mathbf{a}|\mathbf{p})$ have been dropped down because the former does not participate in the maximization and the latter is assumed uniform.

The terms in this equation can be interpreted similarly as those in Equation (9): $\Pr(\mathbf{s} \mid \mathbf{t})$ is the translation model, $\Pr(\mathbf{t}_p \mid \mathbf{p}, \mathbf{a})$ is the error-correction probability that measures the compatibility between the prefix $\mathbf{t}_p$ of the hypothesized translation and the user-defined prefix $\mathbf{p}$, and $\Pr(\mathbf{t}_s \mid \mathbf{p}, \mathbf{a})$ is the language model for the corresponding suffix $\mathbf{t}_s$ conditioned by the user-defined prefix. Again, in the experiments we combine the different models in a log-linear fashion.

The main difference between the two alternative IMT formalization (Equations (11) and (17)) is that in the latter the suffix to be returned is conditioned by the user-validated prefix $\mathbf{p}$. Thus, in the following we will refer to Equation (11) as *independent suffix formalization* while we will denote Equation (17) by *conditioned suffix formalization*.

3 Statistical Models

We now present the statistical models used to estimate the probability distributions described in the previous section. Section 3.1 describes the error-correction model, while Section 3.2 describes the models for the conditional translation probability.

3.1 Statistical Error-Correction Model

Following the vast majority of IMT systems described in the literature, we implement an error-correction model based on the concept of edit distance (Levenshtein, 1966). Typically, IMT systems use non-probabilistic error correction models. The first stochastic error correction model for IMT was proposed in (Ortiz-Martínez, 2011) and it is based on probabilistic finite state machines. Here, we propose a simpler approach which can be seen as a particular case of the previous one. Specifically, the proposed approach models the edit distance as a Bernoulli process where each character of the candidate string has a probability p_e of being erroneous. Under this interpretation, the number of characters that need to be edited E in a sentence of length n is a random variable that follows a binomial distribution, $E \sim \mathrm{B}(n, p_e)$, with parameters n and p_e. The probability of performing exactly k edits in a

sentence of n characters is given by the following probability mass function:

$$f(k; n, p_e) = \frac{n!}{k!(n-k)!} p_e^k (1-p_e)^{n-k} \quad (18)$$

Note that this error-correction model penalizes equally all edit operations. Alternatively, we can model the distance with a multinomial distribution and assign different probabilities to different types of edit operations. Nevertheless, we adhere to the binomial approximation due to its simplicity.

Finally, we compute the error-correction probability between two strings from the total number of edits required to transform the candidate translation into the reference translation. Specifically, we define the error-correction distribution in Equation (11) as:

$$\Pr(\mathbf{p}, \mathbf{a} \mid \mathbf{t}) \approx \frac{|\mathbf{p}|!}{k!(|\mathbf{p}|-k)!} p_e^k (1-p_e)^{|\mathbf{p}|-k} \quad (19)$$

where $k = \mathrm{Lev}(\mathbf{p}, \mathbf{t_a})$ is the character-level Levenshtein distance between the user defined prefix $\mathbf{p}$ and the prefix $\mathbf{t_a}$ of the hypothesized translation $\mathbf{t}$ defined by alignment $\mathbf{a}$. The error-correction probability $\Pr(\mathbf{t}_p \mid \mathbf{p}, \mathbf{a})$ in Equation (17) is computed analogously.

The probability of edition p_e is the single free parameter of this formulation. We will use a separate development corpus to find an adequate value for it.

3.2 Statistical Machine Translation Models

Next sections briefly describe the statistical translation models used to estimate the conditional probability distribution $\Pr(\mathbf{s} \mid \mathbf{t})$. A detailed description of each model can be found in the provided citations.

3.2.1 Phrase-Based Translation Models

Phrase-based translation models (Koehn et al., 2003) are an instance of the noisy-channel approach in Equation (2). The translation of a source sentence $\mathbf{s}$ is obtained through a generative process composed of three steps: first, the $\mathbf{s}$ is divided into K segments (phrases), next, each source phrase, $\tilde{\mathbf{s}}$, is translated into a target phrase $\tilde{\mathbf{t}}$, and finally the target phrases are reordered to compose the final translation.

The usual phrase-based implementation of the translation probability takes a log-linear form:

$$\Pr(\mathbf{s} \mid \mathbf{t}) \approx \lambda_1 \cdot |\mathbf{t}| + \lambda_2 \cdot K + \sum_{k=1}^{K} \left[\lambda_3 \cdot \log(P(\tilde{\mathbf{s}}_k \mid \tilde{\mathbf{t}}_k)) + \lambda_4 \cdot \mathrm{d}(j)\right] \quad (20)$$

where $P(\tilde{\mathbf{s}} \mid \tilde{\mathbf{t}})$ is the translation probability between source phrase $\tilde{\mathbf{s}}$ and target phrase $\tilde{\mathbf{t}}$, and $\mathrm{d}(j)$ is a function (distortion model) that returns the score of translating the k-th source phrase given that it is separated j words from the $(k{-}1)$-th phrase. Weights λ_1 and λ_2 play a special role since they are used to control the number of words and the number of phrases of the target sentence to be generated, respectively.

3.2.2 Hierarchical Translation Models

Phrase-based models have shown a very strong performance when translating between languages that have similar word orders. However, they are not able to adequately capture the complex relationships that exist between the word orders of languages of different families such as English and Chinese. Hierarchical translation models provide a solution to this challenge by allowing gaps in the phrases (Chiang, 2005):

$$\textsf{yu } X_1 \textsf{ you } X_2 \rightarrow \textsf{have } X_2 \textsf{ with } X_1$$

where subscripts denote placeholders for subphrases. Since these rules generalize over possible phrases, they act as discontinuous phrase pairs and may also act as phrase-reordering rules. Hence, they are not only considerably more powerful than conventional phrase pairs, but they also integrate reordering information into a consistent framework.

These hierarchical phrase pairs are formalized as rewrite rules of a synchronous context-free grammar (CFG) (Aho and Ullman, 1969):

$$X \rightarrow < \boldsymbol{\gamma}, \boldsymbol{\alpha}, \sim > \quad (21)$$

where X is a non-terminal, $\boldsymbol{\gamma}$ and $\boldsymbol{\alpha}$ are both strings of terminals (words) and non-terminals , and $\sim$ is a one-to-one correspondence between non-terminal occurrences in $\boldsymbol{\gamma}$ and $\boldsymbol{\alpha}$. Given the example above, $\boldsymbol{\gamma} \equiv$"yu X_1 you X_2", $\boldsymbol{\alpha} \equiv$"have X_2 with X_1", and $\sim$ is indicated by the subscript numbers.

Additionally, two *glue rules* are also defined:

$$S \rightarrow < S_1 X_2 \, , \, S_1 X_2 > \qquad S \rightarrow < X_1 \, , \, X_1 >$$

These give the model the option to build only partial translations using hierarchical phrases, and then combine them serially as in a phrase-based model.

The typical implementation of the hierarchical translation model also takes the form of a log-linear model. Let $\mathbf{s}_{\boldsymbol{\delta}}$ and $\mathbf{t}_{\boldsymbol{\delta}}$ be the source and target strings generated by a derivation $\boldsymbol{\delta}$ of the grammar. Then, the conditional translation probability is given by:

$$\Pr(\mathbf{s}_{\boldsymbol{\delta}} \mid \mathbf{t}_{\boldsymbol{\delta}}) \approx \lambda_1 \cdot |\mathbf{t}_{\boldsymbol{\delta}}| + \lambda_2 \cdot |\boldsymbol{\delta}| + \lambda_3 \cdot \#_{\mathrm{g}}(\boldsymbol{\delta}) + \sum_{r \in \boldsymbol{\delta}} [\lambda_4 \cdot w(r)] \quad (22)$$

where $|\boldsymbol{\delta}|$ denotes the total number of rules used in $\boldsymbol{\delta}$, $\#_{\mathrm{g}}(\boldsymbol{\delta})$ returns the number of applications of the glue rules, $r \in \boldsymbol{\delta}$ are the rules in $\boldsymbol{\delta}$, and $w(r)$ is the weight of rule r. Weights λ_1 and λ_2 have a similar interpretation as for phrase-based models, they respectively give some control over the total number of words and rules that conform the translation. Additionally, λ_3 controls the model's preference for hierarchical phrases over serial combination of phrases. Note that no distortion model is included in the previous equation. Here, reordering is defined at rule level by the one-to-one non-terminal correspondence. In other words, reordering is a property inherent to each rule and it is the individual score of each rule what defines, at each step of the derivation, the importance of reordering.

It should be noted that the IMT formalizations presented in Section 2 can be applied to other hierarchical or syntax-based SMT models such as those described in (Zollmann and Venugopal, 2006; Shen et al., 2010).

4 Search

In offline MT, the generation of the best translation for a given source sentence is carried out by incrementally generating the target sentence[2]. This process fits nicely into a *dynamic programming* (DP) (Bellman, 1957) framework, as hypotheses which are indistinguishable by the models can be recombined. Since the DP search space grows exponentially with the size of the input, standard DP search is prohibitive, and search algorithms usually resort to a beam-search heuristic (Jelinek, 1997).

[2]Phrase-based systems follow a left-to-right generation order while hierarchical systems rely on a CYK-like order.

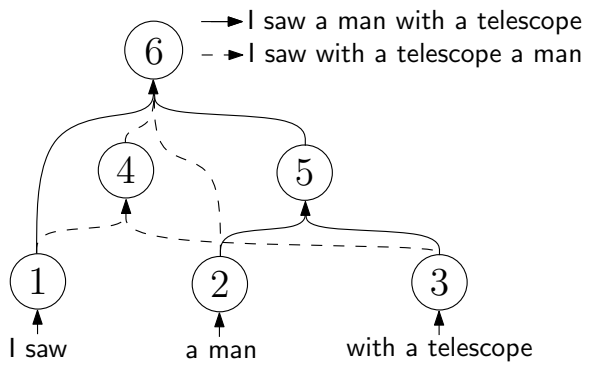


Figure 2: Example of a hypergraph encoding two different translations (one solid and one dotted) for the Spanish sentence "Vi a un hombre con un telescopio".

Due to the demanding temporal constraints inherent to any interactive environment, performing a full search each time the user validates a new prefix is unfeasible. The usual approach is to rely on a certain representation of the search space that includes the most probable translations of the source sentence. The computational cost of this approach is much lower, as the whole search for the translation must be carried out only once, and the generated representation can be reused for further completion requests.

Next, we introduce *hypergraphs*, the formalism chosen to represent the search space of both phrase-based and hierarchical systems (Section 4.1). Then, we describe the algorithms implemented to search for suffix completions in them (Section 4.2).

4.1 Hypergraphs

A hypergraph is a generalization of the concept of graph where the edges (now called hyperedges) may connect several nodes (hypernodes) at the same time. Formally, a hypergraph is a weighted acyclic graph represented by a pair $< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ is a set of hypernodes and $\mathcal{E}$ is a set of hyperedges. Each hyperedge $e \in \mathcal{E}$ connects a head hypernode and a set of tail hypernodes. The number of tail nodes is called the *arity* of the hyperedge and the arity of a hypergraph is the maximum arity of its hyperedges.

We can use hypergraphs to represent the derivations for a given CFG. Each hypernode represents a partial translation generated during the decoding process. Each ingoing hyperedge represents the rule with which the corresponding non-terminal was substituted. Moreover, hypergraphs can represent a whole set of possible translations. An example is

shown in Figure 2. Two alternative translations are constructed from the leave nodes (1, 2 and 3) up to the root node (6) of the hypergraph. Additionally, hypernodes and hyperedges may be shared among different derivations if they represent the same information. Thus, we can achieve a compact representation of the translation space that allows us to derive efficient search algorithms.

Note that *word-graphs* (Ueffing et al., 2002), which are used to represent the search space for phrase-based models, are a special case of hypergraphs in which the maximum arity is one. Thus, hypergraphs allow us to represent both phrase-based and hierarchical systems in a unified framework.

4.2 Suffix Search on Hypergraphs

Now, we describe a unified search process to obtain the suffix $\mathbf{t}_s$ that completes a prefix $\mathbf{p}$ given by the user according to the two IMT formulations (Equation (11) and Equation (17)) described in Section 2.

Given an hypergraph, certain hypernodes define a possible solution to the maximization defined in the two IMT formulations. Specifically, only those hypernodes that generate a prefix of a potential translation are to be taken into account[3]. The probability of the solution defined by each hypernode has two components, namely the probability of the SMT model (given by the language and translation models) and the probability of the error-correction model. On the one hand, the SMT model probability is given by the translation of maximum probability through the hypernode. On the other hand, the error-correction probability is computed between $\mathbf{p}$ and the partial translation of maximum probability actually covered by the hypernode. Among all the solutions defined by the hypernodes, we finally select that of maximum probability.

Once the best-scoring hypernode is identified, the rest of the translation not covered by it is returned as the suffix completion required in IMT.

5 Experimental Framework

The models and search procedure introduced in the previous sections were assessed through a series of IMT experiments with different corpora. These corpora, the experimental methodology, and the evaluation measures are presented in this section.

[3]For example, in Figure 2 the hypernodes that generate prefixes are those labeled with numbers 1 ("I saw"), 4 ("I saw with a telescope) and 6 ("I saw a man with a telescope" and "I saw with a telescope a man").

	EU (Es/En)		
	Train	Development	Test
Sentences	214K	400	800
Token	5.9M / 5.2M	12K / 10K	23K / 20K
Vocabulary	97K / 84K	3K / 3K	5K / 4K
	TED (Zh/En)		
	Train	Development	Test
Sentences	107K	934	1664
Token	2M / 2M	22K / 20K	33K / 32K
Vocabulary	42K / 52K	4K / 3K	4K / 4K

Table 1: Main figures of the processed EU and TED corpora. K and M stand for thousands and millions of elements respectively.

5.1 EU and TED corpora

We tested the proposed methods in two different translation tasks each one involving a different language pair: Spanish-to-English (Es–En) for the *EU* (Bulletin of the European Union) task, and Chinese-to-English (Zh–En) for the *TED* (TED[4] talks) task.

The EU corpora were extracted from the Bulletin of the European Union, which exists in all official languages of the European Union and is publicly available on the Internet. Particularly, the chosen Es–En corpus was part of the evaluation of the TransType2 project (Barrachina et al., 2009). The TED talks is a collection of recordings of public speeches covering a variety of topics, and for which high quality transcriptions and translations into several languages are available. The Zh–En corpus used in the experiments was part of the MT track in the 2011 evaluation campaign of the workshop on spoken language translation (Federico et al., 2011). Specifically, we used the `dev2010` partition for development and the `test2010` partition for test.

We process the Spanish and English parts of the EU corpus to separate words and punctuation marks keeping sentences truecase. Regarding the TED corpus, we tokenized and lowercased the English part (Chinese has no case information), and split Chinese sentences into words with the Stanford word

[4]`www.ted.com`

segmenter (Tseng et al., 2005). Table 1 shows the main figures of the processed EU and TED corpora.

5.2 Model Estimation and User Simulation

We used the standard configuration of the `Moses` toolkit (Koehn et al., 2007) to estimate one phrase-based and one hierarchical model for each corpus; log-linear weights were optimized by minimum error-rate training (Och, 2003) with the development partitions. Then, the optimized models were used to generate the word-graphs and hypergraphs with the translations of the development and test partitions.

A direct evaluation of the proposed IMT procedures involving human users would have been slow and expensive. Thus, following previous works in the literature (Barrachina et al., 2009; González-Rubio et al., 2010), we used the references in the corpora to simulate the translations that a human user would want to obtain. Each time the system suggested a new translation, it was compared to the reference and the *longest common prefix* (LCP) was obtained. Then, the first non-matching character was replaced by the corresponding character in the reference and a new system suggestion was produced. This process is iterated until a full match with the reference was obtained.

Finally, we used this user simulation to optimize the value for the probability of edition p_e in the error-correction model (Section 3.1), and for the log-linear weights in the proposed IMT formulations. In this case, these values were chosen so that they minimize the estimated user effort required to interactively translate the development partitions.

5.3 Evaluation Measures

Different measures have been adopted to evaluate the proposed IMT approach. On the one hand, different IMT systems can be compared according to the effort needed by a human user to generate the desired translations. This effort is usually estimated as the number of actions performed by the user while interacting with the system. In the user simulation described above these actions are: looking for the next error and *moving the mouse pointer* to that position (LCP computation), and correcting errors with some *key strokes*. Hence, we implement the following IMT effort measure (Barrachina et al., 2009):

Key-stroke and mouse-action ratio (KSMR): number of key strokes plus mouse movements performed by the user, divided by the total number of characters in the reference.

On the other hand, we also want to compare the proposed IMT approach against a conventional CAT approach without interactivity, such as a decoupled post-edition system. For such systems, character-level user effort is usually measured by the *Character Error Rate* (CER). However, it is clear that CER is at a disadvantage due to the auto-completion approach of IMT. To perform a fairer comparison between post-edition and IMT, we implement a post-editing system with autocompletion. Here, when the user enters a character to correct some incorrect word, the system automatically completes the word with the most probable word in the task vocabulary. To evaluate the effort of a user using such a system, we implement the following measure proposed in (Romero et al., 2010):

Post-editing key stroke ratio (PKSR): using a post-edition system with word-autocompleting, number of user key strokes divided by the total number of reference characters.

The counterpart of PKSR in an IMT scenario is (Barrachina et al., 2009):

Key-stroke ratio (KSR): number of key strokes, divided by the number of reference characters.

PKSR and KSR are fairly comparable and the relative difference between them gives us a good estimate of the reduction in human effort that can be achieved by using IMT instead of a conventional post-edition system.

We also evaluate the quality of the automatic translations generated by the MT models with the widespread BLEU score (Papineni et al., 2002).

Finally, we provide both confidence intervals for the results and statistical significance of the observed differences in performance. Confidence intervals were computed by pair-wise re-sampling as in (Zhang and Vogel, 2004) while statistical significance was computed using the Tukey's HSD (honest significance difference) test (Hsu, 1996).

	EU		TED	
	WG	HG	WG	HG
1-best BLEU [%]	45.0	45.1	11.0	11.2
1000-best Avg. BLEU [%]	43.6	44.2	10.2	11.0

Table 2: BLEU score of the word-graphs (WG) and hypergraphs (HG) used to implement the IMT procedures.

IMT	EU		TED	
Setup	PB	HT	PB	HT
ISF	27.4±.5	26.5±.5*	53.0±.4	52.3±.4*
CSF	26.6±.5*	**25.1±.5**★	52.2±.4*	**50.8±.4**★

Table 3: IMT results (KSMR [%]) for the EU and TED tasks using the independent suffix formalization (ISF) and the conditioned suffix formalization (CSF). PB stands for phrase-based model and HT stands for hierarchical translation model. For each task, the best result is displayed boldface, an asterisk * denotes a statistically significant better result (99% confidence) with respect to ISF with PB, and a star ★ denotes a statistically significant difference with respect to all the other systems.

6 Results

We start by reporting conventional MT quality results to test if the generated word-graphs and hypergraphs encode translations of similar quality. Table 2 displays the quality (BLEU (Papineni et al., 2002)) of the automatic translations generated for the test partitions. The lower 1-best BLEU results obtained for TED show that this is a much more difficult task than EU. Additionally, the similar average BLEU results obtained for the 1000-best translations indicate that word-graphs and hypergraphs encode translations of similar quality. Thus, the IMT systems that use these word-graphs and hypergraphs can be compared in a fair way.

Then, we evaluated different setups of the proposed IMT approach. Table 3 displays the IMT results obtained for the EU and TED tasks. We report KSMR (as a percentage) for the independent suffix formalization (ISF) and the conditioned suffix formalization (CSF) using both phase-based (PB) and hierarchical (HT) translation models. The KSMR result of ISF using a phrase-based model can be considered our baseline since this setup is comparable to that used in (Barrachina et al., 2009). Results for HT consistently outperformed the corresponding results for PB. Similarly, results for CSF were consistently better than those for ISF. More specifically, no statistically significant difference were found between ISF with HT and CSF with PB but both statistically outperformed the baseline (ISF with PB). Finally, CSF with HT statistically outperformed the other three configurations reducing KSMR by ~ 2.2 points with respect to the baseline. We hypothesize that the better results of HT can be explained by its more efficient representation of word reordering. Regarding the CSF, its better results are due to the better suffixes that can be obtained by taking into account the actual prefix validated by the user.

EU		TED	
PE	**IMT**	**PE**	**IMT**
PKSR [%]	KSR [%]	PKSR [%]	KSR [%]
27.1	**14.1 (48%)**	40.8	**29.7 (27.2%)**

Table 4: Estimation of the effort required to translate the test partition of the EU and TED tasks using post-editing with word-completion (PE) and IMT under the independent suffix formalization (IMT). We used hierarchical MT in both approaches. In parenthesis we display the estimated effort reduction of IMT with respect to PE.

Finally, we compared the estimated human effort required to translate the test partitions of the EU and TED corpora with the best IMT configuration (independent suffix formalization with hierarchical translation model) and a conventional post-editing (PE) CAT system with word-completion. That is, when the user corrects a character, the PE system automatically proposes a different word that begins with the given word prefix but, obviously, the rest of the sentence is not changed. According to the results, the estimated human effort to generate the error-free translations was significantly reduced with respect to using the conventional PE approach. IMT can save about 48% of the overall eastimated effort for the EU task and about 27% for the TED task.

7 Summary and Future Work

We have proposed a new IMT approach that uses hierarchical SMT models as its underlying translation technology. This approach is based on a statistical formalization previously described in the literature that includes stochastic error correction. Additionally, we have proposed a refined formalization that improves the quality of the IMT suffixes by taking

into account the prefix validated by the user. Moreover, since word-graphs constitute a particular case of hypergraphs, we are able to manage both phrase-based and hierarchical translation models in a unified IMT framework.

Simulated results on two different translation tasks showed that hierarchical translation models outperform phrase-based models in our IMT framework. Additionally, the proposed alternative IMT formalization also allows to improve the results of the IMT formalization previously described in the literature. Finally, the proposed IMT system with hierarchical SMT models largely reduces the estimated user effort required to generate correct translations in comparison to that of a conventional post-edition system. We look forward to corroborating these result in test with human translators.

There are many ways to build on the work described here. In the near future, we plan to explore the following research directions:

- Alternative IMT scenarios where the user is not bounded to correct translation errors in a left-to-right fashion. In such scenarios, the user will be allowed to correct errors at any position in the translation while the IMT system will be required to derive translations compatible with these isolated corrections.

- Adaptive translation engines that take advantage of the user's corrections to improve its statistical models. As the translator works and corrects the proposed translations, the translation engine will be able to make better predictions. One of the first works on this topic was proposed in (Nepveu et al., 2004). More recently, Ortiz-Martínez et al. (2010) described a set of techniques to obtain an incrementally updateable IMT system, solving technical problems encountered in previous works.

- More sophisticated measures to estimate the human effort. Specifically, measures that estimate the cognitive load involve in reading, understanding and detecting an error in a translation (Foster et al., 2002), in contrast KSMR simply considers a constant cost. This will lead to a more accurate estimation of the improvements that may be expected by a human user.

Acknowledgments

Work supported by the European Union 7^{th} Framework Program (FP7/2007-2013) under the CasMaCat project (grans agreement n$^{\text{o}}$ 287576), by Spanish MICINN under grant TIN2012-31723, and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/014).

References

Alfred V. Aho and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and Systems Science*, 3(1):37–56, February.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35:3–28, March.

Richard Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19:263–311.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270.

M. Federico, L. Bentivogli, M. Paul, and S. Stüker. 2011. Overview of the iwslt 2011 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 11–20.

George Foster, Pierre Isabelle, and Pierre Plamondon. 1998. Target-text mediated interactive machine translation. *Machine Translation*, 12(1/2):175–194, January.

George Foster, Philippe Langlais, and Guy Lapalme. 2002. User-friendly text prediction for translators. In *Proceedings of the 2002 conference on Empirical methods in natural language processing - Volume 10*, pages 148–155.

Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2010. Balancing user effort and translation error in interactive machine translation via confidence measures. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 173–177.

Jason Hsu. 1996. *Multiple Comparisons: Theory and Methods*. Chapman and Hall/CRC.

Pierre Isabelle and Ken Church. 1998. *Special issue on: New tools for human translators*, volume 12. Kluwer Academic Publishers, January.

Frederick Jelinek. 1997. *Statistical methods for speech recognition*. MIT Press.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Association for Computational Linguistics, demonstration session*, June.

Philippe Langlais and Guy Lapalme. 2002. TransType: development-evaluation cycles to boost translator's productivity. *Machine Translation*, 17(2):77–98, September.

Vladimir Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February.

Laurent Nepveu, Guy Lapalme, Philippe Langlais, and George Foster. 2004. Adaptive language and translation models for interactive machine translation. In *Proceedings of the conference on Empirical Methods on Natural Language Processing*, pages 190–197.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

Franz Josef Och, Richard Zens, and Hermann Ney. 2003. Efficient search for interactive statistical machine translation. In *Proceedings of the European chapter of the Association for Computational Linguistics*, pages 387–393.

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 160–167. Association for Computational Linguistics.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 546–554.

Daniel Ortiz-Martínez. 2011. *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. Ph.D. thesis, Universitat Politècnica de València. Advisors: Ismael García Varea and Francisco Casacuberta.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics.

Veronica Romero, Alejandro H. Toselli, and Enrique Vidal. 2010. Character-level interaction in computer-assisted transcription of text images. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition*, pages 539–544.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4):649–671, December.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.

Nicola Ueffing, Franz J. Och, and Hermann Ney. 2002. Generation of word graphs in statistical machine translation. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 156–163.

Ying Zhang and Stephan Vogel. 2004. Measuring confidence intervals for the machine translation evaluation metrics. In *Proceedings of The international Conference on Theoretical and Methodological Issues in Machine Translation*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141.

Max-Margin Synchronous Grammar Induction for Machine Translation

Xinyan Xiao and **Deyi Xiong***
School of Computer Science and Technology
Soochow University
Suzhou 215006, China
xyxiao.cn@gmail.com, dyxiong@suda.edu.cn

Abstract

Traditional synchronous grammar induction estimates parameters by maximizing likelihood, which only has a loose relation to translation quality. Alternatively, we propose a max-margin estimation approach to discriminatively inducing synchronous grammars for machine translation, which directly optimizes translation quality measured by BLEU. In the max-margin estimation of parameters, we only need to calculate Viterbi translations. This further facilitates the incorporation of various non-local features that are defined on the target side. We test the effectiveness of our max-margin estimation framework on a competitive hierarchical phrase-based system. Experiments show that our max-margin method significantly outperforms the traditional two-step pipeline for synchronous rule extraction by 1.3 BLEU points and is also better than previous max-likelihood estimation method.

1 Introduction

Synchronous grammar induction, which refers to the process of learning translation rules from bilingual corpus, still remains an open problem in statistical machine translation (SMT). Although state-of-the-art SMT systems model the translation process based on synchronous grammars (including bilingual phrases), most of them still learn translation rules via a pipeline with word-based heuristics (Koehn et al., 2003). This pipeline first builds word alignments using heuristic combination strategies, then heuristically extracts rules that are consistent with word alignments. Such heuristic pipeline is not elegant theoretically. It brings an undesirable gap that separates modeling and learning in an SMT system.

Therefore, researchers have proposed alternative approaches to learning synchronous grammars directly from sentence pairs without word alignments, via generative models (Marcu and Wong, 2002; Cherry and Lin, 2007; Zhang et al., 2008; DeNero et al., 2008; Blunsom et al., 2009; Cohn and Blunsom, 2009; Neubig et al., 2011; Levenberg et al., 2012) or discriminative models (Xiao et al., 2012). Theoretically, these approaches describe how sentence pairs are generated by applying sequences of synchronous rules in an elegant way. However, they learn synchronous grammars by maximizing likelihood,[1] which only has a loose relation to translation quality (He and Deng, 2012). Moreover, generative models are normally hard to be extended to incorporate useful features, and the discriminative synchronous grammar induction model proposed by Xiao et al. (2012) only incorporates *local* features defined on parse trees of the source language. *Non-local* features, which encode information from parse trees of the target language, have never been exploited before due to the computational complexity of normalization in max-likelihood estimation.

Consequently, we would like to learn synchronous grammars in a discriminative way that can directly maximize the end-to-end translation quality measured by BLEU (Papineni et al., 2002), and is also able to incorporate non-local features from target parse trees.

We thus propose a max-margin estimation method

*Corresponding author

[1] More precisely, the discriminative model by Xiao et al. (2012) maximizes conditional likelihood.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 255–264,
Seattle, Washington, USA, 18-21 October 2013.

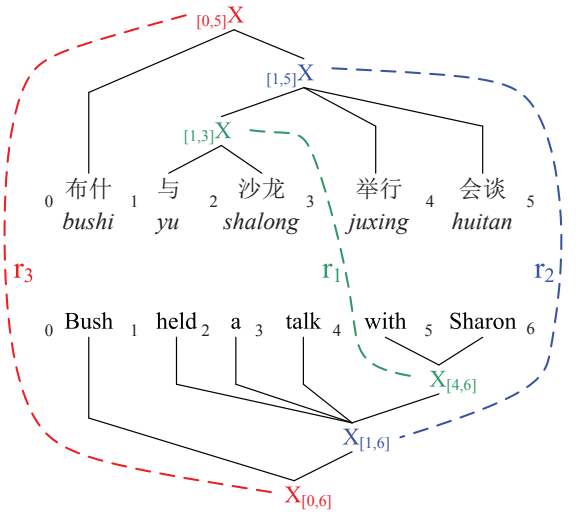
to discriminatively induce synchronous grammar directly from sentence pairs without word alignments. We try to maximize the margin between a reference translation and a candidate translation with translation errors that are measured by BLEU. The more serious the translation errors, the larger the margin. In this way, our max-margin method is able to learn synchronous grammars according to their translation performance. We further incorporate various *non-local* features defined on target parse trees. We efficiently calculate the non-local feature values of a translation over its exponential derivation space using the inside-outside algorithm. Because our max-margin estimation optimizes feature weights only by the feature values of Viterbi and reference translations, we are able to efficiently perform optimization even with non-local features.

We apply the proposed max-margin estimation method to learn synchronous grammars for a hierarchical phrase-based translation system (Chiang, 2007) which typically produces state-of-the-art performance. With non-local features defined on target parse trees, our max-margin method significantly outperforms the baseline that uses synchronous rules learned from the traditional pipeline by 1.3 BLEU points on large-scale Chinese-English bilingual training data.

The remainder of this paper is organized as follows. Section 2 presents the discriminative synchronous grammar induction model with the non-local features. In Section 3, we elaborate our max-margin estimation method which is able to directly optimize BLEU, and discuss how we induce grammar rules. Local and non-local features are described in Section 4. Finally, in Section 5, we verify the effectiveness of our method through experiments by comparing it against both the traditional pipeline and max-likelihood estimation method.

2 Discriminative Model with Non-local Features

Let $\mathcal{S}$ denotes the set of all strings in a source language. Given a source sentence $\mathbf{s} \in \mathcal{S}$, $\mathcal{T}(s)$ denotes all candidate translations in the target language that can be generated by a synchronous grammar $\mathbf{G}$. A translation $t \in \mathcal{T}(s)$ is generated by a sequence of translation steps $(r_1, ..., r_n)$, where we apply a synchronous rule $r \in \mathbf{G}$ in one step. We refer to such a sequence of translation steps as a **derivation** (See Figure 1) and denote it as $\mathbf{d} \in \mathcal{D}(\mathbf{s})$, where $\mathcal{D}(\mathbf{s})$ represents the derivation space of a source sentence. Given an input source sentence $\mathbf{s}$, we output a pair $\langle \mathbf{t}, \mathbf{d} \rangle$ in SMT. Thus, we study the triple $\langle \mathbf{s}, \mathbf{t}, \mathbf{d} \rangle$ in SMT.



r_1: $\langle$ *yu shalong* $\Rightarrow$ with Sharon $\rangle$
r_2: $\langle$ X *juxing huitan* $\Rightarrow$ *held a talk* X $\rangle$
r_3: $\langle$ *bushi* X $\Rightarrow$ Bush X $\rangle$

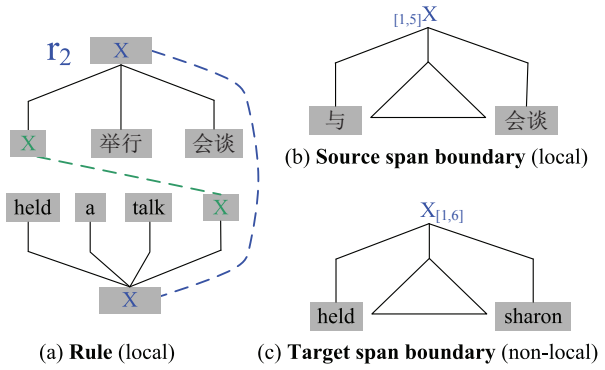
Figure 1: A derivation of a sentence pair represented by a synchronous tree. The above and below part are the parses in the source language side and the target language side respectively. Left subscript of a node X denotes the source span, while right subscript denotes the target span. A dashed line denotes an alignment from a source span to a target span. The annotation for a dashed line corresponds to the rewriting rule used in the corresponding step of the derivation.

In our discriminative model, we calculate the value of a triple $\langle \mathbf{s}, \mathbf{t}, \mathbf{d} \rangle$ according to the following **scoring function**:

$$f(\mathbf{s}, \mathbf{t}, \mathbf{d}) = \theta^T \Phi(\mathbf{s}, \mathbf{t}, \mathbf{d}) \quad (1)$$

where $\theta \in \mathbf{\Theta}$ is a feature weight vector, and Φ is the **feature function**.

There are exponential outputs in SMT. Therefore it is necessary to factorize the feature function in order to perform efficient calculation over the SMT output space using dynamic programming. We decompose the feature function of a triple $\langle \mathbf{s}, \mathbf{t}, \mathbf{d} \rangle$ into



(a) **Rule** (local) (b) **Source span boundary** (local) (c) **Target span boundary** (non-local)

Figure 2: Example features for the derivation in Figure 1. Shaded nodes denote information encoded in the feature.

$\mathbf{s}, \mathbf{S}, \mathcal{S}$	$\mathbf{s}$ is a sentence in a source language; $\mathbf{S}$ means source training sentences; $\mathcal{S}$ denotes all the possible sentences;
$\mathbf{t}, \mathbf{T}, \mathcal{T}$	symbols for the target language that similar to $\mathbf{s}$, $\mathbf{S}$, $\mathcal{S}$;
$\mathbf{d}, \mathcal{D}$	derivation and derivation space;
$\mathcal{D}(\mathbf{s})$	space of derivations for a source sentence;
$\mathcal{D}(\mathbf{s}, \mathbf{t})$	space of derivations for a source sentence with its translation;
$\mathcal{H}(\mathbf{s})$	hypergraph that represents $\mathcal{D}(\mathbf{s})$;
$\mathcal{H}(\mathbf{s}, \mathbf{t})$	hypergraph that represents $\mathcal{D}(\mathbf{s}, \mathbf{t})$;

Table 1: Notations in this paper. We give an abstract of related notations for clarity.

a sum of values of each synchronous rule in the derivation $\mathbf{d}$.

$$\Phi(\mathbf{s}, \mathbf{t}, \mathbf{d}) = \sum_{r \in \mathbf{d}} \underbrace{\phi(r, \mathbf{s})}_{\textbf{local}} + \sum_{r \in \mathbf{d}} \underbrace{\phi(r, \mathbf{s}, \mathbf{t})}_{\textbf{non-local}} \quad (2)$$

Our feature functions include both local and non-local features. A feature is a **local** feature if and only if it can be factored among the translation steps in a derivation. In other words, the value of a local feature for $\langle \mathbf{s}, \mathbf{t}, \mathbf{d} \rangle$ can be calculated as a sum of local scores in each translation step, and the calculation of each local score only requires to look at the rule used in corresponding step and the input sentence. Otherwise, the feature is a **non-local** feature. Our discriminative model allows to incorporate non-local features that are defined on target translations.

For example, a rule feature in Figure 2(a), which indicates the application of a specific rule in a derivation, is a local feature. A source span boundary feature in Figure 2(b) that is defined on the source parse tree is also a local feature. However, a target span boundary feature in Figure 2(c), which assesses the target parse structure, is a non-local feature. According to Figure 1, the span is parsed in step r_2, but it also depends on the translation boundary word "held" generated in previous step r_1. We will describe the details of both local and non-local features that we use in Section 4.

Non-local features enable us to model the target parse structure in a derivation. However, it is computationally expensive to calculate the expected values of non-local features over $\mathcal{D}(\mathbf{s})$, as non-local features require to record states of target boundary words and result in an extremely large number of states during dynamic programming. Fortunately, when integrating out derivations over the derivation space $\mathcal{D}(\mathbf{s}, \mathbf{t})$ of a source sentence and its translation, we can efficiently calculate the non-local features. Because all derivations in $\mathcal{D}(\mathbf{s}, \mathbf{t})$ share the same translation, there is no need to maintain states for target boundary words. We will discuss this computational problem in details in Section 3.3. In the proposed max-margin estimation described in next section, we only need to integrate out derivation for a Viterbi translation and a reference translation when updating feature weights. Therefore, the defined non-local features allow us to not only explore useful knowledge on the target parse trees, but also compute them efficiently over $\mathcal{D}(\mathbf{s}, \mathbf{t})$ during max-margin estimation.

3 Max-Margin Estimation

In this section, we describe how we use a parallel training corpus $\{\mathbf{S}, \mathbf{T}\} = \{(\mathbf{s}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{N}$ to estimate feature weights θ, which contain parameters of the induced synchronous grammars and the defined non-local features.

We choose the parameters that maximize the translation quality measured by BLEU using the *max-margin estimation* (Taskar et al., 2004). Margin refers to the difference of the model score between a reference translation $\mathbf{t}^{(i)}$ and a candidate translation $\mathbf{t}$. We hope that the worse the translation quality of $\mathbf{t}$, the larger the margin between $\mathbf{t}$ and $\mathbf{t}^{(i)}$. In this way, we penalize larger translation

errors more severely than smaller ones. This intuition is expressed by the following equation.

$$\begin{aligned} &\min \ \frac{1}{2}\|\theta\|^2 \\ &s.t. \ f(\mathbf{s}^{(i)}, \mathbf{t}^{(i)}) - f(\mathbf{s}^{(i)}, \mathbf{t}) \geq cost(\mathbf{t}^{(i)}, \mathbf{t}) \\ &\quad \forall \mathbf{t} \in \mathcal{T}(\mathbf{s}^{(i)}) \end{aligned} \tag{3}$$

Here, $f(\mathbf{s}, \mathbf{t})$ is the feature function of a translation, and **cost function** $cost(\mathbf{t}^{(i)}, \mathbf{t})$ measures the translation errors of a candidate translation $\mathbf{t}$ comparing with a reference translation $\mathbf{t}^{(i)}$. We define the cost function via the widely-used translation evaluation metric BLEU. We use the smoothed sentence level BLEU-4 (Lin and Och, 2004) here:

$$cost(\mathbf{t}^{(i)}, \mathbf{t}) = 1 - \text{BLEU-4}(\mathbf{t}^{(i)}, \mathbf{t}) \tag{4}$$

In Section 3.1, we will discuss how we use the scoring function $f(\mathbf{s}, \mathbf{t}, \mathbf{d})$ to calculate $f(\mathbf{s}, \mathbf{t})$. Then in Section 3.2, we recast the equation (3) as an unconstrained empirical loss minimization problem, and describe the learning algorithm for optimizing θ and inducing $\mathbf{G}$. Finally, we give the details of inference for the learning algorithm in Section 3.3.

3.1 Integrate Out Derivation by Averaging

Although we only model the triple $\langle \mathbf{s}, \mathbf{t}, \mathbf{d} \rangle$ in the equation (1), it's necessary to calculate the scoring function $f(\mathbf{s}, \mathbf{t})$ of a translation by integrating out the variable of derivation as derivation is not observed in the training data.

We use an averaging computation over all possible derivations of a translation $\mathcal{D}(\mathbf{s}, \mathbf{t})$. We call this an average derivation based estimation:

$$f(\mathbf{s}, \mathbf{t}) = \frac{1}{|\mathcal{D}(\mathbf{s}, \mathbf{t})|} \sum_{\mathbf{d} \in \mathcal{D}(\mathbf{s}, \mathbf{t})} f(\mathbf{s}, \mathbf{t}, \mathbf{d}) \tag{5}$$

The "average derivation" can be considered as the geometric central point in the space $\mathcal{D}(\mathbf{s}, \mathbf{t})$.

Another possible way to deal with the latent derivation is max-derivation, which uses the max-operator over $\mathcal{D}(\mathbf{s}, \mathbf{t})$. The max derivation method sets $f(\mathbf{s}, \mathbf{t})$ as $\max_{\mathbf{d} \in \mathcal{D}(\mathbf{s}, \mathbf{t})} f(\mathbf{s}, \mathbf{t}, \mathbf{d})$. It is often adopted in traditional SMT systems. Nevertheless, we instead use average-derivation for two reasons.[2] First, as a translation has an exponential number of derivations, finding the max derivation of a reference translation for learning is nontrivial (Chiang et al., 2009). Second, the max derivation estimation will result in a low rule coverage, as rules in a max derivation only covers a small fraction of rules in the $\mathcal{D}(\mathbf{s}, \mathbf{t})$. Because rule coverage is important in synchronous grammar induction, we would like to explore the entire derivation space using the average operator.

[2] Imagine that $\mathcal{H}(\mathbf{s}, \mathbf{t})$ in the Algorithm 1 is replaced by a maximum derivation in $\mathcal{H}(\mathbf{s}, \mathbf{t})$.

3.2 Learning Algorithm

We reformulate the equation (3) as an unconstrained empirical loss minimization problem as follows:

$$\min \frac{\lambda}{2}\|\theta\|^2 + \frac{1}{N}\sum_{n=1}^{N} L(\mathbf{s}^{(i)}, \mathbf{t}^{(i)}, \theta) \tag{6}$$

Where λ denotes the regularization strength for L2-norm. The loss function of a sentence pair $L(\mathbf{s}^{(i)}, \mathbf{t}^{(i)}, \theta)$ is a convex hinge loss function denoted by:

$$\begin{aligned} &\max\{0, -f(\mathbf{s}^{(i)}, \mathbf{t}^{(i)}) \\ &\quad + \max_{\mathbf{t} \in \mathcal{T}(\mathbf{s}^{(i)})} \Big(f(\mathbf{s}^{(i)}, \mathbf{t}) + cost(\mathbf{t}^{(i)}, \mathbf{t}) \Big) \} \end{aligned} \tag{7}$$

According to the second max-operator in the hinge loss function, the optimization towards BLEU is expressed by **cost-augmented** inference. Cost-augmented inference finds a translation that has a maximum model score augmented with cost.

$$\hat{\mathbf{t}} = \max_{\mathbf{t} \in \mathcal{T}(\mathbf{s}^{(i)})} \Big(f(\mathbf{s}^{(i)}, \mathbf{t}) + cost(\mathbf{t}^{(i)}, \mathbf{t}) \Big) \tag{8}$$

We applied the Pegasos algorithm for the optimization of equation (6) (Shalev-Shwartz et al., 2007). This is an online algorithm, which alternates between stochastic gradient descent steps and projection steps. When the loss function is non-zero, it updates weights according to the sub-gradient of the hinge loss function. Using the average scoring function in the equation (5), the sub-gradient of hinge loss function for a sentence pair is the difference of average feature values between a Viterbi translation

Algorithm 1 UPDATE($\mathbf{s}, \mathbf{t}, \theta, \mathbf{G}$) ▷ One step in online algorithm. $\mathbf{s}$, $\mathbf{t}$ are short for $\mathbf{s}^{(i)}$, $\mathbf{t}^{(i)}$ here

1: $\mathcal{H}(\mathbf{s}, \mathbf{t}) \leftarrow$ BIPARSE($\mathbf{s}, \mathbf{t}, \theta$) ▷ Build hypergraph of reference translation
2: $\mathbf{G} \leftarrow \mathbf{G} + \mathcal{H}(\mathbf{s}, \mathbf{t})$ ▷ Discover rules from $\mathcal{H}(\mathbf{s}, \mathbf{t})$
3: $\hat{\mathbf{t}}, \hat{\mathbf{d}} \leftarrow \arg\max_{\langle \mathbf{t}', \mathbf{d}' \rangle \in \mathcal{D}(\mathbf{s})} f(\mathbf{s}, \mathbf{t}', \mathbf{d}') + cost(\mathbf{t}, \mathbf{t}')$ ▷ Find Viterbi translation
4: $\mathcal{H}(\mathbf{s}, \hat{\mathbf{t}}) \leftarrow$ BIPARSE($\mathbf{s}, \hat{\mathbf{t}}, \theta$) ▷ Build hypergraph of Viterbi translation
5: **if** $f(\mathbf{s}, \mathbf{t}) < f(\mathbf{s}, \hat{\mathbf{t}}) + cost(\mathbf{t}, \hat{\mathbf{t}})$ **then**
6: $\quad \theta \leftarrow (1 - \eta\lambda)\theta + \eta \times \frac{\partial \mathbf{L}}{\partial \theta}(\mathcal{H}(\mathbf{s}, \mathbf{t}), \mathcal{H}(\mathbf{s}, \hat{\mathbf{t}}))$ ▷ Update θ by gradient $\frac{\partial \mathbf{L}}{\partial \theta}$ and learning rate η
7: $\quad \theta \leftarrow \min\{1, \frac{1/\sqrt{\lambda}}{\|\theta\|}\} \times \theta$ ▷ Projection by scaling
8: **return** $\mathbf{G}$, θ

and a reference translation.

$$\frac{\partial \mathbf{L}}{\partial \theta} = \frac{1}{|\mathcal{D}(\mathbf{s}^{(i)}, \mathbf{t}^{(i)})|} \sum_{\mathbf{d} \in \mathcal{D}(\mathbf{s}^{(i)}, \mathbf{t}^{(i)})} \Phi(\mathbf{s}^{(i)}, \mathbf{t}^{(i)}, \mathbf{d}) - \frac{1}{|\mathcal{D}(\mathbf{s}^{(i)}, \hat{\mathbf{t}})|} \sum_{\mathbf{d} \in \mathcal{D}(\mathbf{s}^{(i)}, \hat{\mathbf{t}})} \Phi(\mathbf{s}^{(i)}, \hat{\mathbf{t}}, \mathbf{d}) \quad (9)$$

Algorithm 1 shows the procedure of one step in the online optimization algorithm. The procedure discovers rules and updates weights in an online fashion. In the procedure, we first biparse the sentence pair to construct a synchronous hypergraph of a reference translation (line 1). In the biparsing algorithm, synchronous rules for constructing hyperedges are not required to be in $\mathbf{G}$, but can be any rules that follow the form defined in Chiang (2007). Thus, the biparsing algorithm can discover new rules that are not in $\mathbf{G}$. Then we collect the translation rules discovered in the hypergraph of the reference translation (line 2), which are rules indicated by hyperedges in the hypergraph. We then calculate the Viterbi translation according to the scoring function and cost function (see Section 3.3) (line 3), and build the synchronous hypergraph for the Viterbi translation (line 4). Finally, we update weights according to the Pegasos algorithm (line 5). The sub-gradient is calculated based on the hypergraph of Viterbi translation and reference translation.

In practice, in order to process the data in a parallel manner, we use a larger step size of 1000 for the learning algorithm. In each step of our online optimization algorithm, we first biparse 1000 reference sentence pairs in parallel. Then, we collect grammar rules from the generated reference hypergraphs. After that, we compute the gradients of 1000 sentence pairs in parallel, by calculating feature weights over reference hypergraphs and Viterbi hypergraphs. Finally, we update the feature weights using the sum of these gradients.

3.3 Inference

There are two parts that need to be calculated in the learning algorithm: finding a cost-augmented Viterbi translation according to the scoring function and cost function (Equation 8), and constructing synchronous hypergraphs for the Viterbi and reference translation so as to discover rules and calculate average feature values in Equation (9). Following the traditional decoding procedure, we resort to the cube-pruning based algorithm for approximation.

To find the Viterbi translation, we run the traditional translation decoding algorithm (Chiang, 2007) to get the best derivation. Then we use the translation yielded by the best derivation as the Viterbi translation. In order to obtain the BLEU score in the cost function, we need to calculate the ngram precision. It is calculated in a way similar to the calculation of the ngram language model. The computation of BLEU-4 requires to record 3 boundary words in both the left and right side during dynamic programming. Therefore, even when we use a language model whose order is less than 4, we still expands the states to record 3 boundary words so as to calculate the cost measured by BLEU.

We build synchronous hypergraphs using the cube-pruning based biparsing algorithm (Xiao et al., 2012). Algorithm 2 shows the procedure. Using a chart, the biparsing algorithm constructs k-best alignments for every source word (lines 1-5) and k-best hyperedges for every source span (lines 6-13) from the bottom up. Thus, a synchronous hypergraph is generated during the construction of the chart. More specifically, for a source span, it first creates cubes $\mathcal{L}$ for all source parses γ that are in-

Algorithm 2 BIPARSE($\mathbf{s}, \mathbf{t}, \theta$) ▷ (Xiao et al., 2012)

▷ Create k-best alignments for each source word
1: **for** $i \leftarrow 1, .., |\mathbf{s}|$ **do**
2: **for** $j \leftarrow 1, .., |\mathbf{t}|$ **do**
3: $L_j \leftarrow \{\varepsilon, t_j\}$ ▷ s_i aligns to t_j or not
4: $\mathcal{L} \leftarrow \langle L_1, ..., L_{|t|} \rangle$
5: $chart[s, i] \leftarrow$ KBEST($\mathcal{L}$,$\otimes$,θ)
▷ Create k-best hyperedges for each source span
6: $\mathcal{H} \leftarrow \emptyset$
7: **for** $h \leftarrow 1, .., |\mathbf{s}|$ **do** ▷ h is the size of span
8: **for all** i, j s.t. $j - i = h$ **do**
9: $\mathcal{L} \leftarrow \emptyset$
10: **for** γ inferable from $chart$ **do**
11: $\mathcal{L} \leftarrow \mathcal{L}$ + $\langle chart[\gamma_1], ..., chart[\gamma_{|\gamma|}] \rangle$
12: $chart[X, i, j] \leftarrow$ KBEST($\mathcal{L}$,$\otimes$,θ)
13: $\mathcal{H} \leftarrow \mathcal{H}$ + $chart[X, i, j]$ ▷ save hyperedges
14: **return** $\mathcal{H}$

ferable from the chart (lines 9-11). Here γ_i is a partial source parse that covers either a single source word or a span of source words. Then it uses the cube pruning algorithm to keep the top k derivations among all partial derivations that share the same source span $[i, j]$ (line 12). Notably, this biparsing algorithm does not require specific translation rules as input. Instead, it is able to discover new synchronous grammar rules when constructing a synchronous hypergraph: extracting each hyperedge in the hypergraph as a synchronous rule.

Based on the biparsing algorithm, we are able to construct the reference hypergraph $\mathcal{H}(\mathbf{s}^{(i)}, \mathbf{t}^{(i)})$ and Viterbi hypergraph $\mathcal{H}(\mathbf{s}^{(i)}, \hat{\mathbf{t}})$. By the reference hypergraph, we collect new synchronous translation rules and record them in the grammar $\mathbf{G}$. We also calculate the average feature values of hypergraphs using the inside-outside algorithm (Li et al., 2009), so as to compute the gradients.

4 Features

One advantage of the discriminative method is that it enables us to incorporate arbitrary features. As shown in Section 2, our model incorporates both local and non-local features.

4.1 Local Features

Rule features We associate each rule with an indicator feature. Each indicator feature counts the number of times that a rule appears in a derivation. In this way, we are able to learn a weight for every rule according to the entire structure of sentence.

Word association features Lexicalized features are widely used in traditional SMT systems. Here we adopt two lexical weights called noisy-or features (Zens and Ney, 2004). The noisy-or feature is estimated by word translation probabilities output by GIZA++. We set the initial weight of these two lexical scores with equivalent positive values. The lexical weights enable our system to score and rank the hyperedges at the beginning. Although word alignment features are used, we do not constrain the derivation space of a sentence pair by prefixed word alignment, and do not require any heuristic alignment combination strategy.

Length feature We integrate the length of target translation that is used in traditional SMT system as our feature.

Source span boundary features We use this kind of feature to assess the source parse tree in a derivation. Previous work (Xiong et al., 2010) has shown the importance of phrase boundary features for translation. Actually, this kind of feature is a good cue for deciding the boundary where a rule is to be learnt. Following Taskar et al. (2004), for a bispan $[i, j, k, l]$ in a derivation, we define the feature templates that indicates the boundaries of a span by its beginning and end words: $\{B : s_{i+1}; E : s_j; BE : s_{i+1}, s_j\}$.

Source span orientation features Orientation features are only used for those spans that are swapping. In Figure 1, the translation of source span $[1, 3]$ is swapping with that of span $[4, 5]$ by r_2, thus orientation feature for span $[1, 3]$ is activated. We also define three feature templates for a swapping span similar to the boundary features: $\{B : s_{i+1}; E : s_j; BE : s_{i+1}, s_j\}$. In practice, we add a prefix to the orientation features so as to distinguish these features from the boundary features.

4.2 Non-local Features

Target span boundary features We also want to assess the target tree structure in a derivation. We define these features in a way similar to source span boundary features. For a bispan $[i, j, k, l]$ in a derivation, we define the feature templates that indicates

System	Grammar Size	MT03	MT04	MT05	Avg.
Moses	302.5M	34.26	36.56	32.69	34.50
Baseline	77.8M	33.83	35.81	33.23	34.29
Max-margin	59.4M	34.62	37.14	34.00	35.25
+Sparse feature		35.48	37.31	34.07	35.62

Table 2: Experiment results. Baseline is an in-house implementation of hierarchical phrase based system. *Moses* denotes the implementation of hierarchical phrased-model in Moses (Koehn et al., 2007). $+Sparsefeature$ means that those sparse features used in the grammar induction are also used during decoding. The improvement of max-margin over Baseline is statistically significant ($p < 0.01$).

target span boundary as: $\{B : t_{k+1}; E : t_l; BE : t_{k+1}, t_l\}$.

Target span orientation features Similar target orientation features are used for a swapping span $[i, j, k, l]$ with feature templates $\{B : t_{k+1}; E : t_l; BE : t_{k+1}, t_l\}$.

Relative position features Following Blunsom and Cohn (Blunsom and Cohn, 2006), we integrate features indicating the closeness to the alignment matrix diagonal. For an aligned word pair with source position i and target position j, the value of this feature is $|\frac{i}{|\mathbf{s}|} - \frac{j}{|\mathbf{t}|}|$. As this feature depends on the length of the target sentence, it is a non-local feature.

Language model We also incorporate an ngram language model which is an important component in SMT. For efficiency, we use a 3-gram language model trained on the target side of our training data during the induction of synchronous grammars.

5 Experiment

In this section, we present our experiments on the NIST Chinese-to-English translation tasks. We first compare our max-margin based method with the traditional pipeline on a large bitext which contains 1.1 million sentences. We then present a detailed comparison on a smaller dataset, in order to analyze the effectiveness of max-margin estimation comparing with the max likelihood estimation (Xiao et al., 2012), and also the effectiveness of the non-local features that are defined on the target side.

5.1 Setup

The baseline system is the hierarchical phrase based system (Chiang, 2007). We used a bilingual corpus that contains 1.1M sentences (44.6 million words) of up to length 40 from the LDC data.[3] Our 5-gram language model was trained by SRILM toolkit (Stolcke, 2002). The monolingual training data includes the Xinhua section of the English Gigaword corpus and the English side of the entire LDC data (432 million words).

We used the NIST 2002 (MT02) as our development set, and the NIST 2003-2005 (MT03-05) as the test set. Case-insensitive NIST BLEU-4 (Papineni et al., 2002) is used to measure translation performance, and also the cost function in the max-margin estimation. Statistical significance in BLEU differences was tested by paired bootstrap re-sampling (Koehn, 2004). We used minimum error rate training (MERT) (Och, 2003) to optimize feature weights for the traditional log-linear model.

We used the same decoder as the baseline system in all estimation methods. Without special explanation, we used the same features as those in the traditional pipeline: forward and backward translation probabilities, forward and backward lexical weights, count of extracted rules, count of glue rules, length of translation, and language model. For the lexical weights we used the noisy-or in all configurations including the baseline system. For the discriminative grammar induction, rule translation probabilities were calculated using the expectations of rules in the synchronous hypergraphs of sentence pairs.

As our max-margin synchronous grammar induction is trained on the entire bitext, it is necessary to load all the rules into the memory during training. To control the size of rule table, we used Viterbi-

[3]Including LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005T06 and Hansards portion of LDC2004T08.

System	Feature Function	MT03	MT04	MT05	Avg.
Baseline	—	31.76	33.08	31.06	31.96
Max-likelihood	local	32.84	34.54	31.61	33.00
Max-margin	local	32.97	34.92	31.99	33.29
	local,non-local	33.27	34.83	32.32	33.47

Table 3: Comparison of Max-margin and Max-likelihood estimation on a smaller corpus. For max-margin method, we present two results according to the usages of non-local features. The max-margin with non-local features significantly outperforms the Baseline ($p < 0.01$) and also the max-likelihood estimation ($p < 0.05$).

pruning (Huang, 2008) when collecting rules as shown in line 2 of optimization procedure in Section 3.2. Furthermore, we aggressively discarded those large rules (The number of source symbols or the number of target symbols are more than two) that occur only in one sentence. Whenever the learning algorithm processes 50K sentences, we performed this discarding operation for large rules.

5.2 Result on Large Dataset

Table 2 shows the translation results. Our method induces 59.4 million synchronous rules, which are 76.3% of the grammar size of baseline. Note that Moses allows the boundary words of a phrase to be unaligned, while our baseline constraints the initial phrase to be tightly consistent with word alignment. Therefore, Moses extract a much larger rule table than that of our baseline.

With fewer translation rules, our method obtains an average improvement of +0.96 BLEU points on the three test sets over the Baseline. As the difference between the baseline and our max-margin synchronous grammar induction model only lies in the grammar, this result clearly denotes that our learnt grammar does outperform the grammar extracted by the traditional two-step pipeline.

We also incorporate the sparse features during decoding in a way similar to Xiao et al. (2012) and Dyer et al. (2011). In order to optimize these sparse features with the dense features by MERT, we group features of the same type into one coarse "summary feature", and get three such features including: rule, phrase-boundary and phrase orientation features. In this way, we rescale the weights of the three "summary features" with the 8 dense features by MERT. We achieve a further improvement of +0.37 BLEU points. Therefore, our training algorithm is able to learn the useful information encoded by the sparse features for translation.

5.3 Comparison of Estimation Objective and Non-Local Feature

We want to investigate whether the max-margin estimation is able to outperform the max-likelihood estimation method (Xiao et al., 2012). Therefore we carried out experiments to compare them directly. As the max-margin method is able to use non-local features, we compare two settings of features for the max-margin method. One uses only local features, the other uses both local and non-local features. Because the training procedure need to run on the entire corpus, which is time consuming, we therefore use a smaller corpus containing 50K sentences from the entire bitext for comparison.

Table 3 shows the results. When using only local features, the max-margin method consistently outperforms the max-likelihood method in all three test sets. This clearly shows the advantage of learning grammars by optimizing BLEU over likelihood.

When incorporating the non-local features into the max-margin method, we achieve further improvement against the max-margin method without non-local features. With non-local features, our max-margin estimation method outperforms the baseline by 1.5 BLEU points, and is better than the max-likelihood estimation by 0.5 BLEU points. Based on these results, we believe that non-local features, which encode information from target parse structures, are helpful for grammar induction. This further confirms the advance of the max-margin estimation, as it provides us a convenient way to use non-local features.

6 Related Work

As the synchronous grammar is the key component in SMT systems, researchers have proposed various methods to improve the quality of grammars. In addition to the generative and discriminative models introduced in Section 1, researchers also have made efforts on word alignment and grammar weight rescoring.

The first line is to modify word alignment by exploring information of syntactic structures (May and Knight, 2007; DeNero and Klein, 2010; Pauls et al., 2010; Burkett et al., 2010; Riesa et al., 2011). Such syntactic information is combined with word alignment via a discriminative framework. These methods prefer word alignments that are consistent with syntactic structure alignments. However, labeled word alignment data are required in order to learn the discriminative model.

Yet another line is to rescore the weights of translation rules. This line of work tries to improve the relative frequency estimation used in the traditional pipeline. They rescore the weights or probabilities of extracted rules. The rescoring is done by using the similar latent log-linear model as ours (Blunsom et al., 2008; Kääriäinen, 2009; He and Deng, 2012), or incorporating various features using labeled word aligned bilingual data (Huang and Xiang, 2010). However, in rescoring, translation rules are still extracted by the heuristic two-step pipeline. Therefore these previous work still suffers from the inelegance problem of the traditional pipeline.

Our work also relates to the discriminative training (Och, 2003; Watanabe et al., 2007; Chiang et al., 2009; Xiao et al., 2011; Gimpel and Smith, 2012) that has been widely used in SMT systems. Notably, these discriminative training methods are not used to learn grammar. Instead, they assume that grammar are extracted by the traditional two-step pipeline.

7 Conclusion

In this paper we have presented a max-margin estimation for discriminative synchronous grammar induction. By associating the margin with the translation quality, we directly learn translation rules that optimize the translation performance measured by BLEU. Max-margin estimation also provides us a convenient way to incorporate non-local features. Experiment results validate the effectiveness of optimizing parameters by BLEU, and the importance of incorporating non-local features defined on the target language. These results confirm the advantage of our max-margin estimation framework as it can both optimize BLEU and incorporate non-local features.

Feature engineering is very important for discriminative models. Researchers have proposed various types of features for machine translation, which are often estimated from word alignments. We would like to investigate whether further improvement can be achieved by incorporating such features, especially the context model (Shen et al., 2009) in the future. Because our proposed model is quite general, we are also interested in applying this method to induce linguistically motivated synchronous grammars for syntax-based SMT.

Acknowledgments

The first author was partially supported by 863 State Key Project (No. 2011AA01A207) and National Key Technology R&D Program (No. 2012BAH39B03). We are grateful to the anonymous reviewers for their insightful comments. We also thank Yi Lin for her invaluable feedback.

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Prob. ACL 2006*, July.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. ACL 2008*.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proc. ACL 2009*.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proc. NAACL 2010*.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *Proc. SSST 2007, NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*, April.

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

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

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *Proc. EMNLP 2009*.

John DeNero and Dan Klein. 2010. Discriminative modeling of extraction sets for machine translation. In *Proc. ACL 2010*.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proc. EMNLP 2008*.

Chris Dyer, Kevin Gimpel, Jonathan H. Clark, and Noah A. Smith. 2011. The cmu-ark german-english translation system. In *Proc. WMT 2011*.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proc. NAACL 2012*.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proc. ACL 2012*.

Fei Huang and Bing Xiang. 2010. Feature-rich discriminative phrase rescoring for smt. In *Proc. Coling 2010*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. ACL 2008*.

Matti Kääriäinen. 2009. Sinuhe – statistical machine translation using a globally trained conditional exponential family translation model. In *Proc. EMNLP 2009*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL 2007 (demonstration session)*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*.

Abby Levenberg, Chris Dyer, and Phil Blunsom. 2012. A bayesian model for learning scfgs with discontiguous rules. In *Proc. EMNLP 2012*. Association for Computational Linguistics, July.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. ACL 2009*.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Pro. Coling 2004*.

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

Jonathan May and Kevin Knight. 2007. Syntactic realignment models for machine translation. In *Proc. EMNLP 2007*.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proc. ACL 2011*.

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

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

Adam Pauls, Dan Klein, David Chiang, and Kevin Knight. 2010. Unsupervised syntactic alignment with inversion transduction grammars. In *Proc. NAACL 2010*.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proc. EMNLP 2011*.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proc. ICML 2007*.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proc. EMNLP 2009*.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit.

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proc. EMNLP 2004*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. EMNLP-CoNLL 2007*.

Xinyan Xiao, Yang Liu, Qun Liu, and Shouxun Lin. 2011. Fast generation of translation forest for large-scale smt discriminative training. In *Proc. EMNLP 2011*.

Xinyan Xiao, Deyi Xiong, Yang Liu, Qun Liu, and Shouxun Lin. 2012. Unsupervised discriminative induction of synchronous grammar for machine translation. In *Proc. Coling 2012*.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *Proc. NAACL2010*.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Prob. NAACL 2004*.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proc. ACL 2008*.

Error-Driven Analysis of Challenges in Coreference Resolution

Jonathan K. Kummerfeld and **Dan Klein**
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{jkk,klein}@cs.berkeley.edu

Abstract

Coreference resolution metrics quantify errors but do not analyze them. Here, we consider an automated method of categorizing errors in the output of a coreference system into intuitive underlying error types. Using this tool, we first compare the error distributions across a large set of systems, then analyze common errors across the top ten systems, empirically characterizing the major unsolved challenges of the coreference resolution task.

1 Introduction

Metrics produce measurements that concisely summarize performance on the full range of error types, and for coreference resolution there has been extensive work on developing effective metrics (Luo, 2005; Recasens and Hovy, 2011). However, it is also valuable to tease apart the errors to understand their relative importance.

Previous investigations of coreference errors have focused on quantifying the importance of subtasks such as named entity recognition and anaphoricity detection, typically by measuring accuracy improvements when partial gold annotations are provided (Stoyanov et al., 2009; Pradhan et al., 2011; Pradhan et al., 2012). For coreference resolution the drawback of this approach is that decisions are often interdependent, and so even partial gold information is extremely informative. Also, previous work only considered errors by counting links, which does not capture certain errors in a natural way, e.g. when a system incorrectly divides a large entity into two parts, each with multiple mentions. Recent work has considered some of these issues, but only with small scale manual analysis (Holen, 2013).

We present a new tool that automatically classifies errors in the standard output of any coreference resolution system. Our approach is to identify changes that convert the system output into the gold annotations, and map the steps in the conversion onto linguistically intuitive error types. Since our tool uses only system output, we are able to classify errors made by systems of any architecture, including both systems that use link-based inference and systems that use global inference methods.

Using our tool we perform two studies to understand similarities and differences between systems. First, we compare the error distributions on coreference resolution of all of the systems from the CoNLL 2011 shared task plus several publicly available systems. This comparison adds to the analysis from the shared task by illustrating the substantial variation in the types of errors different systems make. Second, we investigate the aggregate behavior of ten state-of-the-art systems, providing a detailed characterization of each error type. This investigation identifies key outstanding challenges and presents the impact that solving each of them would have in terms of changes in the standard coreference resolution metrics.

We find that the best systems are not best across all error types, that a large proportion of span errors are due to superficial parse differences, and that the biggest performance loss is on missed entities that contain a small number of mentions.

This work presents a comprehensive investigation of common errors in coreference resolution, identifying particular issues worth focusing on in future research. Our analysis tool is available at code.google.com/p/berkeley-coreference-analyser/.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 265–277,
Seattle, Washington, USA, 18-21 October 2013.

2 Background

Most coreference work focuses on accuracy improvements, as measured by metrics such as MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF (Luo, 2005), and BLANC (Recasens and Hovy, 2011). The only common forms of further analysis are results for anaphoricity detection and scores for each mention type (nominal, pronoun, proper). Two exceptions are: the detailed analysis of the Reconcile system by Stoyanov et al. (2009), and the multi-system comparisons in the CoNLL shared task reports (Pradhan et al., 2011, 2012).

A common approach to performance analysis is to calculate scores for nominals, pronouns and proper names separately, but this is a very coarse division (Ng and Cardie, 2002; Haghighi and Klein, 2009). More fine consideration of some subtasks does occur, for example, anaphoricity detection, which has been recognized as a key challenge in coreference resolution for decades and regularly has separate results reported (Paice and Husk, 1987; Sobha et al., 2011; Yuan et al., 2012; Björkelund and Farkas, 2012; Zhekova et al., 2012). Some work has also included anecdotal discussion of specific error types or manual classification of a small set of errors, but these approaches do not effectively quantify the relative impact of different errors (Chen and Ng, 2012; Martschat et al., 2012; Haghighi and Klein, 2009). In a recent paper, Holen (2013) presented a detailed manual analysis that considered a more comprehensive set of error types, but their focus was on exploring the shortcomings of current metrics, rather than understanding the behavior of current systems.

The detailed investigation presented by Stoyanov et al. (2009) is the closest to the work we present here. First, they measured accuracy improvements when their system was given gold annotations for three subtasks of coreference resolution: mention detection, named entity recognition, and anaphoricity detection. To isolate other types of errors they defined resolution classes, based on both the type of a mention, and properties of possible antecedents (for example, nominals that have a possible antecedent that is an exact string match). For each resolution class they measured performance while giving the system gold annotations for all other classes. While this approach is effective at characterizing variations between the nine classes they defined, it misses the cascade effect of errors that only occur when all mentions are being resolved at once.

President Clinton$_1$ is questioning the legitimacy of George W. Bush's election victory. Speaking last night to Democratic supporters in Chicago, **he** said Bush won the election only because Republicans stopped the vote-counting in Florida, and Mr. Clinton$_1$ praised Al Gore's campaign manager, Bill Daley, for the way he handled the election. "I$_2$ want to thank Bill Daley for his exemplary service as Secretary of Commerce. He was brilliant. I$_2$ think he did a brilliant job in leading Vice President Gore to victory myself$_2$."

Figure 1: Two coreference errors. Mentions are underlined and subscripts indicate entities. One error is a mention missing from the system output, *he*. The other is the division of references to Bill Clinton into two entities.

The only multi-system comparisons are the CoNLL task reports (Pradhan et al., 2011, 2012), which explored the impact of mention detection and anaphoricity detection through subtasks with different types of gold annotation. With a large set of systems, and well controlled experimental conditions, the tasks provided a great snapshot of progress in the field, which we aim to supplement by characterizing the major outstanding sources of error.

This work adds to previous investigations by providing a comprehensive and detailed analysis of errors. Our tool can automatically analyze any system's output, giving a reliable estimate of the relative importance of different error types.

3 Error Classification

When inspecting the output of coreference resolution systems, several types of errors become immediately apparent: entities that have been divided into pieces, spurious entities, non-referential pronouns that have been assigned antecedents, and so on. Our goal in this work is to automatically assign intuitive labels like these to errors in system output.

A simple approach, refining results by measuring the accuracy of subsets of the mentions, can be misleading. For example, in Figure 1, we can intuitively see two pronoun related mistakes: a missing mention (*he*), and a divided entity where the two pieces are the blue pronouns (I_2, I_2, *myself*$_2$) and the red proper names (*President Clinton*$_1$, *Mr. Clinton*$_1$).

Simply counting the number of incorrect pronoun links would miss the distinction between the two types of mistakes present.

One question in designing an error analysis tool like ours is whether to operate on just system output, or to also consider intermediate system decisions. We focused on using system output because other methods cannot uniformly apply to the full range of coreference resolution decoding methods, from link based methods to global inference methods.

Our overall approach is to transform the system output into the gold annotations, then map the changes made in the conversion process to errors. The transformation process is presented in Section 3.1 and Figure 2, and the mapping process is described in Section 3.2 and Figure 3.

3.1 Transformations

The first part of our error classification process determines the changes needed to transform the system output into the gold annotations. This five stage process is described below, and an abstract example is presented in Figure 2.

1. **Alter Span** transforms an incorrect system mention into a gold mention that has the same head token. In Figure 2 this stage is demonstrated by a mention in the leftmost entity, which has its span altered, indicated by the change from an X to a light blue circle.
2. **Split** breaks the system entities into pieces, each containing mentions from a single gold entity. In Figure 2 there are three changes in this stage: the leftmost entity is split into a red piece and a light blue piece, the middle entity is split into a dark red piece and an X, and the rightmost entity is split into singletons.
3. **Remove** deletes every mention that is not present in the gold annotations. In Figure 2 this means the four singleton X's are removed.
4. **Introduce** creates a singleton entity for each mention that is missing from the system output. In Figure 2 this stage involves the introduction of a light blue mention and two white mentions.
5. **Merge** combines entities to form the final, completely correct, set of entities. In Figure 2 the two red entities are merged, the singleton blue entity is merged with the rest of the blue entity, and the two white mentions are merged.

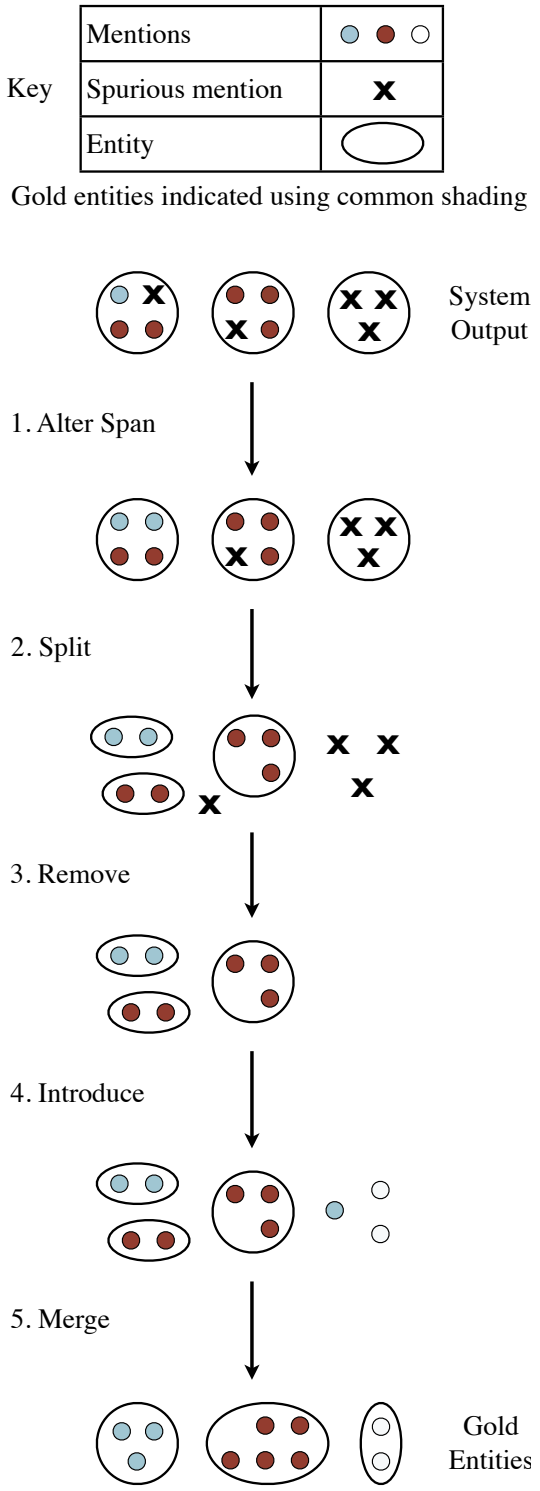


Figure 2: Abstract example of the transformation process that converts system output (at the top) to gold annotations (at the bottom).

	Operation(s)	Error	System	Gold
i)	Alter Span	Span error	*Gorbachev*	*Soviet leader Gorbachev*
ii)	Multiple Introduces and Merges	Missing Entity	- -	*the pills* *the tranquilizing pills*
iii)	Multiple Splits and Removes	Extra Entity	*human rights* *Human Rights*	- -
iv)	Introduce and Merge	Missing Mention	*the Arab region* *the region* -	*the Arab region* *the region* *it*
v)	Split and Remove	Extra Mention	*her story* *this* *it*	*her story* *this* -
vi)	Merge	Divided Entity	$Iraq_1$ *this* $nation_2$ *the* $nation_2$ its_1	$Iraq_1$ *this* $nation_1$ *the* $nation_1$ its_1
vii)	Split	Conflated Entities	*Mohammed* $Rashid_1$ *the Rashid* $case_1$ $Rashid_1$ *the* $case_1$	*Mohammed* $Rashid_1$ *the Rashid* $case_2$ $Rashid_1$ *the* $case_2$

Figure 3: Examples of the error types. In examples (i) - (iv) and (vi) the system output contains a single entity. When multiple entities are involved, they are marked with subscripts. Mentions are in the order in which they appear in the text. All examples are from system output on the dev set of the CoNLL task.

One subtle point in the split stage is how to record an entity being split into several pieces. This could either be a single operation, one entity being split into N pieces, or $N-1$ operations, each involving a single piece being split off from the rest of the entity. We use the second approach, as it fits more naturally with the error mapping we describe in the following section. Similarly, for the merge operation, we record N entities being merged as $N-1$ operations.

3.2 Mapping

The operations in Section 3.1 are mapped onto seven error types. In some cases, a single change maps onto a single error, while in others a single error represents several closely related operations from adjacent stages in the error correction process. The mapping is described below and in Figure 3.

1. **Span Error**. Each Alter Span operation is mapped to a Span Error, e.g. in Figure 3(i), the system mention *Gorbachev* is replaced by the annotated mention *Soviet leader Gorbachev*.
2. **Missing Entity**. A set of Introduce and Merge operations that forms an entirely new entity, e.g. the white entity in Figure 2, and *the pills* in Figure 3(ii). This error is still assigned if the new entity includes pronouns that were already present in the system output. The reasoning for this is that most pronouns in the corpus are coreferent, so including just the pronouns from an entity is not meaningfully different from missing the entity entirely.
3. **Extra Entity**. A set of Split and Remove operations that completely remove an entity, e.g. the rightmost entity in Figure 2, and Figure 3(iii). As for the Missing Entity error type, this error is still assigned if the original entity contained pronouns that were valid.
4. **Missing Mention**. An Introduce and a Merge that apply to the same mention, e.g. *it* in Figure 3(iv), and the blue mention in Figure 2.
5. **Extra Mention**. A Split and a Remove that apply to the same mention, e.g. *it* in Figure 3(v), and the X in the red entity in Figure 2.
6. **Divided Entity**. Each remaining Merge operation is mapped to a Divided Entity error, e.g. Figure 3(vi), and the red entity in Figure 2.
7. **Conflated Entities**. Each remaining Split operation is mapped to a Conflated Entity error, e.g. Figure 3(vii), and the blue and red entities in Figure 2.

4 Methodology

Our tool processes the CoNLL task output, with no other information required. During development, and when choosing examples for this paper, we used the development set of the CoNLL shared task (Hovy et al., 2006; Pradhan et al., 2007; Pradhan et al., 2011). The results we present in the rest of the paper are all for the test set. Using the development set would have been misleading, as the entrants in the shared task used it to tune their systems.

4.1 Systems

We analyzed all of the 2011 CoNLL task systems, as well as several publicly available systems. For the shared task systems we used the output data from the task itself, provided by the organizers. For the publicly available systems we used the default configurations. Finally, we included another run of the Stanford system, with their OntoNotes-tuned parameters (STANFORD-T).

The publicly available systems we used are: BERKELEY (Durrett and Klein, 2013), IMS (Björkelund and Farkas, 2012), STANFORD (Lee et al., 2013), RECONCILE (Stoyanov et al., 2010), BART (Versley et al., 2008), UIUC (Bengtson and Roth, 2008), and CHERRYPICKER (Rahman and Ng, 2009). The systems from the shared task are listed in Table 1 and in the references.

5 Broad System Comparison

Table 1 presents the frequency of errors for each system and F-Scores for standard metrics[1] on the test set of the 2011 CoNLL shared task. Each bar is filled in proportion to the number of errors the system made, with a full bar corresponding to the number of errors listed in the bottom row.

The metrics provide an effective overall ranking, as the systems with high scores generally make fewer errors. However, the metrics do not convey the significant variation in the types of errors systems make. For example, YANG and CHARTON are assigned almost the same scores, but YANG makes more than twice as many Extra Mention errors.

The most frequent error across all systems is Divided Entity. Unlike parsing errors (Kummerfeld et al., 2012), improvements are not monotonic, with better systems often making more errors of one type when decreasing the frequency of another type.

One outlier is the Irwin et al. (2011) system, which makes very few mistakes in five categories, but many in the last two. This reflects a high precision, low recall approach, where clusters are only formed when there is high confidence.

The third section of Table 1 shows results for systems that were run with gold noun phrase span information. This reduces all errors slightly, though most noticeably Extra Mention, Missing Mention, and Span Error. On inspection of the remaining Span Errors we found that many are due to inconsistencies regarding the inclusion of the possessive.

The final section of the table shows results for systems that were provided with the set of mentions that are coreferent. In this setting, three of the error types are not present, but there are still Missing Mentions and Missing Entities because systems do not always choose an antecedent, leaving a mention as a singleton, which is then ignored.

While this broad comparison gives a complete view of the range of errors present, it is still a coarse representation. In the next section, we characterize the common errors on a finer level by breaking down each error type by a range of properties.

6 Common Errors

To investigate the aggregate state of the art, in this section we consider results averaged over the top ten systems: CAI, CHANG, IMS, NUGUES, SANTOS, SAPENA, SONG, STANFORD-T, STOYANOV, URYUPINA-OPEN.[2] These systems represent a broad range of approaches, all of which are effective.

In each section below, we focus on one or two error types, characterizing the mistakes by a range of properties. We then consider a few questions that apply across multiple error types.

6.1 Span Errors

To characterize the Span Errors, we considered the text that is in the gold mention, but not the system

[1] CEAF and BLANC are not included as the most recent version of the CoNLL scorer (v5) is incorrect, and there are no standard implementations available.

[2] For systems that occur multiple times in Table 1, we only use the best instance. The BERKELEY system was not included as it had not been published at submission time.

System	Metric F-Scores Mention	MUC	B^3	Span Error	Conflated Entities	Extra Mention	Extra Entity	Divided Entity	Missing Mention	Missing Entity
PUBLICLY AVAILABLE SYSTEMS										
BERKELEY	75.57	66.43	66.17							
IMS	72.96	64.71	64.73							
STANFORD-T	71.21	61.40	63.06							
STANFORD	58.56	48.37	56.42							
RECONCILE	46.45	49.40	54.90							
BART	56.61	46.00	52.56							
UIUC	50.60	45.21	52.88							
CHERRYPICKER	41.10	40.71	51.39							
CONLL, PREDICTED MENTIONS										
LEE-OPEN	70.94	61.03	62.96							
LEE	70.70	59.56	61.88							
SAPENA	43.20	59.54	61.28							
SONG	67.26	59.95	60.08							
CHANG	64.86	57.13	61.75							
CAI-OPEN	67.45	57.86	60.89							
NUGUES	68.96	58.61	59.75							
URYUPINA-OPEN	68.39	57.63	58.74							
SANTOS	65.45	56.65	59.48							
STOYANOV	67.78	58.43	57.35							
HAO	64.30	54.46	55.82							
YANG	63.93	52.31	55.85							
CHARTON	64.36	52.49	55.61							
KLENNER-OPEN	62.28	49.86	55.62							
SOBHA	64.83	50.48	54.85							
ZHOU	62.31	48.96	53.42							
KOBDANI	61.03	48.62	53.00							
ZHANG	61.13	47.88	52.76							
XINXIN	61.92	46.62	51.50							
KUMMERFELD	62.72	42.70	50.05							
IRWIN-OPEN	35.27	27.21	44.29							
ZHEKOVA	48.29	24.08	41.42							
IRWIN	26.67	19.98	42.73							
CONLL, GOLD NP SPANS										
LEE-OPEN	75.39	65.39	65.88							
LEE	75.16	63.90	64.70							
NUGUES	72.42	62.12	61.67							
CHANG	67.91	59.77	62.97							
SANTOS	67.80	59.52	61.35							
STOYANOV	70.29	61.53	59.07							
SONG	66.68	55.48	58.04							
KOBDANI	66.08	53.94	55.82							
ZHANG	64.89	51.64	54.77							
ZHEKOVA	62.67	35.22	45.80							
CONLL, GOLD MENTIONS										
LEE-OPEN	90.93	81.56	75.95							
CHANG	99.97	82.52	73.68							
Most Errors				2410	3849	2744	5290	4789	2026	3237

Table 1: Counts for each error type on the test set of the 2011 CoNLL task. Bars indicate the number of errors, with white as zero and fully filled as the number in the *Most Errors* row. -OPEN indicates a system using external resources.

Type	Missing	Extra
NP	65.8	45.0
POS	12.4	96.9
,	71.2	22.4
SBAR	55.9	1.9
PP	46.2	10.3
DT	17.0	35.9
Total	271.1	224.6

Table 2: Counts of Span Errors grouped by the label over the extra/missing part of the mention.

mention (missing text), and vice versa (extra text). We then found nodes in the gold parse that covered just this extra/missing text, e.g. in Figure 3(i) we would consider the node over *Soviet leader*. In Table 2 we show the most frequent parse nodes.

Some of these differences are superficial, such as the possessive and the punctuation. Others, such as the missing PP and SBAR cases, may be due to parse errors. Of the system mentions involved in span errors, 27.0% do not correspond to a node in the gold parse. The frequency of punctuation errors could also be parse related, because punctuation is not considered in the standard parser evaluation.

Overall it seems that span errors can best be dealt with by improving parsing, though it is not possible to completely eliminate these errors because of inconsistent annotations.

6.2 Extra Mention and Missing Mention

We consider Extra and Missing Mentions together as they mirror each other, forming a precision-recall tradeoff, where a high precision system will have fewer Extra Mentions and more Missing Mentions, and a high recall system will have the opposite.

Table 3 divides these errors by the type of mention involved and presents some of the most frequent Extra Mentions and Missing Mentions. For the corpus statistics we count as mentions all NP spans in the gold parse plus any word tagged with PRP, WP, WDT, or WRB (following the definition of gold mention boundaries for the CoNLL tasks).

The mentions *it* and *you* are the most common errors, matching observations from several of the papers cited in Section 2. However, there is a surprising imbalance between Extra and Missing cases, e.g. *it* accounts for a third of the extra errors, but only 12% of the Missing errors. This imbalance may

	Av. Errors		Corpus Stats	
Mention	Extra	Missing	Count	% Coref.
Proper Name	281.6	297.7	6915	59.0
Nominal	484.2	516.5	33328	15.9
Pronoun	390.7	323.3	9926	69.7
it	130.4	38.9	1211	57.1
you	85.2	55.9	1028	44.9
we	39.6	19.6	691	64.7
us	23.2	3.2	242	23.6
that	13.8	13.4	2010	11.5
they	9.6	39.5	738	94.3
their	8.6	21.5	410	95.1
Total	1156.5	1137.5	50169	32.5

Table 3: Counts of Missing and Extra Mention errors by mention type, and the most common mentions.

	Proper Name		Nominal	
	Extra	Missing	Extra	Missing
Text match	145.2	163.6	171.2	96.1
Head match	56.8	70.7	149.6	166.0
Other	79.6	63.4	163.4	254.4
NER Matches	143.4	174.4	23.0	32.0
NER Differs	6.6	6.1	2.4	0.0
NER Unknown	131.6	117.2	458.8	484.5
Total	281.6	297.7	484.2	516.5

Table 4: Counts of Extra and Missing Mentions, grouped by properties of the mention and the entity it is in.

be the result of systems being tuned to the metrics, which seem to penalize Missing Mentions more than Extra Mentions (shown in Section 6.7).

In Table 4 we consider the Extra Mention errors and Missing Mention errors involving proper names and nominals. The top section counts errors in which the mention involved in the error has an exact string match with a mention in the cluster, or whether it has just a head match. The second section of the table considers the named entity annotations in OntoNotes, counting how often the mention's type matches the type of the cluster.

In all cases shown in the table it appears that systems are striking a balance between these two types of errors. One exception may be the use of exact string matching for nominals, which seems to be biased towards Extra Mentions.

For these two error types, our observations agree with previous work: the most common specific error is the identification of pleonastic pronouns, named entity types are of limited use, and head matching is already being used about as effectively as it can be.

Composition			Av. Errors	
Name	Nom	Pro	Extra	Missing
0	1	1	70.7	271.6
1	0	1	13.2	28.1
1	1	0	26.6	86.2
2	0	0	61.3	89.3
0	2	0	512.0	347.9
0	0	2	110.9	13.6
3+	0	0	14.7	14.4
0	3+	0	154.8	65.9
0	0	3+	91.0	18.1
Other			51.8	216.4
Total			1107.0	1151.5

Table 5: Counts of Extra and Missing Entity errors, grouped by the composition of the entity (Names, Nominals, Pronouns).

Match	Type	Extra	Missing
Exact	Proper Name	51.4	42.2
	Nominal	338.3	49.5
	Pronoun	141.9	10.3
Head	Proper Name	14.4	27.3
	Nominal	234.7	129.0
None	Proper Name	10.2	34.2
	Nominal	92.8	235.3
	Pronoun	60.0	21.4

Table 6: Counts of Extra and Missing Entity errors grouped by properties of the mentions in the entity.

6.3 Extra Entities and Missing Entities

In this section, we consider the errors that involve an entire entity that was either missing from the system output or does not exist in the annotations.

Table 5 counts these errors based on the composition of the entity. There are several noticeable differences between the two error types, e.g. for entities containing one nominal and one pronoun (row 0 1 1) there are far more Missing errors than Extra errors, while entities containing two pronouns (row 0 0 2) have the opposite trend.

It is clear that entities consisting of a single type of mention are the primary source of these errors, accounting for 85.3% of the Extra Entity errors, and 47.7% of Missing Entity errors. Table 6 shows counts for these cases divided into three groups: when all mentions are identical, when all mentions have the same head, and the rest.

Nominals are the most frequent type in Table 6, and have the greatest variation across the three sections of the table. For the Extra column, Exact match cases are a major challenge, accounting for over half of the nominal errors. These errors include cases like the example below, where two mentions are not considered coreferent because they are generic:

everybody tends to mistake the part for the whole.
Here, mistaking the part for the whole is ...

For missing entities we see the opposite trend, with Exact match cases accounting for less than 12% of nominal errors. Instead, cases with no match are the greatest challenge, such as this example, which requires semantic knowledge to correctly resolve:

The charges related to her sale of ImClone stock.
She sold the share a day before ...

The other common case in Table 5 is an entity containing a pronoun and a nominal. In Table 7 we present the most frequent pronouns for this case and the similar case involving a pronoun and a name.

Mention	Extra	Missing
that	6.9	99.7
it	47.7	47.8
this	0.9	36.2
they	3.8	29.1
their	2.1	23.5
them	0.9	13.8
Any pronoun	83.9	299.7

Table 7: Counts of common Missing and Extra Entity errors where the entity has just two mentions: a pronoun and either a nominal or a proper name.

One way of interpreting these errors is from the perspective of the pronoun, which is either incorrectly coreferent (Extra), or incorrectly non-coreferent (Missing). From this perspective, these errors are similar in nature to those described by Table 3. However, the distribution of errors is quite different, with *it* being balanced here where previously it skewed heavily towards extra mentions, while *that* was balanced in Table 3 but is skewed towards being part of Missing Entities here.

Extra Entity errors and Missing Entity errors are particularly challenging because they are dominated by entities that are either just nominals, or a nominal and a pronoun, and for these cases the string matching features are often misleading. This implies that reducing Extra Entity and Missing Entity errors will require the use of discourse, context, and semantics.

Incorrect Part			Rest of Entity			Av. Errors	
Na	No	Pr	Na	No	Pr	Conflated	Divided
-	-	1+	-	-	1+	312.7	69.9
-	-	1+	-	1+	1+	238.5	179.8
-	-	1+	-	1+	-	189.6	549.3
-	1+	-	-	1+	-	181.5	156.5
-	-	1+	1+	1+	1+	143.6	181.5
-	-	1+	1+	-	1+	109.7	150.5
-	-	1+	1+	-	-	60.0	136.5
Other						454.8	657.7
Total						1690.4	2081.7

Table 8: Counts of Conflated and Divided entities errors grouped by the Name / Nominal / Pronoun composition of the parts involved.

6.4 Conflated Entities and Divided Entities

Table 8 breaks down the Conflated Entities errors and Divided Entity errors by the composition of the part being split/merged and the rest of the entity involved. Each 1+ indicates that at least one mention of that type is present (Name / Nominal / Pronoun).

Clearly pronouns being placed incorrectly is the biggest issue here, with almost all of the common errors involving a part with just pronouns. It is also clear that not having proper names in the rest of the entity presents a challenge. One particularly noticeable issue involves entities composed entirely of pronouns, which are often created by systems conflating the pronouns of two entities together.

Table 8 aggregates errors by the presence of different types of mentions. Aggregating instead by the exact composition of the incorrect part being conflated or divided we found that instances with a part containing a single pronoun account for 38.9% of conflated cases and 35.8% of divided cases.

Finally, it is worth noting that in many cases a part is both conflated with the wrong entity, and divided from its true entity. Only 12.6% of Conflated Entity errors led to a complete gold entity with no other errors, and only 21.3% of Divided Entity errors came from parts that were not involved in another error.

Conflated Entities and Divided Entities are dominated by pronoun link errors: cases where a pronoun was placed in the wrong entity. Finding finer characterizations of these errors is difficult, as almost any division produces sparse counts, reflecting the long tail of mistakes that make up these two error types.

Gold	System Decision	Count
Cataphoric	Same referent	10.6
	Different referent	13.4
	Not cataphoric	208.2
	Not present	42.8
Not cataphoric	Cataphoric	46.2
Not present	Cataphoric	186.8

Table 9: Occurrence of mistakes involving cataphora.

6.5 Cataphora

Cataphora (when an anaphor precedes its antecedent) is a pronoun-specific problem that does not fit easily in the common left-to-right coreference resolution approach. In the CoNLL test set, 2.8% of the pronouns are cataphoric. In Table 9 we show how well systems handle this challenge by counting mentions based on whether they are cataphoric in the annotations, are cataphoric in the system output, and whether the antecedents match.

Systems handle cataphora poorly, missing almost all of the true instances, and introducing a large number of extra cases. However, this issue is a fairly small part of the task, with limited metric impact.

6.6 Entity Properties

Gender, number, person, and named entity type are properties commonly used in coreference resolution systems. In some cases, two mentions with different properties are placed in the same entity. Some of these cases are correct, such as variation in person between mentions inside and outside of quotes. However, many of these cases are errors. In Table 11 we present the percentage of entities that contain mentions with properties of more than one type. For named entity types we considered the annotations in OntoNotes; for the other properties we derive them from the pronouns in each cluster.

For all of the properties, there are many entities that we could not assign a value to, either because no named entity information was available, or because no pronouns with an unambiguous value for the property were present. For named entity information, OntoNotes only has annotations for 68% of gold entities, suggesting that named entity taggers are of limited usefulness, matching observations on the MUC and ACE corpora (Stoyanov et al., 2009).

The results in the 'Gold' column of Table 11 in-

	Mentions			MUC			B^3		
Error type	P	R	F	P	R	F	P	R	F
Span Error	2.8	2.8	2.7	2.8	2.8	2.8	1.0	2.0	1.6
Conflated Entities	1.7	0.0	0.8	9.9	0.0	4.5	**15.9**	0.0	6.2
Extra Mention	5.5	0.0	2.6	6.4	0.0	3.0	5.3	0.0	2.2
Extra Entity	**15.3**	0.0	7.0	**11.4**	0.0	5.2	6.1	0.0	2.4
Divided Entity	1.8	6.8	4.3	5.7	**16.8**	**10.9**	-10.0	**21.6**	4.5
Missing Mention	1.8	7.0	4.4	3.2	9.2	6.1	-1.3	7.3	3.4
Missing Entity	3.8	**16.2**	**9.8**	5.3	13.7	9.3	1.7	11.4	**7.0**

Table 10: Average accuracy improvement if all errors of a particular type are corrected. Each row in the lower section is calculated independently, relative to the change after the span errors have been corrected. Some values are negative because the merge operations involved in fixing the errors are applying to clusters that contain mentions from more than one gold entity.

Property	System	Gold
Named Entity	1.7%	0.7%
Gender	0.8%	0.1%
Number	2.1%	0.8%
Person	6.4%	5.1%

Table 11: Percentage of entities that contain mentions with properties that disagree.

dicate possible errors in the annotations, e.g. in the 0.7% of entities with a mixture of named entity types there may be mistakes in the coreference annotations, or mistakes in the named entity annotations.[3] However, even after taking into consideration cases where the mixture is valid and cases of annotation errors, current systems are placing mentions with different properties in the same clusters.

6.7 Impact of Errors on Metric Scores

Table 10 shows the performance impact of correcting errors of each type. The Span Error row gives improvements over the original scores, while all other rows are relative to the scores after Span Errors are corrected.[4] By fixing each of the other error types in isolation, we can get a sense of the gain if just that error type is addressed. However, it also means some mentions are incorrectly placed in the same cluster, causing some negative scores.

Interaction between the error types and the way the metrics are defined means that the deltas do not add up to the overall average gap in performance, but it is still clear that every error type has a noticeable impact. Missing Entity errors have the most substantial impact, which reflects the precision oriented nature of many coreference resolution systems.

7 Conclusion

While the improvement of metrics and the organization of shared tasks have been crucial for progress in coreference resolution, there is much insight to be gained by performing a close analysis of errors.

We have presented a new means of automatically classifying coreference errors that provides an exhaustive view of error types. Using our tool we have analyzed the output of a large set of coreference resolution systems and investigated the common challenges across state-of-the-art systems.

We find that there is considerable variability in the distribution of errors, and the best systems are not best across all error types. No single source of errors stands out as the most substantial challenge today. However, it is worth noting that while confidence measures can be used to reduce precision-related errors, no system has been able to effectively address the recall-related errors, such as Missed Entities. Our analysis tool is available at code.google.com/p/berkeley-coreference-analyser/.

Acknowledgments

We would like to thank the CoNLL task organizers for providing us with system outputs. This work was supported by a General Sir John Monash fellowship to the first author and by BBN under DARPA contract HR0011-12-C-0014.

[3] This kind of cross-annotation analysis may be a useful way of detecting annotation errors.

[4] This difference was necessary as the later errors make changes relative to the state of the entities after the Span Errors are corrected, e.g. in Figure 2 a blue and red entity is split that previously contained an X instead of one of the blue mentions.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 49–55.

Anders Björkelund and Pierre Nugues. 2011. Exploring lexicalized features for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 45–50.

Jie Cai, Eva Mujdricza-Maydt, and Michael Strube. 2011. Unrestricted coreference resolution via global hypergraph partitioning. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 56–60.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Nick Rizzolo, Mark Sammons, and Dan Roth. 2011. Inference protocols for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 40–44.

Eric Charton and Michel Gagnon. 2011. Poly-co: a multilayer perceptron approach for coreference detection. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 97–101.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 56–63.

Weipeng Chen, Muyu Zhang, and Bing Qin. 2011. Coreference resolution system using maximum entropy classifier. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 127–130.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1152–1161.

Gordana Ilic Holen. 2013. Critical reflections on evaluation practices in coreference resolution. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 1–7.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Joseph Irwin, Mamoru Komachi, and Yuji Matsumoto. 2011. Narrative schema as world knowledge for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 86–92.

Manfred Klenner and Don Tuggener. 2011. An incremental model for coreference resolution with restrictive antecedent accessibility. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 81–85.

Hamidreza Kobdani and Hinrich Schuetze. 2011. Supervised coreference resolution with sucre. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 71–75.

Jonathan K. Kummerfeld, Mohit Bansal, David Burkett, and Dan Klein. 2011. Mention detection: Heuristics for the ontonotes annotations. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 102–106.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059.

Sobha Lalitha Devi, Pattabhi Rao, Vijay Sundar Ram R, M. C S, and A. A. 2011. Hybrid approach for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 93–96.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanfords multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

Xinxin Li, Xuan Wang, and Shuhan Qi. 2011. Coreference resolution with loose transitivity constraints.

In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 107–111.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Sebastian Martschat, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. 2012. A multigraph model for coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 100–106.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 104–111.

Cicero Nogueira dos Santos and Davi Lopes Carvalho. 2011. Rule and tree ensembles for unrestricted coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 51–55.

C. D. Paice and G. D. Husk. 1987. Towards the automatic recognition of anaphoric features in english text: the impersonal pronoun 'it'. *Computer Speech & Language*, 2(2):109–132.

Sameer Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in ontonotes. In *Proceedings of the International Conference on Semantic Computing*, pages 446–453.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the 15th Conference on Computational Natural Language Learning (CoNLL 2011)*, pages 1–27.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977.

M. Recasens and E. Hovy. 2011. BLANC: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17:485–510, 9.

Emili Sapena, Lluís Padró, and Jordi Turmo. 2011. Relaxcor participation in conll shared task on coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 35–39.

Lalitha Devi. Sobha, RK. Rao. Pattabhi, R. Vijay Sundar Ram, CS. Malarkodi, and A. Akilandeswari. 2011. Hybrid approach for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 93–96.

Yang Song, Houfeng Wang, and Jing Jiang. 2011. Link type based pre-cluster pair model for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 131–135.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 656–664.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with reconcile. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 156–161.

Veselin Stoyanov, Uday Babbar, Pracheer Gupta, and Claire Cardie. 2011. Reconciling ontonotes: Unrestricted coreference resolution in ontonotes with reconcile. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 122–126.

Olga Uryupina, Sriparna Saha, Asif Ekbal, and Massimo Poesio. 2011. Multi-metric optimization for coreference: The unitn / iitp / essex submission to the 2011 conll shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 61–65.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. Bart: a modular toolkit for coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session*, pages 9–12.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Uunderstanding Conference*, pages 45–52.

Hao Xiong, Linfeng Song, Fandong Meng, Yang Liu, Qun Liu, and Yajuan Lv. 2011. Ets: An error tolerable system for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 76–80.

Yaqin Yang, Nianwen Xue, and Peter Anick. 2011. A machine learning-based coreference detection system

for ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 117–121.

Bo Yuan, Qingcai Chen, Yang Xiang, Xiaolong Wang, Liping Ge, Zengjian Liu, Meng Liao, and Xianbo Si. 2012. A mixed deterministic model for coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 76–82.

Desislava Zhekova and Sandra Kübler. 2011. Ubiu: A robust system for resolving unrestricted coreference. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 112–116.

Desislava Zhekova, Sandra Kübler, Joshua Bonner, Marwa Ragheb, and Yu-Yin Hsu. 2012. Ubiu for multilingual coreference resolution in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 88–94.

Huiwei Zhou, Yao Li, Degen Huang, Yan Zhang, Chunlong Wu, and Yuansheng Yang. 2011. Combining syntactic and semantic features by svm for unrestricted coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 66–70.

Exploiting Zero Pronouns to Improve Chinese Coreference Resolution

Fang Kong
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
dcskf@nus.edu.sg

Hwee Tou Ng
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417
nght@comp.nus.edu.sg

Abstract

Coreference resolution plays a critical role in discourse analysis. This paper focuses on exploiting zero pronouns to improve Chinese coreference resolution. In particular, a simplified semantic role labeling framework is proposed to identify clauses and to detect zero pronouns effectively, and two effective methods (refining syntactic parser and refining learning example generation) are employed to exploit zero pronouns for Chinese coreference resolution. Evaluation on the CoNLL-2012 shared task data set shows that zero pronouns can significantly improve Chinese coreference resolution.

1 Introduction

As one of the most important tasks in discourse analysis, coreference resolution aims to link a given mention (i.e., entity or event) to its co-referring expression in a text and has been a focus of research in natural language processing (NLP) for decades.

Over the last decade, various machine learning techniques have been applied to coreference resolution and have performed reasonably well (Soon et al., 2001; Ng and Cardie, 2002; Fernandes et al., 2012). Current techniques rely primarily on surface level features such as string match, syntactic features such as apposition, and shallow semantic features such as number, gender, semantic class, etc.

Despite similarities between Chinese and English, there are differences that have a significant impact on coreference resolution. In this paper, we focus on exploiting one of the key characteristics of Chinese text, zero pronouns (ZPs), to improve Chinese coreference resolution. In particular, a simplified semantic role labeling (SRL) framework is proposed to identify Chinese clauses and to detect zero pronouns effectively, and two effective methods are employed to exploit zero pronouns for Chinese coreference resolution. Experimental results show the effectiveness of our approach in improving the performance of Chinese coreference resolution. Our work is novel in that it is the first work that incorporates the use of zero pronouns to significantly improve Chinese coreference resolution

The rest of this paper is organized as follows. Section 2 describes our baseline Chinese coreference resolution system. Section 3 motivates how the detection of zero pronouns can improve Chinese coreference resolution, using an illustrating example. Section 4 presents our approach to detect zero pronouns. Section 5 proposes two methods to exploit zero pronouns to improve Chinese coreference resolution, based on a corpus study and preliminary experiments. Section 6 briefly outlines the related work. Finally, we conclude our work in Section 7.

2 Chinese Coreference Resolution

According to Webber (1978), coreference resolution can be decomposed into two complementary subtasks: (1) anaphoricity determination: deciding whether a given noun phrase (NP) is anaphoric or not; and (2) anaphora resolution: linking together multiple mentions of a given entity in the world. Our Chinese coreference resolution system also contains these two components. Using the training data set of CoNLL-2012 shared task, we first train an anaphoricity classifier to determine whether

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 278–288,
Seattle, Washington, USA, 18-21 October 2013.

a mention is anaphoric or not, and then employ an independently-trained coreference resolution system to resolve those mentions which are classified as anaphoric. The lack of gender and number makes both anaphoricity determination and coreference resolution in Chinese more difficult.

2.1 Anaphoricity Determination

Since only the mentions that take part in coreference chains are annotated in the CoNLL-2012 shared task data set, we first generate a high-recall, low-precision mention extraction module to extract as many mentions as possible. The mention extraction module relies mainly on syntactic parse trees. We extract all NP nodes, QP (quantifier phrase, i.e., complex amount/measure phrase) nodes, and all terminals with part-of-speech tags PN (pronoun) and NR (proper noun) in parse trees to form a mention candidate set. Then, we employ some rules to remove unlikely mentions, e.g., those which contain (1) measure words such as '一年/one year' and '一顿/one time'; (2) named entities whose categories are PERCENT, MONEY, QUANTITY, and CARDINAL; (3) interrogative pronouns such as '什么/what' and '哪儿/where'.

After pruning, we employ a learning-based method to train an independent classifier to determine whether the remaining mentions are anaphoric. Table 1 lists all the features employed in our anaphoricity determination system.

2.2 Coreference Resolution

Our Chinese coreference resolution system adopts the same learning-based model and the same set of 12 features as Soon et al. (2001). Considering the special characteristics of conversation and web texts (i.e., a large proportion of personal pronouns and the organization of a text into several parts[1]) and preparing for dealing with zero pronouns, we add some features shown in Table 2.[2]

[1]A text in the CoNLL-2012 data set is broken down into different "parts".

[2]AN denotes anaphor, CA denotes antecedent candidate, IP denotes a simple clause, and CP denotes a clause headed by a complementizer. For the feature ANPronounRanking, the relative ranking of a given pronoun is based on its semantic role and surface position, and we assign the highest rank to zero pronouns, similar to Kong et al. (2009).

	R	P	F
GS	76.32	87.14	81.37
Auto	64.87	78.42	71.00

Table 3: Performance of anaphoricity determination on the CoNLL-2012 test set

		R	P	F
AM	Mention Detection	65.26	67.20	66.22
	MUC	51.64	61.82	56.27
	BCUBED	73.40	80.38	76.73
	CEAF	53.16	45.66	49.13
	Average			60.71
GMB	Mention Detection	82.01	69.58	75.29
	MUC	76.21	66.18	70.84
	BCUBED	76.15	86.59	81.04
	CEAF	59.75	50.52	54.75
	Average			68.88
GM	Mention Detection	79.80	100.00	88.77
	MUC	80.86	85.48	83.11
	BCUBED	73.66	91.94	81.79
	CEAF	67.54	64.87	66.18
	Average			77.02

Table 4: Performance of our Chinese coreference resolution system on the CoNLL-2012 test set

2.3 Results and Analysis

All experiments in this section are conducted on the CoNLL-2012 shared task data set. The SVM-light toolkit (Joachims, 1999) with radial basis kernel and default learning parameters is employed in both anaphoricity determination and coreference resolution.

Table 3 reports the performance of anaphoricity determination on the CoNLL-2012 test set using gold-standard parse trees (GS) and automatic parse trees (Auto). All performance figures in this paper are given in percentages. The results show that using both gold parse trees and automatic parse trees, our anaphoricity determination system achieves higher precision than recall. In comparison with using gold parse trees, precision decreases by about 9% and recall 11% on automatic parse trees.

Table 4 reports the performance of our Chinese coreference resolution system on the CoNLL-2012 test set under three different experimental settings: with automatic mentions (AM), with gold mention

Feature	Description
NPType	Type of the current mention (pronoun, demonstrative, proper NP).
NPNumber	Number of the current mention (singular, plural).
NPGender	Gender of the current mention (male, female).
IsHeadWord	Whether the current mention is the same as its headword.
StrMatch	Whether there is a string match between the current mention and another phrase in the previous context.
AliasMatch	Whether the current mention is a name alias or abbreviation of another phrase in the previous context.
Appositive	Whether the current mention and another phrase in the previous context are in an appositive relation.
NestIn	Whether another NP is nested in the current mention.
NestOut	Whether the current mention is nested in another NP.
FirstNP	Whether the current mention is the first NP of the sentence.
FrontDistance	The number of words between the current mention and the nearest previous clause.
BackDistance	The number of words between the current mention and the nearest following clause.
WordSense	Whether the current mention and another phrase in the previous context have the same word sense. Word sense annotation is provided in the CoNLL-2012 data set, based on the IMS software (Zhong and Ng, 2010).

Table 1: Features employed in our anaphoricity determination system

Feature	Description
AN/CAPronounType	Whether the anaphor or the antecedent candidate is a zero pronoun, first person, second person, third person, neutral pronoun, or others. In our coreference resolution system, a zero pronoun is viewed as a kind of special pronoun.
AN/CAGrammaticalRole	Whether the anaphor or the antecedent candidate is a subject, object, or others.
AN/CAOwnerClauseType	Whether the anaphor or the antecedent candidate is in a matrix clause, an independent clause, a subordinate clause, or none of the above.
AN/CARootPath	Whether the path of nodes from the anaphor (or the antecedent candidate) to the root of the parse tree contains NP, IP, CP, or VP.
ANPronounRanking	Whether the anaphor is a pronoun and is ranked highest among the pronouns (including zero pronouns) of the sentence.
AN/CAClosestNP	Whether the antecedent candidate is the closest preceding NP of the anaphor.
AN/CAPartDistance	This feature captures the distance (in parts) between the antecedent candidate and the anaphor. If they are in the same part, the value is 0; if they are one part apart, the value is 1; and so on.
AN/CASameSpeaker	Whether the antecedent candidate and the anaphor appear in sentences spoken by the same person.

Table 2: Additional features employed in our Chinese coreference resolution system

boundaries (GMB), and with gold mentions (GM). From the results, we find that:

- Using automatic mentions, our system achieves 56.27, 76.73, and 49.13 in F-measure on MUC, BCUBED, and CEAF evaluation metrics, respectively.
- Using gold mention boundaries improves the performance of our system by 14.57, 4.31, and 5.62 in F-measure, due to large gains in both recall and precision. We also find that using gold mention boundaries can boost the recall of mention detection. As described above, our anaphoricity determination model relies mainly on the parser. Using gold mention boundaries can improve the parser performance. Thus our coreference resolution system can benefit much from using gold mention boundaries (especially the recall).
- Employing gold mentions further boosts our system significantly. In comparison with using gold mention boundaries, the performance improvement is attributed more to an increase in precision.

In comparison with the three best systems of CoNLL-2012 in the Chinese closed track (shown in Table 5), considering average F-measure, we find that using automatic mentions, our system is only inferior to that of Chen and Ng (2012); using gold mention boundaries, our system achieves the best performance; and using gold mentions, our system is only a little worse than that of Chen and Ng (2012).

3 Motivation

In order to analyze the impact of zero pronouns on Chinese coreference resolution, we first use the released OntoNotes v5.0 data (i.e., the training and development portions of the CoNLL-2012 shared task) in a corpus study.

Statistics show that anaphoric zero pronouns account for 10.7% of the mentions in coreference chains in the training data, while in the development data, the proportion is 11.3%. The experimental results of our Chinese coreference resolution system (i.e., the baseline) show that using both gold mention boundaries and gold mentions significantly improves system performance, especially for recall, largely due to improved parser performance. We then analyze the impact of zero pronouns on Chinese syntactic parsing. As a preliminary exploration, we integrate Chinese zero pronouns into the Berkeley parser (Petrov et al., 2006), experimenting with gold-standard or automatically determined zero pronouns kept or stripped off (using gold-standard word segmentation provided in the CoNLL-2012 data). The results indicate that given gold-standard zero pronouns, parsing performance improves by 1.8% in F-measure. Using automatically determined zero pronouns by our zero pronoun detector to be introduced in Section 4, parsing performance also improves by 1.4% in F-measure.

In order to illustrate the impact of zero pronouns on parsing performance, consider the following example:[3]

> Example (1):
>
> 将来我们有**一个重建计划**。
>
> **#**分公园成七个区域，**#**带来多一些的景点。
>
> ...
>
> **这个计划**我们现在是等到政府的批准我们就可以再进行，预算是**#**明年可以动工了。
>
> (In future, we have **a reconstruction plan**.
>
> Divide the park into seven regions, and bring some more attractions.
>
> ...
>
> Now we wait for approval of the government before implementing **this plan** again. It is expected that work can start next year.)

Without considering zero pronouns, the parse tree of the second sentence output by the Berkeley parser is shown in Figure 1.

Prior to parsing, using our zero pronoun detector to be introduced in Section 4, the presence of zero pronouns (denoted by #) can be detected. Figure 2

[3] In this paper, zero pronouns are denoted by "#" and mentions in the same coreference chain are shown in bold for all examples.

		MD	MUC	BCUBED	CEAF	Avg
AM	(Chen and Ng, 2012)	71.64	62.21	73.55	50.97	62.24
	(Yuan et al., 2012)	68.15	60.33	72.90	48.83	60.69
	(Björkelund and Farkas, 2012)	66.37	58.61	73.10	48.19	59.97
	Our baseline system (without ZPs)	66.22	56.27	76.73	49.13	60.71
	Our refined system (with auto ZPs)	70.33	59.58	78.15	51.47	63.07
GMB	(Chen and Ng, 2012)	80.45	71.43	77.04	57.17	68.55
	(Yuan et al., 2012)	74.02	66.44	75.02	51.81	64.42
	(Björkelund and Farkas, 2012)	71.02	63.56	74.52	50.20	62.76
	Our baseline system (without ZPs)	75.29	70.84	81.04	54.75	68.88
	Our refined system (with auto ZPs)	75.77	72.62	81.45	58.04	70.70
GM	(Chen and Ng, 2012)	91.73	83.77	81.15	68.38	77.77
	(Yuan et al., 2012)	89.95	82.79	79.79	65.58	76.05
	(Björkelund and Farkas, 2012)	83.47	76.85	76.30	56.61	69.92
	Our baseline system (without ZPs)	88.77	83.11	81.79	66.18	77.02
	Our refined system (with auto ZPs)	91.49	83.46	82.43	65.88	77.26

Table 5: Performance (F-measure) of the three best Chinese coreference resolution systems on the CoNLL-2012 test set

shows the new parse tree, which includes the detected zero pronouns, output by the Berkeley parser on the same sentence. Comparing these two parse trees, we can see that the detected zero pronouns contribute to better division of clauses and improved parsing performance, which in turn leads to improved Chinese coreference resolution.

Detecting the presence of zero pronouns also helps to improve local salience modeling, leading to improved Chinese coreference resolution. Long sentences containing multiple clauses occur more frequently in Chinese compared to English. Furthermore, a coreference chain can span many sentences. Zero pronouns can occur not only within one sentence (e.g., the first and second zero pronouns of Example (1)), but can also be scattered across multiple sentences (e.g., the first and third zero pronouns of Example (1)). The subjects in the second sentence of Example (1) are omitted.[4] Detection of zero pronouns improves local salience modeling, and leads to the correct identification of all the noun phrases of the coreference chain in Example (1).

4 Zero Pronoun Detection

Empty elements are those nodes in a parse tree that do not have corresponding surface words or phrases. Although empty elements exist in many languages and serve different purposes, they are particularly important for some languages, such as Chinese, where subjects and objects are frequently dropped to keep a discourse concise. Among empty elements, type *pro*, namely zero pronoun, is either used for dropped subjects or objects, which can be recovered from the context (anaphoric), or it is of little interest for the reader or listener to know (non-anaphoric). In the Chinese Treebank, type *pro* constitutes about 20% (Yang and Xue, 2010), and more than 85% of them are anaphoric (Kong and Zhou, 2010). Thus, zero pronouns are very important in bridging the information gap in a Chinese text. In this section, we will introduce our zero pronoun detector.

In Chinese, a zero pronoun always occurs just before a predicate phrase node (e.g., VP). In particular, if the predicate phrase node occurs in a coordinate structure or is modified by an adverbial node, we only need to consider its parent. A simplified semantic role labeling (SRL) framework (only including predicate recognition, argument pruning, and argument identification) is adopted to identify the predicate phrase subtree (Xue, 2008), i.e., the minimal subtree governed by a predicate and all its arguments.

We carry out zero pronoun detection for every predicate phrase subtree in an iterative manner from a parse tree, i.e., determining whether there is a zero pronoun before the given predicate phrase sub-

[4] In Chinese, pro-dropped subjects account for more than 36% of subjects in sentences (Kim, 2000).

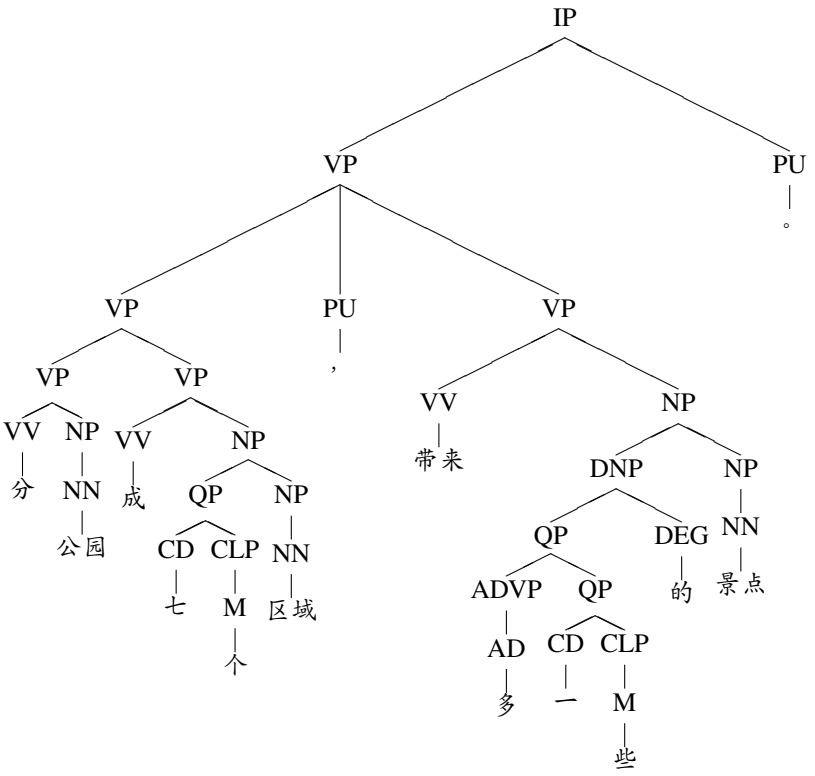


Figure 1: The parse tree without considering zero pronouns

tree. Viewing the position before the given predicate phrase subtree as a zero pronoun candidate, we can perform zero pronoun detection using a machine learning approach.

During training, if a zero pronoun candidate has a counterpart in the same position in the annotated training corpus (either anaphoric or non-anaphoric), a positive example is generated. Otherwise, a negative example is generated. During testing, each zero pronoun candidate is presented to the zero pronoun detector to determine whether it is a zero pronoun.

The features that are employed to detect zero pronouns mainly model the context of the clause itself, the left and right siblings, and the path of the clause to the root node. Table 6 lists the features in detail.

4.1 Results and Analysis

We evaluate our zero pronoun detector using gold parse trees and automatic parse trees produced by the Berkeley parser. The SVM-light toolkit with radial basis kernel and default learning parameters is employed as our learning algorithm.

Table 7 lists the results. From the results, we find that the performance of our zero pronoun detector drops about 12% in F-measure when using automatic parse trees, compared to using gold parse trees. That is, the performance of zero pronoun detection also depends on the performance of the syntactic parser.

	R	P	F
GS	89.32	87.29	88.29
Auto	74.19	77.79	75.95

Table 7: Performance of zero pronoun detection on the test set using gold and automatic parse trees

5 Exploiting Zero Pronouns to Improve Chinese Coreference Resolution

In this section, we will propose two methods, refining the syntactic parser and refining learning example generation, to exploit zero pronouns to improve Chinese coreference resolution.

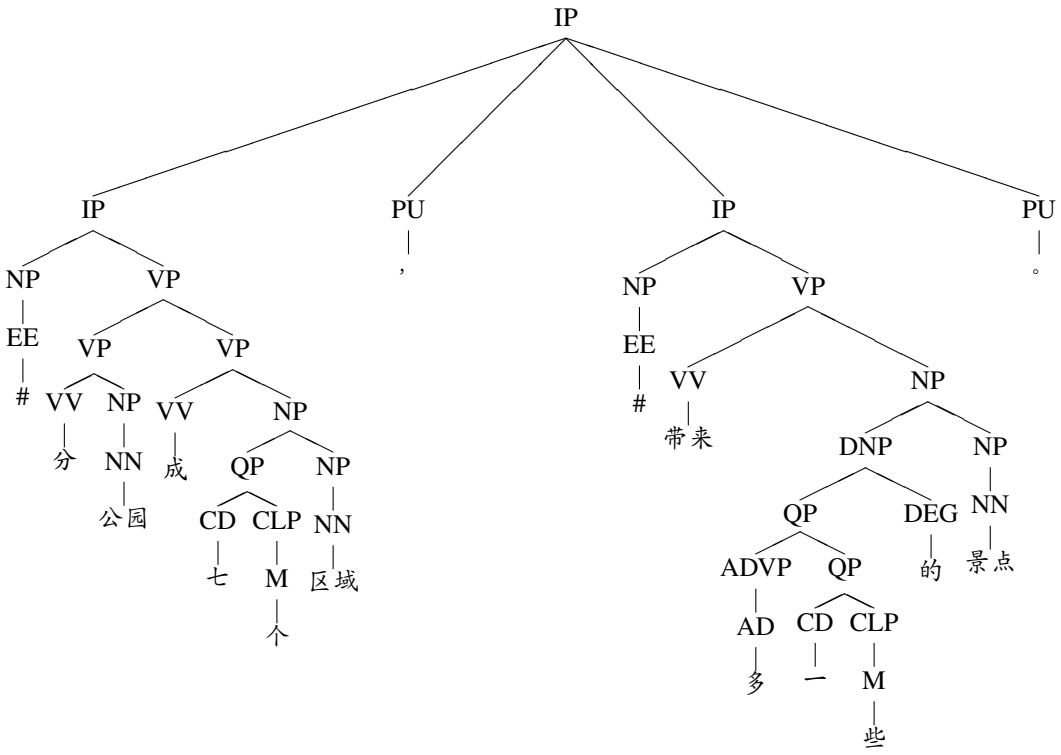


Figure 2: The parse tree with the detected zero pronouns

5.1 Refining the Syntactic Parser

Similar to our preliminary experiments, we retrain the Berkeley parser with explicit, automatically detected zero pronouns in the training set and parse the test set with explicit, automatically detected zero pronouns using the retrained model. In both anaphoricity determination and coreference resolution, the output results of the retrained parser are employed to generate all features.

5.2 Refining Learning Example Generation

In order to model the salience of all entities, we regard all zero pronouns as a special kind of NPs when generating the learning examples. Considering the modest performance of our anaphoricity determination module, we do not determine the anaphoricity of zero pronouns. Instead, in the coreference resolution stage, all zero pronouns will be considered during learning example generation (including both training and test example generation).

For example, consider a coreference chain A1-A2-Z0-A3-A4 containing one zero pronoun found in an annotated training document. A1, A2, A3, and A4 are traditional entity mentions, and Z0 is a zero pronoun. During training, pairs of mentions in the chain that are immediately adjacent (i.e., A1-A2, A2-Z0, Z0-A3, and A3-A4) are used to generate the positive training examples. Among them, two examples (i.e., A2-Z0 and Z0-A3) are associated with a zero pronoun, which can act as both an anaphor and an antecedent. For each positive pair, e.g., Z0-A3, we find any noun phrase and zero pronoun occurring between the anaphor A3 and the antecedent Z0, and pair each of them with A3 to form a negative example. Similarly, test examples can be generated except that only the preceding mentions and zero pronouns in the current and previous two sentences will be paired with an anaphor.

Incorporating zero pronouns models salience of all entities more accurately. The ratio of positive to negative examples is also less skewed as a result of considering zero pronouns – the ratio changes from 1:7.9 to 1:6.8 after considering zero pronouns.

5.3 Reprocessing

Although in the OntoNotes corpus, dropped subjects and objects (i.e., zero pronouns) are considered during coreference resolution for Chinese, they are not

Feature	Description
ClauseClass	Whether the given clause is a terminal clause or non-terminal clause.
LeftSibling	Whether the given clause has a sibling immediately to its left.
LeftSiblingNP	Whether the left siblings of the given clause contain an NP.
RightSibling	Whether the given clause has a sibling immediately to its right.
RightSiblingVP	Whether the right siblings of the given clause contain a VP.
ParentIP/VP	Whether the syntactic category of the immediate parent of the given clause is an IP or VP.
RootPath	Whether the path from the given clause to the root of the parse tree contains an NP or VP or CP. This feature models how the given clause is syntactically connected to the sentence as a whole, reflecting its function within the sentence.
ClauseType	The given clause is an independent clause, a subordinate clause, or others.
Has-Arg0/Arg1	Whether the given clause has an agent or patient argument.

Table 6: Features employed to detect zero pronouns

used in the CoNLL-2012 shared task (i.e., in the gold evaluation keys, all the links formed by zero pronouns are removed).

As described in Subsection 5.2, during training and testing, all links associated with zero pronouns will be considered in our coreference resolution system. That is, we do not distinguish zero pronoun resolution from traditional coreference resolution, and only view zero pronouns as special pronouns. After generating all the links, zero pronouns are included in coreference chains. For every coreference chain, all zero pronouns will be removed before evaluation.

5.4 Experimental Results and Analysis

For fair comparison, all our experiments in this subsection have been conducted using the same experimental settings as our baseline system. When compared to our baseline system, all improvements are statistically significant ($p < 0.005$).

Table 8 lists the coreference resolution performance incorporating automatically detected zero pronouns. The results show that:

- Using automatically detected zero pronouns achieves better performance under all experimental settings. In particular, using automatic mentions, performance improves by 3.31%, 1.42%, and 2.34% in F-measure on the MUC, BCUBED, and CEAF evaluation metric, respectively. Using gold mention boundaries, automatic zero pronouns contribute 1.82% in average F-measure. Using gold mentions, the

		R	P	F
AM	Mention Detection	71.09	69.58	70.33
	MUC	55.06	64.91	59.58
	BCUBED	76.04	80.38	78.15
	CEAF	53.98	49.19	51.47
	Average			63.07
GMB	Mention Detection	82.44	70.10	75.77
	MUC	75.58	69.89	72.62
	BCUBED	76.35	87.27	81.45
	CEAF	65.17	52.31	58.04
	Average			70.70
GM	Mention Detection	84.31	100.00	91.49
	MUC	80.83	86.27	83.46
	BCUBED	74.18	92.74	82.43
	CEAF	69.91	62.29	65.88
	Average			77.26

Table 8: Performance of our Chinese coreference resolution system incorporating zero pronouns

contribution of zero pronouns is only 0.24% in average F-measure. This is because employing either gold mention boundaries or gold mentions improves parsing performance.

- Our system incorporating zero pronouns outperforms the three best systems in the CoNLL-2012 shared task when using automatic mentions or gold mention boundaries. Using gold mentions, our average F-measure is slightly lower than that of Chen and Ng (2012).[5]

Table 9 presents the contribution of our two methods of exploiting zero pronouns and the impact of gold-standard zero pronouns. We conclude that:

- Both the refined parser and refined example generation improve performance. While the refined parser improves the recall of mention detection and coreference resolution, refined example generation contributes more to precision. Combining these two methods further improves coreference resolution.

- There is a performance gap of 6.01%, 4.08%, and 3.19% in F-measure on the MUC, BCUBED, and CEAF evaluation metric, respectively, between the coreference resolution system with gold-standard zero pronouns and without zero pronouns. This suggests the usefulness of zero pronoun detection in Chinese coreference resolution.

- Our proposed methods incorporating automatic zero pronouns reduce the performance gap by about half. This shows the effectiveness of our proposed methods.

5.5 Discussion

Although the evaluation of the CoNLL-2012 shared task does not consider zero pronouns, we also evaluate the performance of zero pronoun resolution on the development data set (i.e., extracting all the resolved coreference links containing zero pronouns, acting as anaphor or antecedent, to conduct the evaluation independently). The results show that, for the correct anaphoric zero pronouns, the precision of our system is 94.76%. So viewing zero pronouns as a special kind of NP, zero pronouns can bridge salience and contribute to coreference resolution. In Example (1), the zero pronouns occurring in the second sentence help to bridge the coreferential relation between the mention "这个计划/this plan" in the last sentence and the mention "一个重建计划/a reconstruction plan" in the first sentence.

[5]Statistical significance testing cannot be conducted since their output files are not released.

6 Related Work

In the last decade, both manual rule-based approaches (Lee et al., 2011) and statistical approaches (Soon et al., 2001; Ng and Cardie, 2002; Fernandes et al., 2012) have been proposed for coreference resolution. Besides frequently used syntactic and semantic features, more linguistic features are exploited in recent work (Ponzetto and Strube, 2006; Ng, 2007; Versley, 2007). There is less research on Chinese coreference resolution compared to English.

Although zero pronouns are prevalent in Chinese, there is relatively little work on this topic. For Chinese zero pronoun resolution, representative work includes Converse (2006), Zhao and Ng (2007), and Kong and Zhou (2010).

For the use of zero pronouns, Chung and Gildea (2010) applied some extracted patterns to recover two types of empty elements (*PRO* and *pro*). Although the performance is still not satisfactory (e.g., 63.0 and 44.0 in F-measure for *PRO* and *pro* respectively), it nevertheless improves machine translation performance by 0.96 in BLEU score.

7 Conclusion

In this paper, we focus on exploiting one of the key characteristics of Chinese text, zero pronouns, to improve Chinese coreference resolution. In particular, a simplified semantic role labeling framework is proposed to detect zero pronouns effectively, and two effective methods are employed to incorporate zero pronouns into Chinese coreference resolution. Experiments on the CoNLL-2012 shared task show the effectiveness of our proposed approach. To the best of our knowledge, this is the first attempt at incorporating zero pronouns into Chinese coreference resolution.

	MD			MUC			BCUBED			CEAF			Avg
	R	P	F	R	P	F	R	P	F	R	P	F	
Baseline	65.26	67.20	66.22	51.64	61.82	56.27	73.40	80.38	76.73	53.16	45.66	49.13	60.71
+RP	72.01	66.24	69.00	55.02	61.47	58.07	77.83	78.97	78.40	50.40	49.81	50.10	62.19
+REG	65.92	70.02	67.91	49.98	66.27	56.98	73.64	83.45	78.24	51.12	47.44	49.21	61.48
+AZPs	71.09	69.58	70.33	55.06	64.91	59.58	76.04	80.38	78.15	53.98	49.19	51.47	63.07
+GZPs	72.18	70.59	71.38	58.61	66.45	62.28	78.79	82.94	80.81	54.12	50.63	52.32	65.14

Table 9: Contributions of the two methods of incorporating zero pronouns and the impact of gold zero pronouns (RP: refining parser using auto zero pronouns, REG: refining example generation using auto zero pronouns, AZPs: combining both RP and REG using auto zero pronouns, and GZPs: combining both RP and REG using gold zero pronouns)

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Proceedings of the Joint Conference on EMNLP and CoNLL – Shared Task*, pages 49–55.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Proceedings of the Joint Conference on EMNLP and CoNLL – Shared Task*, pages 56–63.

Tagyoung Chung and Daniel Gildea. 2010. Effects of empty categories on machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 636–645.

Susan Converse. 2006. *Pronominal Anaphora Resolution in Chinese*. Ph.D. thesis, University of Pennsylvania.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Joint Conference on EMNLP and CoNLL – Shared Task*, pages 41–48.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT-Press.

Young-Joo Kim. 2000. Subject/object drop in the acquisition of Korean: A cross-linguistic comparison. *Journal of East Asian Linguistics*, 9:325–351.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for Chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882–891.

Fang Kong, Guodong Zhou, and Qiaoming Zhu. 2009. Employing the centering theory in pronoun resolution from the semantic perspective. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 987–996.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Vincent Ng. 2007. Semantic class induction and coreference resolution. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 536–543.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 192–199.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Yannick Versley. 2007. Antecedent selection techniques for high-recall coreference resolution. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 496–505.

Bonnie Lynn Webber. 1978. *A Formal Approach to Discourse Anaphora*. Garland Press.

Nianwen Xue. 2008. Labeling Chinese predicates with semantic roles. *Computational Linguistics*, 34(2):225–255.

Yaqin Yang and Nianwen Xue. 2010. Chasing the ghost: Recovering empty categories in the Chinese Treebank. In *Coling 2010: Posters*, pages 1382–1390.

Bo Yuan, Qingcai Chen, Yang Xiang, Xiaolong Wang, Liping Ge, Zengjian Liu, Meng Liao, and Xianbo Si. 2012. A mixed deterministic model for coreference resolution. In *Proceedings of the Joint Conference on EMNLP and CoNLL – Shared Task*, pages 76–82.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 541–550.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83.

Joint Coreference Resolution and Named-Entity Linking with Multi-pass Sieves

Hannaneh Hajishirzi **Leila Zilles** **Daniel S. Weld** **Luke Zettlemoyer**
Department of Computer Science and Electrical Engineering
University of Washington
{hannaneh,lzilles,lsz,weld}@cs.washington.edu

Abstract

Many errors in coreference resolution come from semantic mismatches due to inadequate world knowledge. Errors in named-entity linking (NEL), on the other hand, are often caused by superficial modeling of entity context. This paper demonstrates that these two tasks are complementary. We introduce NECo, a new model for named entity linking and coreference resolution, which solves both problems jointly, reducing the errors made on each. NECo extends the Stanford deterministic coreference system by automatically linking mentions to Wikipedia and introducing new NEL-informed mention-merging sieves. Linking improves mention-detection and enables new semantic attributes to be incorporated from Freebase, while coreference provides better context modeling by propagating named-entity links within mention clusters. Experiments show consistent improvements across a number of datasets and experimental conditions, including over 11% reduction in MUC coreference error and nearly 21% reduction in F1 NEL error on ACE 2004 newswire data.

1 Introduction

Coreference resolution and named-entity linking are closely related problems, but have been largely studied in isolation. This paper demonstrates that they are complementary by introducing a simple joint model that improves performance on both tasks.

Coreference resolution is the task of determining when two textual mentions name the same individual. The biggest challenge in coreference resolution — accounting for 42% of errors in the state-of-the-art Stanford system — is the inability to reason effectively about background semantic knowledge (Lee et al., 2013). For example, consider the sentence in Figure 1. "President" refers to "Donald Tsang" and "the park" refers to "Hong Kong Disneyland," but automated algorithms typically lack the background knowledge to draw such inferences. Incorporating knowledge is challenging, and many efforts to do so have actually hurt performance, e.g. (Lee et al., 2011; Durrett and Klein, 2013).

[Michael Eisner]$_1$ and [Donald Tsang]$_2$ announced the grand opening of [[Hong Kong]$_3$ Disneyland]$_4$ yesterday. [Eisner]$_1$ thanked [the President]$_2$ and welcomed [fans]$_5$ to [the park]$_4$.

Figure 1: A text passage illustrating interactions between coreference resolution and NEL.

Named-entity linking (NEL) is the task of matching textual mentions to corresponding entities in a knowledge base, such as Wikipedia or Freebase. Such links provide rich sources of semantic knowledge about entity attributes — Freebase includes *president* as Tsang's title and `Disneyland` as having the attribute *park*. But NEL is itself a challenging problem, and finding the correct link requires disambiguating based on the mention string and often non-local contextual features. For example, "Michael Eisner" is relatively unambiguous but the isolated mention "Eisner" is more challenging. However, these mentions could be clustered with a coreference model, allowing for improved NEL through link propagation from the easier mentions.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 289–299,
Seattle, Washington, USA, 18-21 October 2013.

We present NECO, a new algorithm for jointly solving named entity linking and coreference resolution. Our work is related to that of Ratinov and Roth (2012), which also uses knowledge derived from an NEL system to improve coreference. However, NECO is the first joint model we know of, is purely deterministic with no learning phase, does automatic mention detection, and improves performance on both tasks.

NECO extends the Stanford's sieve-based model, in which a high recall mention detection phase is followed by a sequence of cluster merging operations ordered by decreasing precision (Raghunathan et al., 2010; Lee et al., 2013). At each step, it merges two clusters only if all available information about their respective entities is consistent. We use NEL to increase recall during the mention detection phase and introduce two new cluster-merging sieves, which compare the Freebase attributes of entities. NECO also improves NEL by initially favoring high precision linking results and then propagating links and attributes as clusters are formed.

In summary we make the following contributions:

- We introduce NECO, a novel, joint approach to solving coreference and NEL, demonstrating that these tasks are complementary by achieving joint error reduction.
- We present experiments showing improved performance at coreference resolution, given both gold and automatic mention detection: e.g., 6.2 point improvement in MUC recall on ACE 2004 newswire text and 3.1 point improvement in MUC precision the CoNLL 2011 test set.
- NECO also leads to better performance at named-entity linking, given both gold and automatic linking, improving F1 from 61.7% to 69.2% on a newly labeled test set.[1]

2 Background

We make use of existing models for coreference resolution and named entity linking.

[1]Our corpus and the source code for NECO can be downloaded from `https://www.cs.washington.edu/research-projects/nlp/neco`.

2.1 Coreference Resolution

Coreference resolution is the the task of identifying all text spans (called *mentions*) that refer to the same entity, forming mention clusters.

Stanford's Sieve Model is a state-of-the-art coreference resolver comprising a pipeline of "sieves" that merge coreferent mentions according to deterministic rules. Mentions are automatically predicted by selecting all noun phrases (NP), pronouns, and named entities. Each sieve either merges a cluster to its single best antecedent from a list of previous clusters, or declines to merge.

Higher precision sieves are applied earlier in the pipeline according to the following order, looking at different aspects of the text, including: (1) speaker identification, (2-3) exact and relaxed string matches between mentions, (4) precise constructs, including appositives, acronyms and demonyms, (5-9) different notions of strict and relaxed head matches between mentions, and finally (10) a number of syntactic and distance cues for pronoun resolution.

2.2 Named Entity Linking

Named-entity linking (NEL) is the task of identifying mentions in a text and linking them to the entity they name in a knowledge base, usually Wikipedia. NECO uses two existing NEL systems: GLOW (Ratinov et al., 2011) and WikipediaMiner (Milne and Witten, 2008).

WikipediaMiner links mentions based on a notion of semantic similarity to Wikipedia pages, considering all substrings up to a fixed length. Since there are often many possible links, it disambiguates by choosing the entity whose Wikipedia page is most semantically related to the nearby context of the mention. The semantic scoring function includes n-gram statistics and also counts shared links to other unambiguous mentions in the text.

GLOW finds mentions by selecting all the NPs and named entities in the text. Linking is framed as an integer linear programming optimization problem that takes into account using similar local constraints but also includes global constraints such as entity link co-occurrence.

Both systems return confidence values. To maintain high precision, NECO uses an ensemble of

- Let $Exemplar(c)$ be a representative mention of the cluster c, computed as defined below
- Let c_j be an antecedent cluster of c_i if c_j has a mention which is before the first mention of c_i
- Let $l(m)$ be a Wikipedia page linked to mention m or $\varnothing$ if there is no link
- Let $l(c)$ be a Wikipedia page linked to mention $Exemplar(c)$ or $\varnothing$ if there is no link

1. **Initialize Linked Mentions:**
   (a) Let $M_{NEL} = \{m_i \mid i = 1 \dots p\}$ be the NEL output mentions, m_i, each with a link $l(m_i)$
   (b) Let $M_{CR} = \{m_i \mid i = 1 \dots q\}$ be the mentions m_i from coreference mention detection
   (c) Let $M \leftarrow M_{CR} \cup M_{NEL}$ (Sec. 3.1)
   (d) Update entity links for all $m \in M$ and prune M (Sec. 3.2)
   (e) Extract attributes from Wikipedia and Freebase for all $m \in M$ (Sec. 3.3)
   (f) Let $C \leftarrow M$ be singleton mention clusters where $Exemplar(c_i) = m_i$, $l(c_i) = l(m_i)$
2. **Merge Clusters:** For every sieve S (including NEL sieves, Sec. 3.6) and cluster $c_i \in C$
   (a) For every cluster $c_j, j = [i-1 \dots 1]$ (traverse the preceding clusters in reverse order)
      i. **NEL constraints**: Prevent merge if $l(c_i) \neq l(c_j)$ (Sec. 3.4)
      ii. If all rules of sieve S are satisfied for clusters c_i and c_j
         A. $c_k \leftarrow Merge(c_i, c_j)$, including entity link and attribute updates (Sec. 3.5)
         B. $C \leftarrow C \cup \{c_k\} \setminus \{c_i, c_j\}$
3. **Output:** Coreference clusters C and linked Wikipedia pages $l(c_i) \forall c_i \in C$

Figure 2: NECO: A joint algorithm for named-entity linking and coreference resolution.

GLOW and WikipediaMiner, selecting only high confidence links.

3 Joint Coreference and Linking

We introduce a joint model for coreference resolution and NEL. Building on the Stanford sieve architecture, our algorithm incrementally constructs clusters of mentions using deterministic coreference rules under NEL constraints.

Figure 2 presents the complete algorithm. The input to NECO is a document and the output is a set C of coreference clusters, with links $l(c)$ to Wikipedia pages for a subset of the clusters $c \in C$. Step 1 detects mentions, merging the outputs of the base systems (Sec. 3.1). Step 2 repeatedly merges coreference clusters, while ensuring that NEL constraints (Sec. 3.4) are satisfied. It uses the original Stanford sieves and also two new NEL-informed sieves (Sec. 3.6). NEL links are propagated to new clusters as they are formed (Sec. 3.5).

3.1 Mention Detection

In Steps 1(a-c) in Fig. 2, NECO combines mentions from the base coreference and NEL systems.

Let M_{CR} be the set of mentions returned by using Stanford's rule-based mention detection algorithm (Lee et al., 2013). Let M_{NEL} be the set of mentions output by the two NEL systems. NECO creates an initial set of mentions, M, by taking the union of all the mentions in M_{NEL} and M_{CR}. In practice, taking the union increases diversity in the mention pool. For example, it is often the case that M_{NEL} will include sub-phrases such as "Suharto" when they are part of a larger mention "ex-dictator Suharto" that is detected in M_{CR}.

3.2 Mention Entity Links and Pruning

Step 1(d) in Fig. 2 assigns Wikipedia links to a subset of the detected mentions.

For mentions m output by the base NEL systems, we assign an *exact* link $l(m)$ if the entire mention span is linked. Mentions m' that differ from an exact linked mention m by only a pre- or post-fix stop word are similarly assigned exact links $l(m') = l(m)$. For example, the mention "the president" will be assigned the same link as "president" but "The governor of Alaska Sarah Palin" would not be assigned an exact link to `Sarah Palin`.

For mentions m' that do not receive an exact link, we assign a *head* link $h(m')$ if the head word[2] m has been linked, by setting $h(m') = l(m)$. For instance, the head link for the mention "President Clinton" (with "Clinton" as head word) will be the Wikipedia title of `Bill Clinton`. We use head links for the Relaxed NEL sieve (Sec. 3.6).

Next, we define $L(m)$ to be the set con-

[2] A head word is assigned to every mention with the Stanford parser head finding rules (Klein and Manning, 2003).

country	*president*	*city*	*area*
company	*state*	*region*	*location*
place	*agency*	*power*	*unit*
body	*market*	*park*	*province*
manager	*organization*	*owner*	*trial*
site	*prosecutor*	*attorney*	*county*
senator	*stadium*	*network*	*building*
attraction	*government*	*department*	*person*
origin	*plant*	*airport*	*kingdom*
capital	*operation*	*author*	*period*
nominee	*candidate*	*film*	*venue*

Figure 3: The most commonly used fine-grained attributes from Freebase and Wikipedia (out of over 500 total attributes).

taining $l(m)$ and $l(m')$ for all sub-phrases m' of m. We add the sub-phrase links only if their confidence is higher than the confidence for $l(m)$. For instance, assuming appropriate confidence values, $L(m)$ would include the pages for {`List of governors of Alaska`, `Alaska`, `Sarah Palin`} given the mention "The governor of Alaska Sarah Palin." We will use $L(m)$ for NEL constraints and filtering (Sec. 3.4).

After updating the entity links for all mentions, NECo prunes spurious mentions that begin or end with a stop word where the remaining subexpression of the mention exists in M. It also removes time expressions and numbers from M if they are not included in M_{NEL}.

3.3 Mention Attributes

Step 1(e) in Fig. 2 also assigns attributes for a mention m linked to Wikipedia page $l(m)$, at both *coarse* and *fine-grained* levels, based on information from the Freebase entry corresponding to exact link $l(m)$ or head link $h(m)$.

The coarse attributes include gender, type, and NER classes such as PERSON, LOCATION, and ORGANIZATION. These attributes are part of the original Stanford coreference system and are used to avoid merging conflicting clusters. We use the Freebase values for these attributes when available. For instance, if the linked entity contains the Freebase type *location* or *organization*, we include the coarse type to LOCATION or ORGANIZATION respectively. In order to account for both links to specific people (Barack Obama) and generic links to positions held by people (President), we include the type PERSON if the linked entity has any of the Freebase types *person*, *job_title*, or *government_office_or_title*. If no coarse Freebase types are available for an attribute, we default to predicted NER classes.

We add fine-grained attributes from Freebase and Wikipedia by importing additional type information. We use all of the Freebase *notable types*, a set of hundreds of commonly used Freebase types, ranging from *us_president* to *tropical_cyclone* and *synthpop_album*. We also include all of the Wikipedia categories, on average six per entity. For example, the mention "Indonesia" is assigned fine-grained attributes such as *book_subject*, *military_power*, and *olympic_participating_country*. Since many of these fine-grained attributes are extremely specific, we use the last word of each attribute to define an additional fine-grained attribute (see Fig. 3). These fine-grained attributes are used in the Relaxed NEL sieve (Sec. 3.6).

3.4 NEL Constraints

While applying sieves to merge clusters in Figure 2 Step 2(a), NECo uses NEL constraints to eliminate some otherwise acceptable merges.

We avoid merging inconsistent clusters that link to different entities. Clusters c_i and c_j are inconsistent if both are linked (i.e., both clusters have non-null entity assignments) and $l(c_i) \neq l(c_j)$ or $h(c_i) \neq h(c_j)$. Also, in order to consider an antecedent cluster c as a merge candidate, we require a pair of entities in the set of linked entities $L(c)$ to be related to one another in Freebase. Two entities are related in Freebase if they both appear in a relation; for example, `Bill Clinton` and `Arkansas` are related because Bill Clinton has a "governor-of" relation with Arkansas.

3.5 Merging Clusters and Update Entity Links

When two clusters c_i and c_j are merged to form a new cluster c_k, the entity link information $L(c_k)$, $l(c_k)$, and $h(c_k)$ must be updated (Step 2 of Fig. 2). We set $L(c_k)$ to the union of the linked entities found in $l(c_i)$ and $l(c_j)$ and merge coarse attributes at this point.

In order to set the exact and head entity links $l(c_k)$ and $h(c_k)$, we use the exemplar mention

$Exemplar(c_k)$ that denotes the most representative mention of the cluster. $Exemplar(c)$ is selected according to a set of rules in the Stanford system, based on textual position and mention type (proper noun vs. common). We augment this function by considering information from exact and head entity links as well. Mentions appearing earlier in text, proper mentions, and mentions that have exact or head named-entity links are preferred to those which do not. Given exemplars, we set $l(c_k) = l(Exemplar(c_k))$ and $h(c_k) = h(Exemplar(c_k))$.

3.6 NEL Sieves

Finally, we introduce two new sieves that use NEL information at the beginning and end of the Stanford sieves pipeline in the merging stage (Step 2 of Fig. 2).

Exact NEL sieve The Exact NEL sieve merges two clusters c_i and c_j if both are linked and their links match, $l(c_i) = l(c_j)$. For example, all mentions that have been linked to `Barack Obama` will become members of the same coreference cluster. Because the Exact NEL sieve has high precision, we place it at the very beginning of the pipeline.

Relaxed NEL sieve The Relaxed NEL sieve uses fine-grained attributes of the linked mentions to merge proper nouns with common nouns when they share attributes. For example, this sieve is able to merge the proper mention "Disneyland" with the "the mysterious park", because *park* is one of the fine-grained attributes assigned to `Disneyland`.

More formally, let $m_i = Exemplar(c_i)$ and $m_j = Exemplar(c_j)$. For every common noun mention m_i, we merge c_i with an antecedent cluster c_j if (1) m_j is a linked proper noun, (2) if m_i or the title of its linked Wikipedia page is in the list of fine-grained attributes of m_j, or (3) if $h(m_j)$ is related to the head link $h(m_i)$ according to Freebase as defined above.

Because this sieve has low precision, we only allow merges between mentions that have a maximum distance of three sentences between one another. We add the Relaxed NEL sieve near the end of the pipeline, just before pronoun resolution.

4 Experimental Setup

Core Components and Baselines The Stanford sieve-based coreference system (Lee et al., 2013), the GLOW NEL system (Ratinov et al., 2011), and WikipediaMiner (Milne and Witten, 2008) provide core functionality for our joint model, and are also the state-of-the-art baselines against which we measure performance.

Parameter Settings Based on performance on the development set, we set the GLOW's confidence parameter to 1.0 and WikipediaMiner's to 0.4 to assure high-precision NEL. We also optimized for the set of fine-grained attributes to import from Wikipedia and Freebase, and the best way to incorporate the NEL constraints into the sieve architecture.

Datasets We report results on the following three datasets: ACE2004-NWIRE, CONLL2011, and ACE2004-NWIRE-NEL. ACE2004-NWIRE, the newswire subset of the ACE 2004 corpus (NIST, 2004), includes 128 documents. The CONLL2011 coreference dataset includes text from five different domains: broadcast conversation (BC), broadcast news (BN), magazine (MZ), newswire (NW), and web data (WB) (Pradhan et al., 2011). The broadcast conversation and broadcast news domains consist of transcripts, whereas magazine and newswire contain more standard written text. The development data includes 303 documents and the test data includes 322 documents.

We created ACE2004-NWIRE-NEL by taking a subset of ACE2004-NWIRE and annotating with gold-standard entity links. We segment and link all the expressions in text that refer to Wikipedia pages, allowing for nested linking. For instance, both the phrase "Hong Kong Disneyland," and the sub-phrase "Hong Kong" are linked. This dataset includes 12 documents and 350 linked entities.

Metrics We evaluate our system using MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and pairwise scores. MUC is a link-based metric which measures how many clusters need to be merged to cover the gold clusters and favors larger clusters; B^3 computes the proportion of intersection between predicted and gold clusters for every mention and favors singletons (Recasens and Hovy, 2010). We computed the scores using the Stanford

Method	MUC			B^3			Pairwise		
	P	R	F1	P	R	F1	P	R	F1
Stanford Sieves	39.9	46.2	42.8	67.9	71.8	69.8	44.2	29.7	35.6
NECo	**46.8**	**52.5**	**49.5**	70.4	**72.6**	**71.5**	**51.5**	**34.6**	**41.4**
No NEL Mentions	46.1	48.3	47.2	71.4	70.0	70.9	49.7	30.9	38.1
No Mention Pruning	43.6	45.6	44.6	70.5	69.9	70.2	46.2	29.4	35.9
No Attributes	45.9	47.4	46.6	**71.8**	69.7	70.7	48.6	27.0	34.7
No Constraints	42.3	49.3	45.5	68.3	72.3	70.2	44.2	28.6	34.7

Table 1: Coreference results on ACE2004-NWIRE with predicted mentions and automatic linking.

coreference software for ACE2004 and using the CoNLL scorer for the CoNLL 2011 dataset.

5 Experimental Results

We first look at NECo's performance at coreference resolution and then evaluate its ability at NEL.

5.1 Coref. Results with Predicted Mentions

Overall System Performance on ACE Data Table 1 shows NECo's performance at coreference resolution on ACE-2004 compared to the Stanford sieve implementation (Lee et al., 2013). The table shows that NECo has both significantly improved precision and recall compared to the Stanford baseline, across all metrics. We generally observe larger gains in MUC due to better mention detection and the Relaxed NEL Sieve.

Contribution of System Components Table 1 also details the performance of four variants of our system that ablate various components and features. Specifically, we consider the following cases:

- **No NEL Mentions:** We discard additional mentions, M_{NEL}, provided by NEL (Sec. 3.1). This increases B^3 precision at the expense of recall. Inspection shows that some of the errors introduced by M_{NEL} are actually due to correctly linked entities that were not annotated as mentions in the dataset, but also some improperly linked mentions.
- **No Mention Pruning:** We disable the initial step of updating mention boundaries and removing spurious mentions (Sec. 3.2). As expected, removing this step drops precision and recall significantly, even compared to the No NEL Mentions variant.
- **No Attributes:** Ablating coarse and fine-grained attributes (Sec. 3.3) drops F1 and recall measures across all metrics. To understand this effect, note that NECo uses attributes in two different settings. Updating coarse attributes tends to increase precision because it prevents dangerous merges, such as merging "Staples" with the mention "it" in a situation when "Staples" refers to the person entity `Todd Staples`. Fine-grained attributes also help with recall, when merging a specific name of an entity with a mention that uses a more general term; for instance, "Hong Kong Disneyland" can be merged with "the mysterious park" because "park" is a fine-grained attribute for `Disneyland`. However, when fine-grained attributes are used, precision sometimes drops (e.g., when "president" might merge with "Bush" when it should really merge with "Clinton").
- **No NEL Constraints:** Removing these constraints (Sec. 3.4) drops precision dramatically leading to drop in F1. In the case of incorrect linking, however, NEL constraints can affect recall. For instance, NEL constraints might prevent merging "Staples" with "Todd Staples" if the former were linked to the company and the latter to the politician.

Overall System Performance on CoNLL Data We also compare our full system (with added NEL sieves, constraints, and mention pruning[3]) with the Stanford sieve coreference system on CoNLL data

[3]Due to CoNLL annotation guidelines, a named entity is added to the mention list if it is *not* inside a larger mention with an exact named entity link.

Category: Method	MUC			B^3		
	P	R	F1	P	R	F1
BC: NECo	**62.1**	64.7	**63.4**	**69.8**	57.8	**63.2**
BC: Stanford Sieves	60.9	**65.0**	62.9	69.2	**58.0**	63.1
BN: NECo	**69.3**	**59.4**	**64.0**	78.8	**60.8**	**68.6**
BN: Stanford Sieves	68.0	58.9	63.1	**79.0**	60.2	68.3
MZ: NECo	**67.6**	62.9	**65.2**	**78.4**	61.1	68.7
MZ: Stanford Sieves	66.0	**63.4**	64.9	77.9	**61.5**	68.7
NW: NECo	**62.0**	**54.5**	**58.0**	74.9	**57.4**	**65.0**
NW: Stanford Sieves	60.0	54.2	56.9	**75.3**	57.0	64.9

Table 3: Coreference results on the individual categories of CoNLL 2011 development data. (BC=broadcast conversation, BN=broadcast news, MZ=magazine, NW=newswire)

Method	MUC			B^3		
	P	R	F1	P	R	F1
Development Data						
NECo	**64.1**$^+$	**59.4**	**61.7**$^+$	74.7	**58.7**	**65.7**
Stanford	62.7	59.0	60.8	**74.8**	58.3	65.6
NECo*	**56.4**$^+$	50.0	**53.0**$^+$	**72.6**	**51.6**	**60.3**
Stanford*	53.5	50.0	51.6	71.8	51.3	59.9
Test Data						
NECo	**61.2**$^+$	58.4	**59.8**$^+$	**72.2**	**56.4**	**63.3**
Stanford	59.2	**58.8**	59.0	71.3	56.1	62.8
NECo*	**55.1**$^+$	51.7	**53.3**$^+$	**70.0**	50.8	**58.8**
Stanford*	52.0	**52.3**$^+$	52.1	68.9	50.8	58.5

Table 2: Coreference results on CoNLL 2011 development and test data, using predicted mentions. Rows denoted with * indicate runs using the fully automated Stanford CoreNLP pipeline rather than the predicted annotations provided with the CoNLL data. Given the relatively close results, we ran the Mann-Whitney U test for this table; values with the $^+$ superscript are significant with $p < 0.05$.

(Table 2). We ran NECo and the baseline in two settings: in the first, we use the standard predicted annotations (for POS, parses, NER, and speaker tags) provided with the CoNLL data, and in the second, we use the automated Stanford CoreNLP pipeline to predict this information. On both the development and test sets, we gain about 1 point in MUC F1 as well as a smaller improvement in B^3. Closer inspection indicates that our system increases precision primarily due to mention pruning and NEL constraints. Due to the differences in mention annotation guidelines between ACE and CoNLL, performance on ACE benefits more from improved mention detection from NEL. Moreover, the ACE corpus is all newswire text, which contains more entities that can benefit from linking. CoNLL, on the other hand, contains a wider variety of texts, some of which do not mention many named entities in Wikipedia.

To examine the performance of our system on the different domains covered by the CoNLL data, we also test our system on each domain separately (Table 3). We found NEL provided the biggest improvement for the news domains, broadcast news (BN) and newswire (NW). These domains especially benefit from the improved mention detection and pruning provided by NEL, and strong linking benefitted both precision and recall in these domains. We found that the magazine (MZ) section of the corpus benefited the least from NEL, as there were relatively few entities that our NEL systems were able to connect to Wikipedia.

5.2 Coreference Results with Gold Linking

Some of the errors introduced in our system are due to incorrect or incomplete links discovered by the automatic linking system. To assess the effect of NEL performance on NECo, we tested on a portion of ACE2004-NWIRE dataset for which we hand-labeled correct links for the gold and predicted mentions. "NECo + Gold NEL" denotes a version of our system which uses gold links instead of those predicted by NEL. As shown in Table 4, gold linking significantly improves the performance of our system across all measures. This suggests that further work to improve automatic NEL may have substantial reward.

Gold linking improves precision for two main rea-

Method	MUC			B^3			Pairwise		
	P	R	F1	P	R	F1	P	R	F1
Gold Mentions									
NECo + Gold NEL	**85.8**	**75.5**	**80.3**	**91.4**	**81.2**	**86.0**	89.1	**68.0**	**77.1**
NECo	84.6	74.0	78.9	90.5	80.4	85.2	83.9	66.0	73.9
Stanford Sieves	84.5	72.2	77.8	89.9	77.7	83.4	**89.9**	57.3	68.1
Predicted Mentions									
NECo + Gold NEL	**56.4**	**58.8**	**57.5**	**78.2**	**78.3**	**78.3**	**68.0**	**54.3**	**60.4**
NECo	51.3	53.5	52.4	76.5	76.4	76.5	61.2	45.6	52.2
Stanford Sieves	43.9	46.4	45.1	74.4	74.2	74.3	51.3	36.1	42.4

Table 4: Coreference results on ACE2004-NWIRE-NEL with gold and predicted mentions and gold or automatic linking.

Method	MUC			B^3			Pairwise		
	P	R	F1	P	R	F1	P	R	F1
NECo	**85.0**	**76.6**	**80.6**	**87.6**	**76.4**	**81.6**	79.3	**56.1**	**65.8**
Stanford Sieves	84.6	75.1	79.6	87.3	74.1	80.2	**79.4**	50.1	61.4
Haghighi and Klein (2009)	77.0	75.9	76.5	79.4	74.5	76.9	66.9	49.2	56.7
Poon and Domingos (2008)	71.3	70.5	70.9	-	-	-	62.6	38.9	48.0
Finkel and Manning (2008)	78.7	58.5	67.1	86.8	65.2	74.5	76.1	44.2	55.9

Table 5: Coreference results on ACE2004-NWIRE with gold mentions and automatic linking.

sons. First, it reduces the coreference errors caused by incorrect NEL links. For instance, gold linking replaces the erroneous link generated by our NEL systems for "Nasser al-Kidwa" to the correct Wikipedia entity. As another example, two mentions of "Rutgers" will not be merged if one links to the university and the other links to their football team. Second, gold linking leads to better mention detection and better linked mentions. For instance, under gold linking, the whole mention, "The governor of Alaska, Sarah Palin," is linked to the politician, while automatic linking systems only link the substring containing her name, "Sarah Palin." Still, gold NEL cannot compensate for all coreference errors in cases of generic or unlinked entities.

5.3 Coreference Results with Gold Mentions

Many of the previous papers evaluate coreference resolution assuming gold mentions so we also run under that condition (Table 5) using ACE2004-NWIRE data. As the table shows, with gold mentions our system outperforms Haghighi and Klein (2009), Poon and Domingos (2008), Finkel and Manning (2008) and the Stanford sieve algorithm across all metrics. Our method shows a relatively smaller gain in precision, because this condition adds no benefit to our technique of using NEL information for pruning mentions.

5.4 Improving Named Entity Linking

While our previous experiments show that named-entity linking can improve coreference resolution, we now address the question of whether coreference techniques can help NEL. We compare NECo with a baseline ensemble[4] composed of GLOW (Ratinov et al., 2011) and WikipediaMiner (Milne and Witten, 2008) on our ACE2004-NWIRE-NEL dataset (Table 6). Our system gains about 8% in absolute recall and 5% in absolute precision. For instance, our system correctly adds links from "Bullock" to the entity `Sandra Bullock` because coreference resolution merges two mentions. In another example, it correctly links "company" to `Nokia`. Overall, there is a 21% relative reduction in F1 error.

[4]We take the union of all the links returned by GLOW and WikipediaMiner, but if they link a mention to two different entities, we use only the output of WikipediaMiner.

Method	F1	Precision	Recall
NECo	**70.6**	**72.0**	**69.2**
Baseline NEL	64.4	67.4	61.7

Table 6: NEL performance of our system and the ensemble baseline linker on ACE2004-NWIRE-NEL.

5.5 Error Analysis

We analyzed 90 precision and recall errors and present our findings in Table 7. Spurious mentions accounted for the majority of non-semantic errors. Despite the improvements that come from NEL, a large portion of coreference errors can still be attributed to incomplete semantic information, including precision errors caused by incorrect linking. For instance, the mention "Disney" sometimes refers to the company, and other times refers to the amusement park; however, the NEL systems we used had difficulty disambiguating these cases, and NECo often incorrectly merges such mentions. Overly general fine-grained attributes caused precision errors in cases where many proper noun mentions were potential antecedents for a common noun. Although attributes such as *country* are useful for resolving a generic "country" mention, this information is insufficient when two distinct mentions such as "China" and "Russia" both have the *country* attribute.

However, many recall errors are also caused by the lack of fine-grained attributes. Finding the ideal set of fine-grained attributes remains an open problem.

6 Related Work

Coreference resolution has a fifty year history which defies brief summarization; see Ng (2010) for a recent survey. Section 2.1 described the Stanford multi-pass sieve algorithm, which is the foundation for NECo.

Earlier coreference resolution systems used shallow semantics and pioneered knowledge extraction from online encyclopedias (Ponzetto and Strube, 2006; Daumé III and Marcu, 2005; Ng, 2007). Some recent work shows improvement in coreference resolution by incorporating semantic information from Web-scale structured knowledge bases. Haghighi and Klein (2009) use a rule-based system to extract fine-grained attributes for mentions by analyzing precise constructs (e.g., appositives) in Wikipedia articles. Subsequently, Haghighi and Klein (2010) used a generative approach to learn entity types from an initial list of unambiguous mention types. Bansal and Klein (2012) use statistical analysis of Web n-gram features including lexical relations.

Rahman and Ng (2011) use YAGO to extract type relations for all mentions. These methods incorporate knowledge about all possible meanings of a mention. If a mention has multiple meanings, extraneous information might be associated with it. Zheng et al. (2013) use a ranked list of candidate entities for each mention and maintain the ranked list when mentions are merged. Unlike previous work, our method relies on NEL systems to disambiguate possible meanings of a mention and capture high-precision semantic knowledge from Wikipedia categories and Freebase notable types.

Ratinov and Roth (2012) investigated using NEL to improve coreference resolution, but did not consider a joint approach. They extracted attributes from Wikipedia categories and used them as features in a learned mention-pair model, but did not do mention detection. Unfortunately, it is difficult to compare directly to the results of both systems, since they reported results on portions of ACE and CoNLL datasets using gold mentions. However, our approach provides independent evidence for the benefit of NEL, and joint modeling in particular, since it outperforms the state-of-the-art Stanford sieve system (winner of the CoNLL 2011 shared task (Pradhan et al., 2011)) and other recent comparable approaches on benchmark datasets.

Our work also builds on a long trajectory of work in named entity resolution stemming from SemTag (Dill et al., 2003). Section 2.2 discussed GLOW and WikipediaMiner (Ratinov et al., 2011; Milne and Witten, 2008). Kulkarni et al. (2009) present an elegant collective disambiguation model, but do not exploit the syntactic nuances gleaned by within-document coreference resolution. Hachey et al. (2013) provide an insightful summary and evaluation of different approaches to NEL.

7 Conclusions

Observing that existing coreference resolution and named-entity linking have complementary strengths

Error Type	Percentage	Example
Extra mentions	31.1	**The other thing Paula really important** is that they talk a lot about the fact ...
Pronoun	27.7	However , [*all 3 women gymnasts , taking part in the internationals for the first time*], performed well , because they had strong events and their movements had difficulty .
Contextual semantic	16.6	[*The Chinese side*] hopes that each party concerned continues to make constructive efforts to ...Considering the requirements of the Korean side , ... the Chinese government decided to ...
NEL semantic	13.3	The most important thing about **Disney** is that it is a global brand. ... The subway to Disney has already been constructed.
Attributes	11.1	The **Hong Kong** government turned over to Disney Corporation [*200 hectares of land ...*]. ... this area has become a prohibited zone in Hong Kong.

Table 7: Examples of different error categories and the relative frequency of each. For every example, the mention to be resolved is underlined, and the correct antecedent is *italicized*. For precision errors, the wrongly merged mention is **bolded**. For recall errors, the missed mention is surrounded by [brackets].

and weaknesses, we present a joint approach. We introduce NECO, a novel algorithm which solves the problems *jointly*, demonstrating improved performance on both tasks.

We envision several ways to improve the joint model. While the current implementation of NECO only introduces NEL once, we could also integrate predictions with different levels of confidence into different sieves. It would be interesting to more tightly integrate the NEL system so it operates on clusters rather than individual mentions — after each sieve merges an unlinked cluster, the algorithm would retry NEL with the new context information. NECO uses a relatively modest number of Freebase attributes. While using more semantic knowledge holds the promise of increased recall, the challenge is maintaining precision. Finally, we would also like to explore the extent to which a joint probabilistic model (e.g., (Durrett and Klein, 2013)) might be used to learn how to best make this tradeoff.

8 Acknowledgements

The research was supported in part by grants from DARPA under the DEFT program through the AFRL (FA8750-13-2-0019) and the CSSG (N11AP20020), the ONR (N00014-12-1-0211), and the NSF (IIS-1115966). Support was also provided by a gift from Google, an NSF Graduate Research Fellowship, and the WRF / TJ Cable Professorship. The authors thank Greg Durrett, Heeyoung Lee, Mitchell Koch, Xiao Ling, Mark Yatskar, Kenton Lee, Eunsol Choi, Gabriel Schubiner, Nicholas FitzGerald, Tom Kwiatkowski, and the anonymous reviewers for helpful comments and feedback on the work.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*.

Mohit Bansal and Dan Klein. 2012. Coreference semantics from web features. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. 2003. SemTag and Seeker: bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th International Conference on World Wide Web*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with Wikipedia. *Artificial Intelligence Journal*, 194.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of Wikipedia entities in Web text. In *Proceedings of the 2009 Conference on Knowledge Discovery and Data Mining*.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Conference on Computational Natural Language Learning*.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

Dan Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proceedings of the ACM Conference on Information and Knowledge Management*.

Vincent Ng. 2007. Shallow semantics for coreference resolution. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

NIST. 2004. The ACE 2004 evaluation planXPToolkit architecture.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, Wordnet and Wikipedia for coreference resolution. In *Proceedings of the North American Association for Natural Language Processing on Human Language Technologies*.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 shared task: modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Lev Ratinov and Dan Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Marta Recasens and Eduard Hovy. 2010. Coreference resolution across corpora: languages, coding schemes, and preprocessing information. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message Understanding*.

Jiaping Zheng, Luke Vilnis, Sameer Singh, Jinho D. Choi, and Andrew McCallum. 2013. Dynamic knowledge-base alignment for coreference resolution. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Interpreting Anaphoric Shell Nouns using Antecedents of Cataphoric Shell Nouns as Training Data

Varada Kolhatkar
Department of Computer Science
University of Toronto
varada@cs.toronto.edu

Heike Zinsmeister
Institut für Germanistik
Universität Hamburg
heike.zinsmeister@uni-hamburg.de

Graeme Hirst
Department of Computer Science
University of Toronto
gh@cs.toronto.edu

Abstract

Interpreting *anaphoric shell nouns* (ASNs) such as *this issue* and *this fact* is essential to understanding virtually any substantial natural language text. One obstacle in developing methods for automatically interpreting ASNs is the lack of annotated data. We tackle this challenge by exploiting *cataphoric shell nouns* (CSNs) whose construction makes them particularly easy to interpret (e.g., *the fact that X*). We propose an approach that uses automatically extracted antecedents of CSNs as training data to interpret ASNs. We achieve precisions in the range of 0.35 (baseline = 0.21) to 0.72 (baseline = 0.44), depending upon the shell noun.

1 Introduction

Anaphors such as *this fact* and *this issue* encapsulate complex abstract entities such as propositions, facts, and events. An example is shown below.

(1) Here is another bit of advice: Environmental Defense, a national advocacy group, notes **that "Mowing the lawn with a gas mower produces as much pollution in half an hour as driving a car 172 miles."** **This fact** may help to explain the recent surge in the sales of the good old-fashioned push mowers or the battery-powered mowers.

Here, the anaphor *this fact* is interpreted with the help of the clausal antecedent marked in bold. The antecedent here is complex because it involves a number of entities and events (e.g., mowing the lawn, a gas mower) and relationships between them, and is abstract because the antecedent itself is not a purely physical entity.

The distinguishing property of these anaphors is that they contain semantically rich abstract nouns (e.g., *fact* in (1)) which *characterize* and *label* their corresponding antecedents. Linguists and philosophers have studied such abstract nouns for decades (Vendler, 1968; Halliday and Hasan, 1976; Francis, 1986; Ivanic, 1991; Asher, 1993). Our work is inspired by one such study, namely that of Schmid (2000). Following Schmid, we refer to these abstract nouns as *shell nouns*, as they serve as conceptual *shells* for complex chunks of information. Accordingly, we refer to the anaphoric occurrences of shell nouns (e.g., *this fact* in (1)) as *anaphoric shell nouns* (ASNs).

An important reason for studying ASNs is their ubiquity in all kinds of text. Schmid (2000) observed that shell nouns such as *fact*, *idea*, *point*, and *problem* were among the 100 most frequently occurring nouns in a corpus of 225 million words of British English. Moreover, ASNs can play several roles in organizing a discourse such as encapsulation of complex information, cohesion, and topic boundary marking. So correct interpretation of ASNs can be an important step for correct interpretation of a discourse, and in a number of NLP applications such as text summarization, information extraction, and non-factoid question answering.

Despite their importance, ASNs have not received much attention in Computational Linguistics, and research in this field remains in its earliest stages. At

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 300–310,
Seattle, Washington, USA, 18-21 October 2013.

present, the major obstacle is that there is very little annotated data available that could be used to train a supervised machine learning system for robustly interpreting these anaphors, and manual annotation is an expensive and time-consuming task.

We tackle this challenge by exploiting a category of examples, as shown in (2), whose construction is particularly easy to interpret.

(2) Congress has focused almost solely on **the fact that special education is expensive – and that it takes away money from regular education**.

Here, in contrast with (1), *the fact* is not anaphoric in the traditional sense, but is an easy case of a forward-looking anaphor — a cataphor. While the resolution process of *this fact* in (1) is quite challenging as it requires the use of semantics and world knowledge, it is fairly easy to interpret *the fact* in (2) based on the syntactic structure alone. We refer to these easy-to-interpret cataphoric occurrences of shell nouns as *cataphoric shell nouns* (CSNs). The interpretation of both ASNs and CSNs will be referred to as *antecedent*.[1] The antecedent of *the fact* in (2) is given in the post-nominal *that* clause. We use the term *shell concept* to refer to the general *notion* of a shell noun, i.e., the semantic type of the antecedent. For example, the notion of an *issue* is an important problem which requires a solution.

In this work, we propose an approach to interpret ASNs that exploits *unlabelled* but easy-to-interpret CSN examples to extract characteristic features associated with the antecedent of different shell concepts. We evaluate our approach using crowdsourcing. Our results show that these unlabelled CSN examples provide useful linguistic properties that help in interpreting ASNs.

2 Related work

The resolution of anaphors to non-nominal antecedents has been well analyzed taking discourse structure and semantic types into account (Webber, 1991; Passonneau, 1989; Asher, 1993). Most work in machine anaphora resolution, however, is restricted to anaphora that involve nominal antecedents only (Poesio et al., 2011).

[1]Sadly, the more-logical term for the interpretation of a CSN, *succedent*, does not actually exist.

There are some notable exceptions which have tackled the challenge of interpreting non-nominal antecedents (Eckert and Strube, 2000; Strube and Müller, 2003; Byron, 2004; Müller, 2008). These approaches are limited as they either rely heavily on domain-specific syntactic and semantic annotation or prepossessing, or mark only verbal proxies for non-nominal antecedents.

Recently, Kolhatkar and Hirst (2012) presented a machine-learning based resolution system for *this issue* anaphora, identifying full syntactic phrases as antecedents. Although they achieved promising results, their approach was limited in two respects. First, it focused on only one type of shell noun anaphora (*issue* anaphora). Second, their training data was restricted to MEDLINE abstracts in which *this issue* is used in a rather systematic way. Furthermore, their work is based on manually labelled ASN antecedents, whereas we use automatically identified CSN antecedents, which we interpret as explicitly expressed antecedents in comparison to the more implicitly expressed ASN antecedents.

Using explicitly expressed structure in the text to identify implicit structure is not new. The same idea has been applied before in computational linguistics. Marcu and Echihabi (2002) identified implicit discourse relations using explicit ones. Markert and Nissim (2005) used Hearst's (1992) explicit patterns to learn lexical semantic relations for NP-coreference and *other*-anaphora resolution from the web. Although our work focuses on a different topic, the methodology is in the same vein.

3 Hypothesis of this work

The hypothesis of this work is that CSN antecedents and ASN antecedents share some linguistic properties and hence linguistic knowledge encoded in CSN antecedents will help in interpreting ASNs. Accordingly, we examine which features present in CSN antecedents are relevant in interpreting ASNs.

The motivation and intuition behind this hypothesis is as follows. The antecedents of both ASNs and CSNs represent the corresponding shell concept. So are there any characteristic features associated with this shell concept? Do speakers of English follow certain patterns of syntactic shape or words, for instance, when they state facts, decisions,

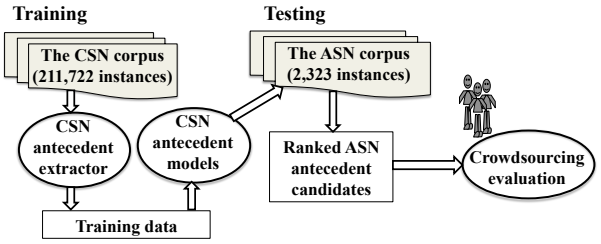


Figure 1: Overview of our approach

or issues? There is an abundance of data for CSN antecedents and if we are able to capture particular linguistic characteristic features associated with a shell concept using this data, we can use this information to interpret ASNs. For instance, example (2) demonstrates characteristic properties of antecedents of the shell noun *fact* including that (a) they are propositions and are generally expressed with clauses or sentences rather than noun phrases, and (b) they are generally expressed in the present tense. Observe that these properties also hold for the antecedent of *this fact* in example (1).

We test our hypothesis by building machine learning models that are trained on automatically extracted CSN antecedents and then applying these models to recover ASN antecedents. Figure 1 shows an overview of our methodology.

4 Background

Formal definition *Shell-nounhood* is a functional notion; it is defined by the *use* of an abstract noun rather than the inherent properties of the noun itself (Schmid, 2000). An abstract noun is a shell noun when the speaker decides to *use* it as a shell noun.

Shell noun categorization Schmid (2000) gives a list of 670 English nouns which are frequently used as shell nouns. He divides them into six broad semantic classes: *factual*, *linguistic*, *mental*, *modal*, *eventive*, and *circumstantial*. Table 1 shows this classification, along with example shell nouns for each category. For this work, we selected six frequently occurring shell nouns covering four of Schmid's six classes: *fact* and *reason* from factual, *issue* and *decision* from mental, *question* from linguistic, and *possibility* from modal. These shell nouns tend to have antecedents that lie within a single sentence. We excluded *eventive* and *circumstantial* classes because the shell nouns in these classes tend to have rather unclear and long antecedents.[2]

Class	Description	Examples
factual	states of affairs	***fact***, ***reason***
linguistic	linguistic acts	***question***
mental	ideas	***issue***, ***decision***
modal	judgements	***possibility***
eventive	events	*act*, *reaction*
circumstantial	situations	*situation*, *way*

Table 1: Schmid's classification of shell nouns. The nouns given in the *Example* column tend to occur frequently with the respective class. The shell nouns used in this work are shown in boldface.

Pattern	Example
N-to	Several people at the group said the **decision *to*** *write the letters* was not controversial internally.
N-be-to	The principal **reason is *to*** *create a representative government* rather than to select the most talented person.
N-that	Mr. Shoval left open the **possibility *that*** *Israel would move into other West Bank cities*.
N-be-that	The simple and reassuring **fact is *that*** *a future generation of leaders is seeking new challenges during challenging times*.
N-wh	There is now some **question *whether*** *the country was ever really in a recession*.
N-be-wh	Of course, the central, and probably insoluble, **issue is *whether*** *animal testing is cruel*.

Table 2: Easy-to-interpret CSN patterns given by Schmid (2000). In the *Example* column, the patterns are marked in boldface and the antecedents are marked in italics.

Shell noun patterns Schmid (2000) also provides a number of lexico-grammatical patterns for shell nouns. In Section 1, we noted two such patterns: *this-N* (*this fact* in example (1)) and *N-that* (*fact that* in example (2)). We also noted that CSNs with pattern *N-that* are fairly easy to interpret compared to the ASN pattern *this-N*. Table 2 shows some other easy-to-interpret CSN patterns given by Schmid. Generally, for all these patterns, the antecedent is

[2]These observations are based on an exploratory pilot annotation we carried out on sample data of 150 ASN instances.

quite easy to extract with a few predefined rules.

Shell antecedent properties Antecedents of CSNs and ASNs share some properties while they are distinguished by others. The distinguishing property is that CSNs, by their construction, have their antecedents in the same sentence, as shown in example (2). On the other hand, ASNs can have long-distance as well as short-distance antecedents.[3] The common properties are as follows. First, antecedents of both ASNs and CSNs represent the corresponding shell concept, e.g., the notion of a *fact* or an *issue*. Second, in both cases, the antecedents are complex abstract entities, which involve a number of entities and relationships between them. Finally, in both cases, there is no one-to-one correspondence between the syntactic type of an antecedent and semantic type of its referent (Webber, 1991). For instance, a semantic type such as *fact* can be expressed with different syntactic shapes such as a clause, a verb phrase, or a complex sentence. Conversely, a syntactic shape, such as a clause, can function as several semantic types, including *fact*, *proposition*, and *event*.

5 Training phase

As shown in Figure 1, the goal of the training phase is to build training data from CSNs and their antecedents and train models which can be used for resolving ASNs.

5.1 The CSN corpus

We automatically constructed a corpus, a subset of the New York times (NYT) corpus[4], which contains 211,722 sentences following CSN patterns from Table 2. We considered part-of-speech information[5] while looking for the patterns. For instance, instead of the pattern *N-that*, we actually looked for {shell_noun_*NN that_IN*}.

[3] In our annotated sample data, we observed ASN antecedents as close as the same sentence and as far as 7 sentences away from the anaphor.

[4] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2008T19

[5] http://nlp.stanford.edu/software/tagger.shtml

5.2 Antecedent extractor for CSNs

The goal of the antecedent extractor is to create automatically labelled CSN antecedent data. Recall that antecedents of CSNs can be extracted using simple predefined rules that are based on the syntactic structure alone. For instance, the antecedent extraction rule for example (2) would be: if the example follows the pattern *fact-that*, extract the post-nominal *that* clause as the antecedent. To come up with a list of such extraction rules, we systematically analyzed a sample of examples (about 20 examples) of each pattern for each shell noun. Table 3 summarizes the resulting antecedent extraction rules.

The actual antecedent extraction works as follows. First, we parsed the examples from the CSN corpus using the Stanford parser.[6] Then for each example, we applied rules from Table 3 depending on the shell noun and the pattern it follows to extract an appropriate syntactic constituent as the CSN antecedent. For instance, for the noun *fact* following the *N-that* pattern, as in example (2), we first looked for the NP constituent containing the shell noun *fact*, and then extracted the sentential constituent following the NP constituent as the CSN antecedent. Although, in most of the cases, the antecedent is given in the post-nominal *wh*, *that*, or infinitive clauses, sometimes it is not present in the immediately following clause but is given only as a predicate, as shown in (3).

(3) The primary **reason that** the archdiocese cannot pay teachers more is **that its students cannot afford higher tuition**.

In such cases, we looked for the pattern *(VP (VB be_verb) X)* in the right sibling of the NP containing the pattern *shell_noun-that* and extracted *X* as the CSN antecedent.

Two contradictory goals need to be achieved while extracting antecedents of CSNs. The first requires only considering CSNs with high-confidence patterns, whereas the second requires considering as many patterns as possible to allow a wide variety of antecedent examples with different linguistic properties (e.g., syntactic shape). Our antecedent extractor tries to find a balance between the two goals.

[6] http://nlp.stanford.edu/software/lex-parser.shtml

	fact	***reason***	***issue***	***decision***	***question***	***possibility***
N-to	–	–	–	***inf* clause**	predicate	*inf* clause
N-be-to	–	*inf* clause	*inf* clause	*inf* clause	*inf*/*wh* clause	*inf* clause
N-that	***that* clause**	**predicate**	**predicate**	–	**predicate**	***that* clause**
N-be-that	*that* clause	*that* clause	*that* clause	*that* clause	*that* clause	*that* clause
N-wh	–	predicate	*wh* clause	*wh* clause	*wh* clause	–
N-be-wh	*wh* clause	*wh* clause	*wh* clause	*wh* clause	*wh* clause	*wh* clause

Table 3: Content extraction patterns for CSNs. Patterns in boldface are the prominent patterns for the respective shell noun. *inf* clause = infinitive clause. Discarded patterns are denoted by –.

To address the first goal, we filter examples following noisy patterns, i.e., the patterns that do not unambiguously encode antecedents of that CSN. For instance, the pattern *N-to* is a highly preferred pattern for *decision*, as shown in (4). The antecedent extraction rule here is relatively simple: if the example follows the pattern *decision-to*, extract the postnominal infinitive clause as the correct antecedent.

(4) President Jacques Chirac's arrogant **decision to defy the world and go ahead with two nuclear bomb tests in Polynesia** deserves contempt.

But the same pattern is noisy for *reason*. In (5), for example, the actual reason is not given anywhere in the sentence. So we discard the examples following the pattern *N-to* for *reason*.

(5) Investors have had **reason to** worry about stocks.

We also discard examples with negative determiners, as in (6), because in such cases, the extraction rules do not precisely give the antecedent of the given CSN.

(6) He was careful to repeat anew that he had made **no decision to** go to war.

For the *N-wh* pattern, we exclude certain *wh* words for certain nouns. For example, we exclude the *wh* word *which* for *question* as the Penn Treebank tagset[7] does not distinguish between *which* as a relative pronoun and as a question. We are interested in the latter but not the former. Other discarded *wh* words include *which* and *when* for *fact*; all *wh* words except *when* and *why* for *reason*, all *wh* words except *how* and *whether* for *issue*; *which*, *whom*, *when*, and *why* for *decision*; *which* and *when* for *question*; and all *wh* words for *possibility*.

[7]The Stanford tagger we employ uses the Penn Treebank tagset (Marcus et al., 1993).

To address the second goal of allowing a wide variety of antecedent examples, we try to include as many patterns as possible for each shell noun, as shown in Table 3. For instance, the patterns *question-to* and *question-be-to* will have infinitive clauses as antecedents (marked as VP or S+VP by the parser), whereas for the examples following patterns *question-wh* and *question-be-wh* the antecedent will be in the *wh* clauses (marked as SBAR). For the pattern *question-that*, the antecedent will be in the predicate (similar to example (3)), which can be a prepositional phrase, a noun phrase or a clause.

5.3 Models for CSN antecedents

The antecedent extractor gives labels for each instance in the CSN corpus. Using this labelled data, we train machine learning ranking models for different shell concepts that capture the characteristic features associated with that shell concept. The following sections describe each step of our ranking models in detail.

5.3.1 Candidate extraction

The first step is to extract the set of eligible antecedent candidates $C = \{C_1, C_2, ..., C_k\}$ for the CSN instance a_i. To train a machine learning model we need positive and negative examples. We already have positive examples for antecedent candidates — the *true* antecedents given by the antecedent extractor. But we also need negative examples of antecedent candidates. By their construction, CSNs have their antecedents in the same sentence. So we extract all syntactic constituents of this sentence, given by the Stanford parser. All the syntactic constituents, except the *true* antecedent, are considered as negative examples. With this candidate extraction method, we end up with many more negative exam-

ples than positive examples, but that is exactly what we expect with ASN antecedent candidates, i.e., the test data on which we will be applying our models.

5.3.2 Features

Although our problem is similar to anaphora resolution, we cannot make use of the usual anaphora or coreference resolution features such as agreement or string matching (Soon et al., 2001) because of the nature of ASN and CSN antecedents. We came up with a set of features based on the properties that were common in both ASN and CSN antecedents, according to our judgement.

Syntactic type of the candidate (S) We observed that each shell noun prefers specific CSN patterns and each pattern involves a particular syntactic type. For instance, *decision* prefers the pattern *N-to* and consequently realizes as its antecedents more verb phrases than, for example, noun phrases. We employ two versions of syntactic type: fine-grained syntactic type given by the Stanford parser (e.g., NP-TMP, RRC) and coarse-grained syntactic type (e.g., NP, VP, S, PP) in which we consider ten basic syntactic categories and map all fine-grained syntactic types to these categories.

Context features (C) Context features allow our models to learn about the contextual clues that signal the antecedent. This class contains two features: (a) coarse-grained syntactic type of left and right siblings of the candidate, and (b) part-of-speech tag of the preceding and following words of the candidate.

Embedding level features (E) These features (Müller, 2008) encode the embedding level of the candidate within its sentence. We consider two embedding level features: top embedding level and immediate embedding level. Top embedding level is the level of embedding of the given candidate with respect to its top clause (the root node), and immediate embedding level is the level of embedding with respect to its immediate clause (the closest ancestor of type S or SBAR). The intuition behind this feature is that if the candidate is deep in the parse tree, it is possibly not salient enough to be an antecedent. As we consider all syntactic constituents as potential candidates, there are many that clearly cannot be antecedents. This feature will allow us to get rid of this noise.

Subordinating conjunctions (SC) As we can see in Table 2, subordinating conjunctions are common with CSN and ASN antecedents. Vendler (1968) points out that the shell noun *fact* prefers a *that*-clause, and *question* and *issue* prefer a *wh*-question clause. We observed that the pattern *because X* is common with *reason*. The subordinating conjunction feature encodes these preferences for different shell nouns. The feature checks whether the candidate follows the pattern *SBAR* $\rightarrow$ *(IN sconj) (S ...)*, where *sconj* is a subordinating conjunction.

Verb features (V) A prominent property of CSN and ASN antecedents is that they tend to contain verbs. All examples from Table 2, for example, contain verbs. Moreover, certain shell nouns have tense and aspect preferences. For instance, for shell noun *fact*, lexical verbs in past and present tenses predominate (Schmid, 2000), whereas modal forms are extremely common for *possibility*. We use three verb features that capture this idea: (a) presence of verbs in general, (b) whether the main verb is finite or non-finite, and (c) presence of modals.

Length features (L) The intuition behind these features is that CSN and ASN antecedents tend to be long, especially for nouns such as *fact*. We consider two length features: (a) length of the candidate in words, and (b) relative length of the candidate with respect to the sentence containing the antecedent.

Lexical features (LX) Our extractor gives us a large number of antecedent examples for each shell noun. A natural question is whether certain words tend to occur more frequently in the antecedent than non-antecedent parts of the sentence. To deal with this question, we extracted all antecedent unigrams (i.e., unigrams occurring in antecedent part of the sentence) and non-antecedent unigrams (i.e., unigrams occurring in non-antecedent parts of the sentence) for each shell noun. Then for all antecedent unigrams for a particular shell noun, we computed *term goodness* in terms of information gain (Yang and Pedersen, 1997) and considered the first 50 highly ranked unigrams as the lexical features for that noun. Note that, in contrast with the other features, these lexical features are tailored for each shell noun and are extracted *a priori*.

5.3.3 Candidate ranking models

Now that we have the set of candidate antecedents and a set of features, we are ready to train CSN antecedent models. We follow the *candidate-ranking* models proposed by Denis and Baldridge (2008) because they allow us to evaluate how good an antecedent candidate is relative to *all* other candidates.

For every shell noun, we gather automatically extracted antecedent data given by the extractor for all instances of that shell noun. Then for each instance in this data, we extract the set C as explained in Section 5.3.1. For each candidate $C_i \in C$, we extract a feature vector to create a corresponding set of feature vectors, $C_f = \{C_{f1}, C_{f2}, ..., C_{fk}\}$. For every CSN a_i and a set of feature vectors corresponding to its eligible candidates $C_f = \{C_{f1}, C_{f2}, ..., C_{fk}\}$, we create training examples $(a_i, C_{fi}, rank), \forall C_{fi} \in C_f$. The rank is 1 if C_i is same as the *true* antecedent, i.e., the automatically extracted antecedent for that CSN, otherwise the rank is 2. We use the `svm_rank_learn` call of SVM^{rank} (Joachims, 2002) for training the candidate-ranking models.

6 Testing phase

In this phase, we use the learned candidate ranking models to identify the antecedents of ASNs.

6.1 The ASN corpus

We started with about 450 instances for each of the six selected shell nouns (2,700 total instances), containing the pattern {*this* shell_noun}. The instances were extracted from the NYT. Each instance contains three paragraphs from the corresponding NYT article: the paragraph containing the ASN and two preceding paragraphs as context. After automatically removing duplicates and ASNs with a non-abstract sense (e.g., *this issue* with a publication-related sense), we were left with 2,323 instances.

6.2 Antecedent identification

Candidate extraction The search space of ASN antecedents is quite large for two reasons: ASNs tend to have long-distance as well as short-distance antecedents, and there is no clear restriction on the syntactic type of the antecedents. In the ASN corpus, each sentence on average had 49.5 distinct syntactic constituents given by the Stanford parser. If we consider n preceding sentences, the sentence containing the anaphor, and one following sentence[8] as sources for antecedents, then the average number of antecedent candidates will be $49.5 \times (n+2)$. This is large compared to the search space of ordinary nominal anaphora. In our previous work (Kolhatkar et al., 2013), we have developed methods that identify the sentence containing the antecedent of the ASN before identifying the precise antecedent. In brief, given a set of a fixed number of sentences around the sentence containing an ASN, these methods reliably identify the sentence containing the antecedent. In this paper, we treat these methods as a black box.

Given the sentence containing the antecedent, we extract all syntactic constituents given by the Stanford parser from that sentence as potential antecedent candidates as for the training phase. In the training phase, the antecedent is always contained in the set of syntactic constituents given by the Stanford parser because the extractor obtains the appropriate antecedent using the syntactic information. But in the testing phase, we cannot guarantee that the true antecedent occurs in the extracted syntactic constituents due to the parser's errors. So for robust candidate extraction, we extract all distinct constituents from the 30-best parses instead of only considering the best parse, which increases the average number of candidates from 49.5 to 55.2.

Feature extraction and candidate ranking Given the antecedent candidates, feature extraction and candidate ranking are essentially the same as for the training phase, except of course we do not know the *true* antecedent. Once we have the feature vectors for each antecedent candidate, the appropriate trained model, i.e., the model trained for the corresponding shell noun, is invoked and the candidates are ranked using the `svm_rank_classify` call of SVM^{rank}.

7 Evaluation

We evaluate the ranked candidates of ASN instances using crowdsourcing.

[8] The ASN corpus contains a few cataphoric examples that do not follow the standard patterns of the CSNs shown in Table 2, but actually refer to an antecedent in the following sentence (e.g., *Mr. Dukakis put **this question** to him: **X***).

Interface We chose to use CrowdFlower[9] as our crowdsourcing interface because of its integrated quality-control mechanism. For instance, it throws gold questions randomly at the workers and the workers who do not answer them correctly are not allowed to continue.

We presented to the crowd evaluators the ASN instances from the ASN corpus. Recall that each ASN instance is made up of the paragraph containing the ASN and two preceding paragraphs as context. We displayed the first 10 highly-ranked candidates (ordered randomly) given by our testing phase and asked the evaluators to choose the best answer that represents the ASN antecedent. We encouraged the evaluators to select *None* when they did not agree with any of the displayed answers. We also asked them how satisfied they were with the displayed answers. We provided them with three options: *unsatisfied*, *satisfied*, and *partially satisfied*.

Our job contained 2,323 evaluation units. We asked for 8 judgements per instance and paid 6 cents per evaluation unit. As we were interested in the verdict of native speakers of English, we limited the allowed demographic region to English-speaking countries.

Results Among the 2,323 ASN instances, 96% of them were labelled as *satisfied*, 3% as *partially satisfied* and 1% as *unsatisfied*. Only 2% of the instances were labelled as *None*. As expected, evaluators were *unsatisfied* or *partially satisfied* with the options of these instances. These results suggest that our resolution models trained on automatically extracted antecedents of CSNs bring the relevant candidates of ASN antecedents to the top, i.e., within first 10 highly-ranked candidates. This itself is a positive result given the large search space of ASN antecedent candidates (more than 55 candidates on average).

Among the evaluation units, more than half of the evaluators agreed on an answer for 1,810 units. We used these instances for further analysis.

To examine which CSN antecedent features are relevant in identifying ASN antecedents, we carried out ablation experiments with all feature class combinations. We compared the rankings given by our ranker to the crowd's answer using *precision at n* (P@n).[10] More specifically, we count the number of instances where the crowd's answers occur within our ranker's first n choices. P@n then is this count divided by the total number of instances. Note that P@1 is equivalent to the standard precision.

We compared our results against two baselines: *preceding sentence* and *chance*. The preceding sentence baseline chooses the previous sentence as the correct antecedent. The chance baseline chooses a candidate from a uniform random distribution over the set of 10 top-ranked candidates.

The results are shown in Table 4. Although different feature combinations gave the best results for different shell nouns, the features that occur frequently in many best-performing combinations were embedding level (E), lexical (LX), and subordinating conjunction (SC) features. The SC features were particularly effective for *issue* and *question*, where we expected patterns such as *whether X*.

Surprisingly, the syntactic type features (S) did not show up very often in the best-performing feature combinations, suggesting that the ASN antecedents had a greater variety of syntactic types than what was available in our CSN training data.

The context features (C) did not appear in any of the best-performing feature combinations. In fact, they resulted in a sharp decline in the precision. For instance, for *question*, adding the context features to the best-performing combination {E,SC,V,L,LX} resulted in a drop of 16 percentage points. This result was not surprising because although the antecedents of ASNs and CSNs share similar properties such as common words, we know that their context is generally different.

We did not observe specific features associated with Schmid's semantic categories. An exception was the E features which were particularly effective for the factual nouns *fact* and *reason*: the results with them alone gave high precision (0.68 for *fact* and 0.72 for *reason*). That said, the E features were present in most of the best-performing combinations even for the shell nouns in other semantic categories.

[9] `http://crowdflower.com/`

[10] CrowdFlower gives us a unique answer for each instance, which we take to be the crowd's answer. During annotation, every annotator is presented with a few gold questions randomly and each annotator is assigned a trust score based on her performance on these gold questions. The unique answer for an instance is the answer with the highest sum of trusts.

fact (43,000 train and 472 test instances)

Features	P@1	P@2	P@3	P@4
{E,L,LX}	**.70***	.85	.91	.94
{E,V,L,LX}	.68*	**.86**	**.92**	**.95**
{E,SC,L,LX}	.66*	.83	**.92**	**.95**
PSbaseline	.40	–	–	–

reason (4,520 train and 443 test instances)

Features	P@1	P@2	P@3	P@4
{E,V,L}	**.72***	**.86**	**.90**	.93
{E,V}	**.72***	.85	**.90**	.92
{E,SC,LX}	.69*	.84	**.90**	**.94**
PSbaseline	.44	–	–	–

issue (3,000 train and 303 test instances)

Features	P@1	P@2	P@3	P@4
{SC,L}	**.47***	.59	.71	.78
{SC,L,LX}	.46*	.60	.70	**.81**
{S,E,SC,L,LX}	.40*	**.61**	**.72**	**.81**
PSbaseline	.30	–	–	–

decision (42,332 train and 390 test instances)

Features	P@1	P@2	P@3	P@4
{E,LX}	**.35***	**.53**	**.67**	**.76**
{E,SC,LX}	.30*	.48	.65	.75
{E,SC,V,L,LX}	.27	.44	.57	.69
PSbaseline	.21	–	–	–

question (9,336 train and 440 test instances)

Features	P@1	P@2	P@3	P@4
{E,SC,V,L,LX}	**.70***	.82	.87	.90
{E,SC,LX}	.68*	**.83**	**.88**	**.91**
{E,SC,V,LX}	.69*	.80	.87	**.91**
PSbaseline	.25	–	–	–

possibility (11,735 train and 278 test instances)

Features	P@1	P@2	P@3	P@4
{SC,L,LX}	**.56***	.75	**.87**	**.92**
{E,SC}	**.56***	**.76**	**.87**	.91
{E,L,LX}	.54*	**.76**	.86	.91
PSbaseline	.34	–	–	–

Table 4: Evaluation of our ranker for antecedents of six ASNs. For each noun we show the three best-performing feature combinations. P@n is the precision at rank n (P@1 = standard precision). Boldface indicates the best in the column. PSbaseline = preceding sentence baseline. The P@1 results significantly higher than PSbaseline are marked with *(two-sample χ^2 test: $p < 0.05$). The chance baseline results were 0.1, 0.2, 0.3, and 0.4 for P@1, P@2, P@3, and P@4, respectively.

The only previous work with which our results could be compared is that of Kolhatkar and Hirst (2012). The work reports precision in the range of 0.41 to 0.61 in resolving *this issue* anaphora in the Medline domain. In our case, for *this issue* instances from the NYT corpus, we achieved precision in the range of 0.40 to 0.47. Furthermore, we applied our models to resolve *this issue* instances from Kolhatkar and Hirst's (2012) work.[11] Even with models trained on automatically labelled CSN antecedents, we achieved similar results to Kolhatkar and Hirst's results: P@1 of 0.45, P@2 of 0.59, P@3 of 0.65, and P@4 of 0.67. These results show the domain robustness of our methods with respect to the shell noun *issue*. Recall that Kolhatkar and Hirst (2012) looked at only very specific cases of *this issue* and used manually annotated data (Section 2), as opposed to the automatically extracted CSN antecedent data we use.

[11] We thank an anonymous reviewer for suggesting this to us.

8 Discussion and conclusion

The goal of this paper was to examine to what extent CSNs help in interpreting ASNs. Based on the evaluators' satisfaction level and very few *None* responses, we conclude that our models trained on CSN antecedents were able to bring the relevant ASN antecedent candidates into the top 10 candidates.

When we applied the models trained on CSN antecedents to interpret ASNs, we achieved precision in the range of 0.35 to 0.72. The precision results as high as 0.72 for *reason* and 0.70 for *fact* and *question* support our hypothesis that the linguistic knowledge provided by CSN antecedents helps in identifying the antecedents of ASNs. We observed different behaviour for different nouns. The mental nouns *issue* and *decision* in general were harder to interpret than other shell nouns. The models trained on CSNs achieved precisions of 0.35 for *decision* and 0.47 for *issue*. So there is still much room for improvement. That said, for the same nouns, the antecedents were in the first four ranks about 76% to 81% of the times,

suggesting that in future research, our models can be used as base models to reduce the large search space of ASN antecedent candidates.

We observed a wide range of performance for different shell nouns. One reason is that the size of the training data was different for different shell nouns. After excluding the noisy examples (Section 5.2), there were about 43,000 training examples for *fact*, but only about 3,000 for *issue*. In addition, a particular shell concept itself can be difficult, e.g., the very idea of what counts as an *issue* is more fuzzy than what counts as a *fact*.

One limitation of our approach is that it only learns the properties that are present in CSN antecedents. However, ASN antecedents have additional properties which are not always captured by CSN antecedents. For instance, for the shell noun *decision*, most of the training examples were infinitive phrases of the form *to X*. But antecedents of the ASN *decision* were mostly court decisions and were expressed with full sentences.

Moreover, although the models trained on CSN antecedents are able to encode characteristic features associated with the general shell concept, they are unable to distinguish between two competing candidates both containing the characteristic features of that shell concept. For instance, our approach will not be able to handle the constructed examples in (7).

(7) The teacher erased the solutions before John had time to copy them out, as he had momentarily been distracted by a band playing outside.

a) **This fact** infuriated him, as the teacher always erased the board quickly and John suspected it was just to punish anyone who was lost in thought, even for a moment.

b) **This fact** infuriated the teacher, who had already told John several times to focus on class work.

Here, both propositions possess properties of the shell concept *fact*. Understanding the context of the anaphor itself is crucial in correctly identifying the fact in each case, which cannot be learnt from CSN antecedents due to their specific context patterns.

A number of extensions are planned for this work. First, we plan to use both kinds of data, CSN and ASN antecedent data, which will give us a basis for developing a better performing ASN resolver. We also plan to incorporate contextual features (e.g., right-frontier rule (Webber, 1991) and context ranking (Eckert and Strube, 2000)). Finally, we will examine whether a model trained for one shell noun can be generalized to other shell nouns from the same semantic category.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This material is based upon work supported by the United States Air Force and the Defense Advanced Research Projects Agency under Contract No. FA8650-09-C-0179, Ontario/Baden-Württemberg Faculty Research Exchange, and the University of Toronto.

References

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers, Dordrecht, Netherlands.

Donna K. Byron. 2004. *Resolving pronominal reference to abstract entities*. Ph.D. thesis, Rochester, New York: University of Rochester.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669, Honolulu, Hawaii, October. Association for Computational Linguistics.

Miriam Eckert and Michael Strube. 2000. Dialogue acts, synchronizing units, and anaphora resolution. *Journal of Semantics*, 17:51–89.

Gill Francis. 1986. *Anaphoric Nouns*. Discourse Analysis Monographs 11, Birmingham: English Language Research, University of Birmingham.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman Pub Group.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France. Association for Computational Linguistics.

Roz Ivanic. 1991. Nouns in search of a context: A study of nouns with both open- and closed-system characteristics. *International Review of Applied Linguistics in Language Teaching*, 29:93–114.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference*

on Knowledge Discovery and Data Mining (KDD), pages 133–142.

Varada Kolhatkar and Graeme Hirst. 2012. Resolving "this-issue" anaphora. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1255–1265, Jeju Island, Korea, July. Association for Computational Linguistics.

Varada Kolhatkar, Heike Zinsmeister, and Graeme Hirst. 2013. Annotating anaphoric shell nouns with their antecedents. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 112–121, Sofia, Bulgaria, August. Association for Computational Linguistics.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 368–375, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Christoph Müller. 2008. *Fully Automatic Resolution of It, This and That in Unrestricted Multi-Party Dialog*. Ph.D. thesis, Universität Tübingen.

Rebecca J. Passonneau. 1989. Getting at discourse referents. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 51–59, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Massimo Poesio, Simone Ponzetto, and Yannick Versley. 2011. Computational models of anaphora resolution: A survey. Unpublished.

Hans-Jörg Schmid. 2000. *English Abstract Nouns As Conceptual Shells: From Corpus to Cognition*. Topics in English Linguistics 34. Mouton de Gruyter, Berlin.

Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Michael Strube and Christoph Müller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Sapporo, Japan, July. Association for Computational Linguistics.

Zeno Vendler. 1968. *Adjectives and Nominalizations.* Mouton de Gruyter, The Hague.

Bonnie Lynn Webber. 1991. Structure and ostension in the interpretation of discourse deixis. In *Language and Cognitive Processes*, pages 107–135.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, Nashville, TN.

Exploring Representations from Unlabeled Data with Co-training for Chinese Word Segmentation

Longkai Zhang Houfeng Wang* Xu Sun Mairgup Mansur
Key Laboratory of Computational Linguistics (Peking University) Ministry of Education, China
zhlongk@qq.com, wanghf@pku.edu.cn, xusun@pku.edu.cn, mairgup@gmail.com,

Abstract

Nowadays supervised sequence labeling models can reach competitive performance on the task of Chinese word segmentation. However, the ability of these models is restricted by the availability of annotated data and the design of features. We propose a scalable semi-supervised feature engineering approach. In contrast to previous works using pre-defined task-specific features with fixed values, we dynamically extract representations of label distributions from both an in-domain corpus and an out-of-domain corpus. We update the representation values with a semi-supervised approach. Experiments on the benchmark datasets show that our approach achieve good results and reach an f-score of 0.961. The feature engineering approach proposed here is a general iterative semi-supervised method and not limited to the word segmentation task.

1 Introduction

Chinese is a language without natural word delimiters. Therefore, Chinese Word Segmentation (CWS) is an essential task required by further language processing. Previous research shows that sequence labeling models trained on labeled data can reach competitive accuracy on the CWS task, and supervised models are more accurate than unsupervised models (Xue, 2003; Low et al., 2005). However, the resource of manually labeled training corpora is limited. Therefore, semi-supervised learning has become one of the most natural forms of training for CWS. Traditional semi-supervised methods focus on adding new unlabeled instances to the training set by a given criterion. The possible mislabeled instances, which are introduced from the automatically labeled raw data, can hurt the performance and not easy to exclude by setting a sound selecting criterion.

In this paper, we propose a simple and scalable semi-supervised strategy that works by providing semi-supervision at the level of representation. Previous works mainly assume that context features are helpful to decide the potential label of a character. However, when some of the context features do not appear in the training corpus, this assumption may fail. An example is shown in table 1. Although the context of "水" and "篮" is totally different, they share a homogeneous structure as "verb-noun". Therefore. A much better way is to map the context information to a kind of representation. More precisely, the mapping should let the similar contexts map to similar representations, while let the distinct contexts map to distinct representations.

	吃水果	打篮球
Label	B	B
Character	吃 水 果	打 篮 球
Context Features	C-1= 吃 C0= 水 C1= 果	C-1= 打 C0= 篮 C1= 球

Table 1: Example of the context of "水" in "吃水果 (Eat fruits)" and the context of "篮" in "打篮球 (Play basketball)"

We use the label distribution information that

*Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 311–321,
Seattle, Washington, USA, 18-21 October 2013.

is extracted from the unlabeled corpus as this representation to enhance the supervised model. We add "pseudo-labels" by tagging the unlabeled data with the trained model on the training corpus. These "pseudo-labels" are not accurate enough. Therefore, we use the label distribution, which is much more accurate.

To accurately calculate the precise label distribution, we use a framework similar to the co-training algorithm to adjust the feature values iteratively. Generally speaking, unlabeled data can be classified as in-domain data and out-of-domain data. In previous works these two kinds of unlabeled data are used separately for different purposes. In-domain data is mainly used to solve the problem of data sparseness (Sun and Xu, 2011). On the other hand, out-of domain data is used for domain adaptation (Chang and Han, 2010). In our work, we use in-domain and out-of-domain data together to adjust the labels of the unlabeled corpus.

We evaluate the performance of CWS on the benchmark dataset of Peking University in the second International Chinese Word Segmentation Bakeoff. Experiment results show that our approach yields improvements compared with the state-of-art systems. Even when the labeled data is insufficient, our methods can still work better than traditional methods. Compared to the baseline CWS model, which has already achieved an f-score above 0.95, we further reduce the error rate by 15%.

Our method is not limited to word segmentation. It is also applicable to other problems which can be solved by sequence labeling models. We also applied our method to the Chinese Named Entity Recognition task, and also achieved better results compared to traditional methods.

The main contributions of our work are as follows:

- We proposed a general method to utilize the label distribution given text contexts as representations in a semi-supervised framework. We let the co-training process adjust the representation values from label distribution instead of using manually predefined feature templates.
- Compared with previous work, our method achieved a new state-of-art accuracy on the CWS task as well as on the NER task.

The remaining part of this paper is organized as follows. Section 2 describes the details of the problem and our algorithm. Section 3 describes the experiment and presents the results. Section 4 reviews the related work. Section 5 concludes this paper.

2 System Architecture

2.1 Sequence Labeling

Nowadays the character-based sequence labeling approach is widely used for the Chinese word segmentation problem. It was first proposed in Xue (2003), which assigns each character a label to indicate its position in the word. The most prevalent tag set is the BMES tag set, which uses 4 tags to carry word boundary information. This tag set uses B, M, E and S to represent the Beginning, the Middle, the End of a word and a Single character forming a word respectively. We use this tag set in our method. An example of the "BMES" representation is shown in table 2.

Character:	我	爱	北	京	天	安	门
Tag:	S	S	B	E	B	M	E

Table 2: An example for the "BMES" representation. The sentence is "我爱北京天安门" (I love Beijing Tian-an-men square), which consists of 4 Chinese words: "我" (I), "爱" (love), "北京" (Beijing), and "天安门" (Tian-an-men square).

2.2 Unlabeled Data

Unlabeled data can be divided into in-domain data and out-of-domain data. In previous works, these two kinds of unlabeled data are used separately for different purposes. In-domain data only solves the problem of data sparseness (Sun and Xu, 2011). Out-of domain data is used only for domain adaptation (Chang and Han, 2010). These two functionalities are not contradictory but complementary. Our study shows

that by correctly designing features and algorithms, both in-domain unlabeled data and out-of-domain unlabeled data can work together to help enhancing the segmentation model. In our algorithm, the dynamic features learned from one corpus can be adjusted incrementally with the dynamic features learned from the other corpus.

As for the out-of-domain data, it will be even better if the corpus is not limited to a specific domain. We choose a Chinese encyclopedia corpus which meets exactly this requirement. We use the corpus to learn a large set of informative features. In our experiment, two different views of features on unlabeled data are considered:

Static Statistical Features (SSFs): These features capture statistical information of characters and character n-grams from the unlabeled corpus. The values of these features are fixed during the training process once the unlabeled corpus is given.

Dynamic Statistical Features (DSFs): These features capture label distribution information from the unlabeled corpus given fixed text contexts. As the training process proceeds, the value of these features will change, since the trained tagger at each training iteration may assign different labels to the unlabeled data.

2.3 Framework

Suppose we have labeled data L, two unlabeled corpora U_a and U_b (one is an in-domain corpus and the other is an out-of-domain corpus). Our algorithm is shown in Table 3.

During each iteration, we tag the unlabeled corpus U_a using T_b to get pseudo-labels. Then we extract features from the pseudo-labels. We use the label distribution information as dynamic features. We add these features to the training data to train a new tagger T_a. To adjust the feature values, we extract features from one corpus and then apply the statistics to the other corpus. This is similar to the principle of co-training (Yarowsky, 1995; Blum and Mitchell, 1998; Dasgupta et al., 2002). The difference is that there are not different views of features, but different kinds of unlabeled data. Detailed description of features is given in the next section.

Algorithm
Init:
Using baseline features only:
Train an initial tagger T_0 based on L ()
Label U_a and U_b individually using T_0
BEGIN LOOP:
Generate DSFs from tagged U_a
Augment L with DSFs to get L_a
Generate DSFs from tagged U_b
Augment L with DSFs to get L_b
Using baseline features, SSFs and DSFs:
Train new tagger T_a using L_a
Train new tagger T_b using L_b
Label U_a using T_b
Label U_b using T_a
LOOP until performance does not improve
RETURN the tagger which is trained with in-domain features.

Table 3: Algorithm description

2.4 Features

2.4.1 Baseline Features

Our baseline feature templates include the features described in previous works (Sun and Xu, 2011; Sun et al., 2012). These features are widely used in the CWS task. To be convenient, for a character c_i with context $\ldots c_{i-1}c_ic_{i+1}\ldots$, its baseline features are listed below:

- Character uni-grams: c_k $(i-3<k<i+3)$
- Character bi-grams: c_kc_{k+1} $(i-3<k<i+2)$
- Whether c_k and c_{k+1} are identical $(i-2<k<i+2)$
- Whether c_k and c_{k+2} are identical $(i-4<k<i+2)$

The last two feature templates are designed to detect character reduplication, which is a morphological phenomenon in Chinese language. An example is "十全十美" (Perfect), which is a Chinese idiom with structure "ABAC".

2.4.2 Static statistical features

Statistical features are statistics that distilled from the large unlabeled corpus. They are proved useful in the Chinese word segmentation task. We define Static Statistical Features (SSFs) as features whose value do not change during the training process. The SSFs in our approach includes Mutual information, Punctuation information and Accessor variety. Previous works have already explored the functions of the three static statistics in the Chinese word segmentation task, e.g. Feng et al. (2004); Sun and Xu (2011). We mainly follow their definitions while considering more details and giving some modification.

Mutual information

Mutual information (MI) is a quantity that measures the mutual dependence of two random variables. Previous works showed that larger MI of two strings claims higher probability that the two strings should be combined. Therefore, MI can show the tendency of two strings forming one word. However, previous works mainly focused on the balanced case, i.e., the MI of strings with the same length. In our study we find that, in Chinese, there remains large amount of imbalanced cases, like a string with length 1 followed by a string with length 2, and vice versa. We further considered the MI of these string pairs to capture more information.

Punctuation information

Punctuations can provide implicit labels for the characters before and after them. The character after punctuations must be the first character of a word. The character before punctuations must be the last character of a word. When a string appears frequently after punctuations, it tends to be the beginning of a word. The situation is similar when a string appears frequently preceding punctuations. Besides, the probability of a string appears in the corpus also affects this tendency. Considering all these factors, we propose "punctuation rate" (PR) to capture this information. For a string with length len and probability p in the corpus, we define the left punctuation rate LPR_{len} as the number of times the string appears after punctuations, divided by p. Similarly, the right punctuation rate RPR_{len} is defines as the number of times it appears preceding punctuations divided by its probability p. The length of string we consider is from 1 to 4.

Accessor variety

Accessor variety (AV) is also known as letter successor variety (LSV) (Harris, 1955; Hafer and Weiss, 1974). If a string appears after or preceding many different characters, this may provide some information of the string itself. Previous work of Feng et al. (2004), Sun and Xu (2011) used AV to represent this statistic. Similar to punctuation rate, we also consider both left AV and right AV. For a string s with length l, we define the left accessor variety (LAV) as the types of distinct characters preceding s in the corpus, and the right accessor variety (RAV) as the types of distinct characters after s in the corpus. The length of string we consider is also from 1 to 4.

2.4.3 Dynamic statistical features

The unlabeled corpus lacks precise labels. We can use the trained tagger to give the unlabeled data "pseudo-labels". These labels cannot guarantee an acceptable precision. However, the label distribution will not be largely affected by small mistakes. Using the label distribution information is more accurate than using the pseudo-labels directly.

Based on this assumption, we propose "dynamic statistical features" (DSFs). The DSFs are intended to capture label distribution information given a text context. The word "Dynamic" is in accordance with the fact that these feature values will change during the training process.

We give a formal description of DSFs. Suppose there are K labels in our task. For example, $K = 4$ if we take BMES labeling method. We define the whole character sequence with length n as $X = (x_1, x_2 \cdots x_j \cdots x_n)$. Given a text context C_i, where i is current character position, the DSFs can be represented as a list,

$$DSF(C_i) = (DSF(C_i)_1, \cdots, DSF(C_i)_K)$$

Each element in the list represents the probability of the corresponding label in the distribution.

For convenience, we further define function 'count(condition)' as the total number of times a 'condition' is true in the unlabeled corpus. For example, count (current='a') represents the times the current character equals 'a', which is exactly the number of times character 'a' appears in the unlabeled corpus.

According to different types of text context C_i, we can divide DSFs into 3 types:

1.Basic DSF

For Basic DSF of C_i, we define $D(C_i)$:

$$D(C_i) = (D(C_i)_1, \ldots, D(C_i)_K)$$

We define Basic DSF with current character position i, text context C_i and label l (the lth dimension in the list) as:

$$\begin{aligned} D(C_i)_l &= P(y = l | C_i = x_i) \\ &= \frac{count(C_i = x_i \wedge y = l)}{count(C_i = x_i)} \end{aligned}$$

In this equation, the numerator counts the number of times current character is x_i with label l. The denominator counts the number of times current character is x_i.

We use the term "Basic" because this kind of DSFs only considers the character of position i as its context. The text context refers to the current character itself. This feature captures the label distribution information given the character itself.

2.BigramDSF

Basic DSF is simple and very easy to implement. The weakness is that it is less powerful to describe word-building features. Although characters convey context information, characters themselves in Chinese is sometimes meaningless. Character bi-grams can carry more context information than uni-grams. We modify Basic DSFs to bi-gram level and propose Bigram DSFs.

For Bigram DSF of C_i, we define $B(C_i)$:

$$B(C_i) = (B(C_i)_1, \ldots, B(C_i)_K)$$

We define Bigram DSF with current character position i, text context C_i and label l (the lth dimension in the list) as:

$$\begin{aligned} B(C_i)_l &= P(y = l | C_i = x_{i-j}x_{i-j+1}) \\ &= \frac{count(C_i = x_{i-j}x_{i-j+1} \wedge y = l)}{count(C_i = x_{i-j}x_{i-j+1})} \end{aligned}$$

j can take value 0 and 1.

In this equation, the numerator counts the number of times current context is $x_{i-j}x_{i-j+1}$ with label l. The denominator counts the number of times current context is $x_{i-j}x_{i-j+1}$.

3.WindowDSF

Considering Basic DSF and Bigram DSF only might cause the over-fitting problem, therefore we introduce another kind of DSF. We call it Window DSF, which considers the surrounding context of a character and omits the character itself.

For Window DSF, we define $W(C_i)$:

$$W(C_i) = (W(C_i)_1, \ldots, W(C_i)_K)$$

We define Window DSF with current character position i, text context C_i and label l (the lth dimension in the list) as:

$$\begin{aligned} W(C_i)_l &= P(y = l | C_i = x_{i-1}x_{i+1}) \\ &= \frac{count(C_i = x_{i-1}x_{i+1} \wedge y = l)}{count(C_i = x_{i-1}x_{i+1})} \end{aligned}$$

In this equation, the numerator counts the number of times current context is $x_{i-1}x_{i+1}$ with label l. The denominator counts the number of times current context is $x_{i-1}x_{i+1}$.

2.4.4 Discrete features VS. Continuous features

The statistical features may be expressed as real values. A more natural way is to use discrete values to incorporate them into the sequence labeling models . Previous works like Sun and Xu (2011) solve this problem by setting thresholds and converting the real value into boolean values. We use a different method to solve this, which does not need to consider tuning thresholds. In our method, we process static and dynamic statistical features using different strategies.

For static statistical value:

For mutual information, we round the real value to their nearest integer. For punctuation rate and accessor variety, as the values tend to be large, we first get the log value of the feature and then use the nearest integer as the corresponding discrete value.

For dynamic statistical value:

Dynamic statistical features are distributions of a label. The values of DSFs are all percentage values. We can solve this by multiply the probability by an integer N and then take the integer part as the final feature value. We set the value of N by cross-validation..

2.5 Conditional Random Fields

Our algorithm is not necessarily limited to a specific baseline tagger. For simplicity and reliability, we use a simple Conditional Random Field (CRF) tagger, although other sequence labeling models like Semi-Markov CRF Gao et al. (2007) and Latent-variable CRF Sun et al. (2009) may provide better results than a single CRF. Detailed definition of CRF can be found in Lafferty et al. (2001); McCallum (2002); Pinto et al. (2003).

3 Experiment

3.1 Data and metrics

We used the benchmark datasets provided by the second International Chinese Word Segmentation Bakeoff[1] to test our approach. We chose the Peking University (PKU) data in our experiment. Although the benchmark provides another three data sets, two of them are data of traditional Chinese, which is quite different from simplified Chinese. Another is the data from Microsoft Research (MSR). We experimented on this data and got 97.45% in f-score compared to the state-of-art 97.4% reported in Sun et al. (2012). However, this corpus is much larger than the PKU corpus. Using the labeled data alone can get a relatively good tagger and the unlabeled data contributes little to the performance. For simplicity and efficiency, our further experiments are all conducted on the PKU data. Details of the PKU data are listed in table 4.

We also used two un-segmented corpora as unlabeled data. The first one is Chinese Gigaword[2] corpus. It is a comprehensive archive of newswire data. The second one is articles from Baike[3] of baidu.com. It is a Chinese encyclopedia similar to Wikipedia but contains more Chinese items and their descriptions. In the experiment we used about 5 million characters from each corpus for efficiency. Details of unlabeled data can be found in table 5.

In our experiment, we did not use any extra resources such as common surnames, part-of-speech or other dictionaries.

F-score is used as the accuracy measure. We define precision P as the percentage of words in the output that are segmented correctly. We define recall R as the percentage of the words in reference that are correctly segmented. Then F-score is as follows:

$$F = \frac{2 \times P \times R}{P + R}$$

The recall of out-of-vocabulary is also taken into consideration, which measures the ability of the model to correctly segment out of vocabulary words.

3.2 Main Results

Table 6 summarizes the segmentation results on test data with different feature combinations. We performed incremental evaluation. In this table, we first present the results of the tagger only using baseline features. Then we show the results of adding SSF and DSF individually. In the end we compare the results of combining SSF and DSF with baseline features.

Because the baseline features is strong to reach a relative good result, it is not easy to largely enhance the performance. Nevertheless, there are significant increases in f-score and OOV-Recall when adding these features. From table 6 we can see that by adding SSF and DSF individually, the F-score is improved by +1.1%

[1] http://www.sighan.org/bakeoff2005/

[2] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2003T09

[3] http://baike.baidu.com/

Identical words	Total word	Identical Character	Total character
5.5×10^4	1.1×10^6	5×10^3	1.8×10^6

Table 4: Details of the PKU data

Corpus	**Character used**
Gigaword	5000193
Baike	5000147

Table 5: Details of the unlabeled data.

	P	R	F	OOV
Baseline	0.950	0.943	0.946	0.676
+SSF	0.961	0.953	0.957	0.728
+DSF	0.958	0.953	0.955	0.678
+SSF+DSF	0.965	0.958	0.961	0.731

Table 6: Segmentation results on test data with different feature combinations. The symbol "+" means this feature configuration contains features set containing the baseline features and all features after '+' . The size of unlabeled data is fixed as 5 million characters.

and +0.9%. The OOV-Recall is also improved, especially after adding SSFs. When considering SSF and DSF together, the f-score is improved by +1.5% while the OOV-Recall is improved by +5.5%.

To compare the contribution of unlabeled data, we conduct experiments of using different sizes of unlabeled data. Note that the SSFs are still calculated using all the unlabeled data. However, each iteration in the algorithm uses unlabeled data with different sizes.

Table 7 shows the results when changing the size of unlabeled data. We experimented on three different sizes: 0.5 million, 1 million and 5 million characters.

	P	R	F	OOV
DSF(0.5M)	0.962	0.954	0.958	0.727
DSF(1M)	0.963	0.955	0.959	0.728
DSF(5M)	0.965	0.958	0.961	0.731

Table 7: Comparison of results when changing the size of unlabeled data. (0.5 million, 1 million and 5 million characters).

We further experimented on unlabeled corpus with larger size (up to 100 million characters). However the performance did not change significantly. Besides, because the number of features in our method is very large, using too large unlabeled corpus is intractable in real applications due to the limitation of memory.

Our method can keep working well even when the labeled data are insufficient. Table 8 shows the comparison of f-scores when changing the size of labeled data. We compared the results of using all labeled data with 3 different situations: using 1/10, 1/2 and 1/4 of all the labeled data. In fact, the best system on the Second International Chinese Word Segmentation bakeoff reached 0.95 in f-score by using all labeled data. From table 8 we can see that our algorithm only needs 1/4 of all labeled data to achieve the same f-score.

	Baseline	+SSF+DSF	Improve
1/10	0.934	0.943	+0.96%
1/4	0.946	0.951	+0.53%
1/2	0.952	0.956	+0.42%
All	0.957	0.961	+0.42%

Table 8: Comparison of f-scores when changing the size of labeled data. (1/10, 1/4, 1/2 and all labeled data. The size of unlabeled data is fixed as 5 million characters.)

We also explored how the performance changes as iteration increases. Figure 1 shows the change of F-score during the first 10 iterations. From figure 1 we find that f-score has a fast improvement in the first few iterations, and then stables at a fixed point. Besides, as the size of labeled data increases, it converges faster.

Using an in-domain corpus and an out-of-domain corpus is better than use one corpus alone. We compared our approach with the method which uses only one unlabeled corpus. To use only one corpus, we modify our algorithm to extract DSFs from the Chinese Giga word corpus and apply the learned features to itself.

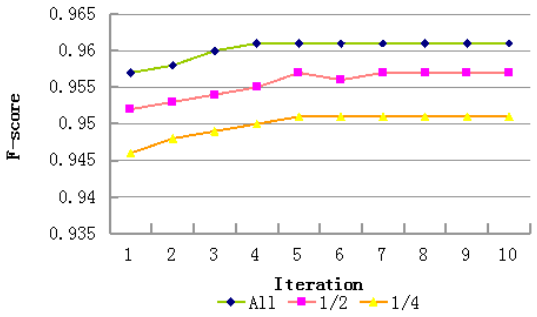


Figure 1: Learning curve of using different size of labeled data

Table 9 shows the result. We can see that our method outperforms by +0.2% in f-score and +0.7% in OOV-Recall.

Finally, we compared our method with the state-of-art systems reported in the previous papers. Table 10 listed the results. Best05 represents the best system reported on the Second International Chinese Word Segmentation Bakeoff. CRF + Rule system represents a combination of CRF model and rule based model presented in Zhang et al. (2006). Other three systems all represent the methods using their corresponding model in the corresponding papers. Note that these state-of-art systems are either using complicated models with semi-Markov relaxations or latent variables, or modifying models to fit special conditions. Our system uses a single CRF model. As we can see in table 10, our method achieved higher F-scores than the previous best systems.

3.3 Results on NER task

Our method is not limited to the CWS problem. It is applicable to all sequence labeling problems. We applied our method on the Chinese NER task. We used the MSR corpus of the sixth SIGHAN Workshop on Chinese Language Processing. It is the only NER corpus using simplified Chinese in that workshop. We compared our method with the pure sequence labeling approach in He and Wang (2008). We reimplemented their method to eliminate the difference of various CRFs implementations. Experiment results are shown in table 11. We can see that our methods works better, especially when handling the out-of-vocabulary named entities;

4 Related work

Recent studies show that character sequence labeling is an effective method of Chinese word segmentation for machine learning (Xue, 2003; Low et al., 2005; Zhao et al., 2006a,b). These supervised methods show good results. Unsupervised word segmentation (Maosong et al., 1998; Peng and Schuurmans, 2001; Feng et al., 2004; Goldwater et al., 2006; Jin and Tanaka-Ishii, 2006) takes advantage of the huge amount of raw text to solve Chinese word segmentation problems. These methods need no annotated corpus, and most of them use statistics to help model the problem. However, they usually are less accurate than supervised ones.

Currently "feature-engineering" methods have been successfully applied into NLP applications. Miller et al. (2004) applied this method to named entity recognition. Koo et al. (2008) applied this method to dependency parsing. Turian et al. (2010) applied this method to both named entity recognition and text chunking. These papers shared the same concept of word clustering. However, we cannot simply equal Chinese character to English word because characters in Chinese carry much less information than words in English and the clustering results is less meaningful.

Features extracted from large unlabeled corpus in previous works mainly focus on statistical information of characters. Feng et al. (2004) used the accessor variety criterion to extract word types. Li and Sun (2009) used punctuation information in Chinese word segmentation by introducing extra labels 'L' and 'R'. Chang and Han (2010), Sun and Xu (2011) used rich statistical information as discrete features in a sequence labeling framework. All these approaches can be viewed as using static statistics features in a supervised approach. Our method is different from theirs. For the static statistics features in our approach, we not only consider richer string pairs with the different lengths, but also consider term frequency when processing

	P	R	F	OOV
Using one corpus	0.963	0.955	0.959	0.724
Our method	0.965	0.958	0.961	0.731

Table 9: Comparison of our approach with using only the Gigaword corpus

Method	P	R	F-score
Best05 (Chen et al. (2005))	0.953	0.946	0.950
CRF + rule-system (Zhang et al. (2006))	0.947	0.955	0.951
Semi-perceptron (Zhang and Clark (2007))	N/A	N/A	0.945
Latent-variable CRF (Sun et al. (2009))	0.956	0.948	0.952
ADF-CRF (Sun et al. (2012))	0.958	0.949	0.954
Our method	0.965	0.958	0.961

Table 10: Comparison of our approach with the state-of-art systems

	P	R	F	OOV
Traditional	0.925	0.872	0.898	0.712
Our method	0.916	0.887	0.902	0.737

Table 11: Comparison of our approach with traditional NER systems

punctuation features.

There are previous works using features extracted from label distribution of unlabeled corpus in NLP tasks. Schapire et al. (2002) use a set of features annotated with majority labels to boost a logistic regression model. We are different from their approach because there is no pseudo-example labeling process in our approach. Qi et al. (2009) investigated on large set of distribution features and used these features in a self-training way. They applied the method on three tasks: named entity recognition, POS tagging and gene name recognition and got relatively good results. Our approach is different from theirs. Although we all consider label distribution, the way we use features are different. Besides, our approach uses two unlabeled corpora which can mutually enhancing to get better result.

5 Conclusion and Perspectives

In this paper, we presented a semi-supervised method for Chinese word segmentation. Two kinds of new features are used for the iterative modeling: static statistical features and dynamic statistical features. The dynamic statistical features use label distribution information for text contexts, and can be adjusted automatically during the co-training process. Experimental results show that the new features can improve the performance on the Chinese word segmentation task. We further conducted experiments to show that the performance is largely improved, especially when the labeled data is insufficient.

The proposed iterative semi-supervised method is not limited to the Chinese word segmentation task. It can be easily extended to any sequence labeling task. For example, it works well on the NER task as well. As our future work, we plan to apply our method to other natural language processing tasks, such as text chunking.

Acknowledgments

This research was partly supported by Major National Social Science Fund of China(No. 12&ZD227),National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101) and National Natural Science Foundation of China (No.91024009). We also thank Xu Sun and Qiuye Zhao for proofreading the paper.

References

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Chang, B. and Han, D. (2010). Enhancing domain portability of chinese segmentation model using chi-square statistics and bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 789–798. Association for Computational Linguistics.

Chen, A., Zhou, Y., Zhang, A., and Sun, G. (2005). Unigram language model for chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 138–141. Association for Computational Linguistics Jeju Island, Korea.

Dasgupta, S., Littman, M. L., and McAllester, D. (2002). Pac generalization bounds for co-training. *Advances in neural information processing systems*, 1:375–382.

Feng, H., Chen, K., Deng, X., and Zheng, W. (2004). Accessor variety criteria for chinese word extraction. *Computational Linguistics*, 30(1):75–93.

Gao, J., Andrew, G., Johnson, M., and Toutanova, K. (2007). A comparative study of parameter estimation methods for statistical natural language processing. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 824.

Goldwater, S., Griffiths, T., and Johnson, M. (2006). Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics.

Hafer, M. A. and Weiss, S. F. (1974). Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Harris, Z. S. (1955). From phoneme to morpheme. *Language*, 31(2):190–222.

He, J. and Wang, H. (2008). Chinese named entity recognition and word segmentation based on character. In *Sixth SIGHAN Workshop on Chinese Language Processing*, page 128.

Jin, Z. and Tanaka-Ishii, K. (2006). Unsupervised segmentation of chinese text by use of branching entropy. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 428–435. Association for Computational Linguistics.

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Li, Z. and Sun, M. (2009). Punctuation as implicit annotations for chinese word segmentation. *Computational Linguistics*, 35(4):505–512.

Low, J., Ng, H., and Guo, W. (2005). A maximum entropy approach to chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 164. Jeju Island, Korea.

Maosong, S., Dayang, S., and Tsou, B. (1998). Chinese word segmentation without using lexicon and hand-crafted training data. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1265–1271. Association for Computational Linguistics.

McCallum, A. (2002). Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 403–410. Morgan Kaufmann Publishers Inc.

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL*, volume 4.

Peng, F. and Schuurmans, D. (2001). Self-supervised chinese word segmentation. *Ad-*

vances in Intelligent Data Analysis, pages 238–247.

Pinto, D., McCallum, A., Wei, X., and Croft, W. (2003). Table extraction using conditional random fields. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 235–242. ACM.

Qi, Y., Kuksa, P., Collobert, R., Sadamasa, K., Kavukcuoglu, K., and Weston, J. (2009). Semi-supervised sequence labeling with self-learned features. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 428–437. IEEE.

Schapire, R., Rochery, M., Rahim, M., and Gupta, N. (2002). Incorporating prior knowledge into boosting. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 538–545.

Sun, W. and Xu, J. (2011). Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics.

Sun, X., Wang, H., and Li, W. (2012). Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 253–262, Jeju Island, Korea. Association for Computational Linguistics.

Sun, X., Zhang, Y., Matsuzaki, T., Tsuruoka, Y., and Tsujii, J. (2009). A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64. Association for Computational Linguistics.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Xue, N. (2003). Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

Zhang, R., Kikui, G., and Sumita, E. (2006). Subword-based tagging by conditional random fields for chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196. Association for Computational Linguistics.

Zhang, Y. and Clark, S. (2007). Chinese segmentation with a word-based perceptron algorithm. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 840.

Zhao, H., Huang, C., and Li, M. (2006a). An improved chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 117. Sydney: July.

Zhao, H., Huang, C., Li, M., and Lu, B. (2006b). Effective tag set selection in chinese word segmentation via conditional random field modeling. In *Proceedings of PACLIC*, volume 20, pages 87–94.

Efficient Higher-Order CRFs for Morphological Tagging

Thomas Müller[†‡], Helmut Schmid[†‡], and Hinrich Schütze[†]

[†]Center for Information and Language Processing, University of Munich, Germany
[‡]Institute for Natural Language Processing , University of Stuttgart, Germany
muellets@cis.lmu.de

Abstract

Training higher-order conditional random fields is prohibitive for huge tag sets. We present an approximated conditional random field using coarse-to-fine decoding and early updating. We show that our implementation yields fast and accurate morphological taggers across six languages with different morphological properties and that across languages higher-order models give significant improvements over 1^{st}-order models.

1 Introduction

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are arguably one of the best performing sequence prediction models for many Natural Language Processing (NLP) tasks. During CRF training forward-backward computations, a form of dynamic programming, dominate the asymptotic runtime. The training and also decoding times thus depend polynomially on the size of the tagset and exponentially on the order of the CRF. This probably explains why CRFs, despite their outstanding accuracy, normally only are applied to tasks with small tagsets such as Named Entity Recognition and Chunking; if they are applied to tasks with bigger tagsets – e.g., to part-of-speech (POS) tagging for English – then they generally are used as 1^{st}-order models.

In this paper, we demonstrate that fast and accurate CRF training and tagging is possible for large tagsets of even thousands of tags by approximating the CRF objective function using coarse-to-fine decoding (Charniak and Johnson, 2005; Rush and Petrov, 2012). Our pruned CRF (PCRF) model has much smaller runtime than higher-order CRF models and may thus lead to an even broader application of CRFs across NLP tagging tasks.

We use POS tagging and combined POS and morphological (POS+MORPH) tagging to demonstrate the properties and benefits of our approach. POS+MORPH disambiguation is an important preprocessing step for syntactic parsing. It is usually tackled by applying sequence prediction. POS+MORPH tagging is also a good example of a task where CRFs are rarely applied as the tagsets are often so big that even 1^{st}-order dynamic programming is too expensive. A workaround is to restrict the possible tag candidates per position by using either morphological analyzers (MAs), dictionaries or heuristics (Hajič, 2000). In this paper, however we show that when using pruning (i.e., PCRFs), CRFs can be trained in reasonable time, which makes hard constraints unnecessary.

In this paper, we run successful experiments on six languages with different morphological properties; we interpret this as evidence that our approach is a general solution to the problem of POS+MORPH tagging. The tagsets in our experiments range from small sizes of 12 to large sizes of up to 1811. We will see that even for the smallest tagset, PCRFs need only 40% of the training time of standard CRFs. For the bigger tagset sizes we can reduce training times from several days to several hours. We will also show that training higher-order PCRF models takes only several minutes longer than training 1^{st}-order models and – depending on the language – may lead to substantial accuracy im-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 322–332,
Seattle, Washington, USA, 18-21 October 2013.

Language	Sentences	Tokens	POS Tags	MORPH Tags	POS+MORPH Tags	OOV Rate
ar (Arabic)	15,760	614,050	38	516	516	4.58%
cs (Czech)	38,727	652,544	12	1,811	1,811	8.58%
en (English)	38,219	912,344	45		45	3.34%
es (Spanish)	14,329	427,442	12	264	303	6.47%
de (German)	40,472	719,530	54	255	681	7.64%
hu (Hungarian)	61,034	1,116,722	57	1,028	1,071	10.71%

Table 1: Training set statistics. Out-Of-Vocabulary (OOV) rate is regarding the development sets.

provements. For example in German POS+MORPH tagging, a 1^{st}-order model (trained in 32 minutes) achieves an accuracy of 88.96 while a 3^{rd}-order model (trained in 35 minutes) achieves an accuracy of 90.60.

The remainder of the paper is structured as follows: Section 2 describes our CRF implementation[1] and the feature set used. Section 3 summarizes related work on tagging with CRFs, efficient CRF tagging and coarse-to-fine decoding. Section 4 describes experiments on POS tagging and POS+MORPH tagging and Section 5 summarizes the main contributions of the paper.

2 Methodology

2.1 Standard CRF Training

In a standard CRF we model our sentences using a globally normalized log-linear model. The probability of a tag sequence $\vec{y}$ given a sentence $\vec{x}$ is then given as:

$$p(\vec{y}|\vec{x}) = \frac{\exp \sum_{t,i} \lambda_i \cdot \phi_i(\vec{y}, \vec{x}, t)}{Z(\vec{\lambda}, \vec{x})}$$
$$Z(\vec{\lambda}, \vec{x}) = \sum_{\vec{y}} \exp \sum_{t,i} \lambda_i \cdot \phi_i(\vec{y}, \vec{x}, t)$$

Where t and i are token and feature indexes, ϕ_i is a feature function, λ_i is a feature weight and Z is a normalization constant. During training the feature weights λ are set to maximize the conditional log-likelihood of the training data D:

$$ll_D(\vec{\lambda}) = \sum_{(\vec{x},\vec{y}) \in D} \log\, p(\vec{y}|\vec{x}, \vec{\lambda})$$

In order to use numerical optimization we have to calculate the gradient of the log-likelihood, which is a vector of partial derivatives $\partial ll_D(\vec{\lambda})/\partial \lambda_i$. For a training sentence $\vec{x}, \vec{y}$ and a token index t the derivative wrt feature i is given by:

$$\phi_i(\vec{y}, \vec{x}, t) - \sum_{\vec{y}'} \phi_i(\vec{y}', \vec{x}, t)\, p(\vec{y}'|\vec{x}, \vec{\lambda})$$

This is the difference between the empirical feature count in the training data and the estimated count in the current model $\vec{\lambda}$. For a 1^{st}-order model, we can replace the expensive sum over all possible tag sequences $\vec{y}'$ by a sum over all pairs of tags:

$$\phi_i(y_t, y_{t+1}, \vec{x}, t) - \sum_{y,y'} \phi_i(y, y', \vec{x}, t)\, p(y, y'|\vec{x}, \vec{\lambda})$$

The probability of a tag pair $p(y, y'|\vec{x}, \vec{\lambda})$ can then be calculated efficiently using the forward-backward algorithm. If we further reduce the complexity of the model to a 0-order model, we obtain simple maximum entropy model updates:

$$\phi_i(y_t, \vec{x}, t) - \sum_{y} \phi_i(y, \vec{x}, t)\, p(y|\vec{x}, \vec{\lambda})$$

2.2 Pruned CRF Training

As we discussed in the introduction, we want to decode sentences by applying a variant of coarse-to-fine tagging. Naively, to later tag with n^{th}-order

[1] Our java implementation MarMoT is available at https://code.google.com/p/cistern/

accuracy we would train a series of n CRFs of increasing order. We would then use the CRF of order $n-1$ to restrict the input of the CRF of order n. In this paper we approximate this approach, but do so while training only one integrated model. This way we can save both memory (by sharing feature weights between different models) and training time (by saving lower-order updates).

The main idea of our approach is to create increasingly complex lattices and to filter candidate states at every step to prevent a polynomial increase in lattice size. The first step is to create a 0-order lattice, which as discussed above, is identical to a series of independent local maximum entropy models $p(y|\vec{x},t)$. The models base their prediction on the current word x_t and the immediate lexical context. We then calculate the posterior probabilities and remove states y with $p(y|\vec{x},t) < \tau_0$ from the lattice, where τ_0 is a parameter. The resulting reduced lattice is similar to what we would obtain using candidate selection based on an MA.

We can now create a first order lattice by adding transitions to the pruned lattice and pruning with threshold τ_1. The only difference to 0-order pruning is that we now have to run forward-backward to calculate the probabilities $p(y|\vec{x},t)$. Note that in theory we could also apply the pruning to transition probabilities of the form $p(y,y'|\vec{x},t)$; however, this does not seem to yield more accurate models and is less efficient than state pruning.

For higher-order lattices we merge pairs of states into new states, add transitions and prune with threshold τ_i.

We train the model using l_1-regularized Stochastic Gradient Descent (SGD) (Tsuruoka et al., 2009). We would like to create a cascade of increasingly complex lattices and update the weight vector with the gradient of the last lattice. The updates, however, are undefined if the gold sequence is pruned from the lattice. A solution would be to simply reinsert the gold sequence, but this yields poor results as the model never learns to keep the gold sequence in the lower-order lattices. As an alternative we perform the gradient update with the highest lattice still containing the gold sequence. This approach is similar to "early updating" (Collins and Roark, 2004) in perceptron learning, where during beam search an update with the highest scoring partial hypothesis is performed whenever the gold candidate falls out of the beam. Intuitively, we are trying to optimize an n^{th}-order CRF objective function, but apply small lower-order corrections to the weight vector when necessary to keep the gold candidate in the lattice. Figure 1 illustrates the lattice generation process. The lattice generation during decoding is identical, except that we always return a lattice of the highest order n.

```
1: function GetSumLattice(sentence, τ⃗)
2:     gold-tags ← getTags(sentence)
3:     candidates ← getAllCandidates(sentence)
4:     lattice ← ZeroOrderLattice(candidates)
5:     for i = 1 → n do
6:         candidates ← lattice.prune(τ_{i−1})
7:         if gold-tags ∉ candidates then
8:             return lattice
9:         end if
10:        if i > 1 then
11:            candidates ← mergeStates(candidates)
12:        end if
13:        candidates ← addTransitions(candidates)
14:        lattice ← SequenceLattice(candidates, i)
15:    end for
16:    return lattice
17: end function
```

Figure 1: Lattice generation during training

The savings in training time of this integrated approach are large; e.g., training a maximum entropy model over a tagset of roughly 1800 tags and more than half a million instances is slow as we have to apply 1800 weight vector updates for every token in the training set and every SGD iteration. In the integrated model we only have to apply 1800 updates when we lose the gold sequence during filtering. Thus, in our implementation training a 0-order model for Czech takes roughly twice as long as training a 1^{st}-order model.

2.3 Threshold Estimation

Our approach would not work if we were to set the parameters τ_i to fixed predetermined values; e.g., the τ_i depend on the size of the tagset and should be adapted during training as we start the training with a uniform model that becomes more specific. We therefore set the τ_i by *specifying μ_i, the average number of tags per position that should remain in the lattice after pruning*. This also guarantees stable lattice sizes and thus stable training times. We

achieve stable average number of tags per position by setting the τ_i dynamically during training: we measure the real average number of candidates per position $\hat{\mu}_i$ and apply corrections after processing a certain fraction of the sentences of the training set. The updates are of the form:

$$\tau_i = \begin{cases} +0.1 \cdot \tau_i & \text{if } \hat{\mu}_i < \mu_i \\ -0.1 \cdot \tau_i & \text{if } \hat{\mu}_i > \mu_i \end{cases}$$

Figure 2 shows an example training run for German with $\mu_0 = 4$. Here the 0-order lattice reduces the number of tags per position from 681 to 4 losing roughly 15% of the gold sequences of the development set, which means that for 85% of the sentences the correct candidate is still in the lattice. This corresponds to more than 99% of the tokens. We can also see that after two iterations only a very small number of 0-order updates have to be performed.

2.4 Tag Decomposition

As we discussed before for the very large POS+MORPH tagsets, most of the decoding time is spent on the 0-order level. To decrease the number of tag candidates in the 0-order model, we decode in two steps by separating the fully specified tag into a coarse-grained part-of-speech (POS) tag and a fine-grained MORPH tag containing the morphological features. We then first build a lattice over POS candidates and apply our pruning strategy. In a second step we expand the remaining POS tags into all the combinations with MORPH tags that were seen in the training set. We thus build a sequence of lattices of both increasing order and increasing tag complexity.

2.5 Feature Set

We use the features of Ratnaparkhi (1996) and Manning (2011): the current, preceding and succeeding words as unigrams and bigrams and for rare words prefixes and suffixes up to length 10, and the occurrence of capital characters, digits and special characters. We define a rare word as a word with training set frequency ≤ 10. We concatenate every feature with the POS and MORPH tag and every morphological feature. E.g., for the word "der", the POS tag `art` (article) and the MORPH tag `gen|sg|fem` (genitive, singular, feminine) we get the following features for the current word template: `der+art`, `der+gen|sg|fem`, `der+gen`, `der+sg` and `der+fem`.

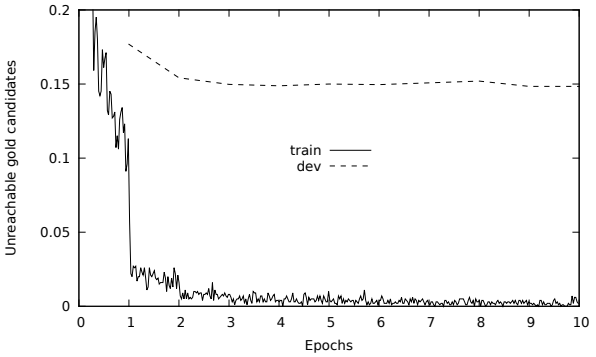


Figure 2: Example training run of a pruned 1^{st}-order model on German showing the fraction of pruned gold sequences (= sentences) during training for training (train) and development sets (dev).

We also use an additional binary feature, which indicates whether the current word has been seen with the current tag or – if the word is rare – whether the tag is in a set of open tag classes. The open tag classes are estimated by 10-fold cross validation on the training set: We first use the folds to estimate how often a tag is seen with an unknown word. We then consider tags with a relative frequency $\geq 10^{-4}$ as open tag classes. While this is a heuristic, it is safer to use a "soft" heuristic as a feature in the lattice than a hard constraint.

For some experiments we also use the output of a morphological analyzer (MA). In that case we simply use every analysis of the MA as a simple nominal feature. This approach is attractive because it does not require the output of the MA and the annotation of the treebank to be identical; in fact, it can even be used if treebank annotation and MA use completely different features.

Because the weight vector dimensionality is high for large tagsets and productive languages, we use a hash kernel (Shi et al., 2009) to keep the dimensionality constant.

3 Related Work

Smith et al. (2005) use CRFs for POS+MORPH tagging, but use a morphological analyzer for candidate selection. They report training times of several days

and that they had to use simplified models for Czech.

Several methods have been proposed to reduce CRF training times. Stochastic gradient descent can be applied to reduce the training time by a factor of 5 (Tsuruoka et al., 2009) and without drastic losses in accuracy. Lavergne et al. (2010) make use of feature sparsity to significantly speed up training for moderate tagset sizes (< 100) and huge feature spaces. It is unclear if their approach would also work for huge tag sets (> 1000).

Coarse-to-fine decoding has been successfully applied to CYK parsing where full dynamic programming is often intractable when big grammars are used (Charniak and Johnson, 2005). Weiss and Taskar (2010) develop cascades of models of increasing complexity in a framework based on perceptron learning and an explicit trade-off between accuracy and efficiency.

Kaji et al. (2010) propose a modified Viterbi algorithm that is still optimal but depending on task and especially for big tag sets might be several orders of magnitude faster. While their algorithm can be used to produce fast decoders, there is no such modification for the forward-backward algorithm used during CRF training.

4 Experiments

We run POS+MORPH tagging experiments on Arabic (ar), Czech (cs), Spanish (es), German (de) and Hungarian (hu). The following table shows the type-token (T/T) ratio, the average number of tags of every word form that occurs more than once in the training set ($\overline{A}$) and the number of tags of the most ambiguous word form ($\hat{A}$):

	T/T	$\overline{A}$	$\hat{A}$
ar	0.06	2.06	17
cs	0.13	1.64	23
es	0.09	1.14	9
de	0.11	2.15	44
hu	0.11	1.11	10

Arabic is a Semitic language with nonconcatenative morphology. An additional difficulty is that vowels are often not written in Arabic script. This introduces a high number of ambiguities; on the other hand it reduces the type-token ratio, which generally makes learning easier. In this paper, we work with the transliteration of Arabic provided in the Penn Arabic Treebank. Czech is a highly inflecting Slavic language with a large number of morphological features. Spanish is a Romance language. Based on the statistics above we can see that it has few POS+MORPH ambiguities. It is also the language with the smallest tagset and the only language in our setup that – with a few exceptions – does not mark case. German is a Germanic language and – based on the statistics above – the language with the most ambiguous morphology. The reason is that it only has a small number of inflectional suffixes. The total number of nominal inflectional suffixes for example is five. A good example for a highly ambiguous suffix is "en", which is a marker for infinitive verb forms, for the 1^{st} and 3^{rd} person plural and for the polite 2^{nd} person singular. Additionally, it marks plural nouns of all cases and singular nouns in genitive, dative and accusative case.

Hungarian is a Finno-Ugric language with an agglutinative morphology; this results in a high type-token ratio, but also the lowest level of word form ambiguity among the selected languages.

POS tagging experiments are run on all the languages above and also on English.

4.1 Resources

For Arabic we use the Penn Arabic Treebank (Maamouri et al., 2004), parts 1–3 in their latest versions (LDC2010T08, LDC2010T13, LDC2011T09). As training set we use parts 1 and 2 and part 3 up to section `ANN20020815.0083`. All consecutive sections up to `ANN20021015.0096` are used as development set and the remainder as test set. We use the unvocalized and pretokenized transliterations as input. For Czech and Spanish, we use the CoNLL 2009 data sets (Hajič et al., 2009); for German, the TIGER treebank (Brants et al., 2002) with the split from Fraser et al. (2013); for Hungarian, the Szeged treebank (Csendes et al., 2005) with the split from Farkas et al. (2012). For English we use the Penn Treebank (Marcus et al., 1993) with the split from Toutanova et al. (2003).

We also compute the possible POS+MORPH tags for every word using MAs. For Arabic we use the AraMorph reimplementation of Buckwalter (2002), for Czech the "free" morphology (Hajič, 2001), for Spanish Freeling (Padró and Stanilovsky, 2012), for German DMOR (Schiller, 1995) and for Hungarian

Magyarlanc 2.0 (Zsibrita et al., 2013).

4.2 Setup

To compare the training and decoding times we run all experiments on the same test machine, which features two Hexa-Core Intel Xeon X5680 CPUs with 3,33 GHz and 6 cores each and 144 GB of memory. The baseline tagger and our PCRF implementation are run single threaded.[2] The taggers are implemented in different programming languages and with different degrees of optimization; still, the run times are indicative of comparative performance to be expected in practice.

Our Java implementation is always run with 10 SGD iterations and a regularization parameter of 0.1, which for German was the optimal value out of $\{0, 0.01, 0.1, 1.0\}$. We follow Tsuruoka et al. (2009) in our implementation of SGD and shuffle the training set between epochs. All numbers shown are averages over 5 independent runs. Where not noted otherwise, we use $\mu_0 = 4$, $\mu_1 = 2$ and $\mu_2 = 1.5$. We found that higher values do not consistently increase performance on the development set, but result in much higher training times.

4.3 POS Experiments

In a first experiment we evaluate the speed and accuracy of CRFs and PCRFs on the POS tagsets. As shown in Table 1 the tagset sizes range from 12 for Czech and Spanish to 54 and 57 for German and Hungarian, with Arabic (38) and English (45) in between. The results of our experiments are given in Table 2. For the 1^{st}-order models, we observe speed-ups in training time from 2.3 to 31 at no loss in accuracy. For all languages, training pruned higher-order models is faster than training unpruned 1^{st}-order models and yields more accurate models. Accuracy improvements range from 0.08 for Hungarian to 0.25 for German. We can conclude that for small and medium tagset sizes PCRFs give substantial improvements in both training and decoding speed[3] and thus allow for higher-order tagging, which for all languages leads to significant[4] accuracy improvements.

4.4 POS+MORPH Oracle Experiments

Ideally, for the full POS+MORPH tagset we would also compare our results to an unpruned CRF, but our implementation turned out to be too slow to do the required number of experiments. For German, the model processed ≈ 0.1 sentences per second during training; so running 10 SGD iterations on the 40,472 sentences would take more than a month. We therefore compare our model against models that perform oracle pruning, which means we perform standard pruning, but always keep the gold candidate in the lattice. The oracle pruning is applied during training and testing on the development set. The oracle model performance is thus an upper bound for the performance of an unpruned CRF.

The most interesting pruning step happens at the 0-order level when we reduce from hundreds of candidates to just a couple. Table 3 shows the results for 1^{st}-order CRFs.

We can roughly group the five languages into three groups: for Spanish and Hungarian the damage is negligible, for Arabic we see a small decrease of 0.07 and only for Czech and German we observe considerable differences of 0.14 and 0.37. Surprisingly, doubling the number of candidates per position does not lead to significant improvements.

We can conclude that except for Czech and German losses due to pruning are insignificant.

4.5 POS+MORPH Higher-Order Experiments

One argument for PCRFs is that while they might be less accurate than standard CRFs they allow to train higher-order models, which in turn might be more accurate than their standard lower-order counterparts. In this section, we investigate how big the improvements of higher-order models are. The results are given in the following table:

n	ar	cs	es	de	hu
1	90.90	92.45	97.95	88.96	96.47
2	91.86*	93.06*	98.01	90.27*	96.57*
3	91.88*	92.97*	97.87	90.60*	96.50

[2]Our tagger might actually use more than one core because the Java garbage collection is run in parallel.

[3]Decoding speeds are provided in an appendix submitted separately.

[4]Throughout the paper we establish significance by running approximate randomization tests on sentences (Yeh, 2000).

	n	ar		cs		es		de		hu		en	
		TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC
CRF	1	106	96.21	10	98.95	7	98.51	234	97.69	374	97.63	154	97.05
PCRF	1	5	96.21	4	98.96	3	98.52	7	97.70	12	97.64	5	97.07
PCRF	2	6	96.43*	5	99.01*	3	98.65*	9	97.91*	13	97.71*	6	97.21*
PCRF	3	6	96.43*	6	99.03*	4	98.66*	9	97.94*	14	97.69	6	97.19*

Table 2: POS tagging experiments with pruned and unpruned CRFs with different orders n. For every language the training time in minutes (TT) and the POS accuracy (ACC) are given. * indicates models significantly better than CRF (first line).

		ar	cs	es	de	hu
1	Oracle $\mu_0 = 4$	90.97	92.59	97.91	89.33	96.48
2	Model $\mu_0 = 4$	90.90	92.45*	97.95	88.96*	96.47
3	Model $\mu_0 = 8$	90.89	92.48*	97.94	88.94*	96.47

Table 3: Accuracies for models with and without oracle pruning. * indicates models significantly worse than the oracle model.

We see that 2^{nd}-order models give improvements for all languages. For Spanish and Hungarian we see minor improvements ≤ 0.1.

For Czech we see a moderate improvement of 0.61 and for Arabic and German we observe substantial improvements of 0.96 and 1.31. An analysis on the development set revealed that for all three languages, case is the morphological feature that benefits most from higher-order models. A possible explanation is that case has a high correlation with syntactic relations and is thus affected by long-distance dependencies.

German is the only language where fourgram models give an additional improvement over trigram models. The reason seem to be sentences with long-range dependencies, e.g., "Die Rebellen haben kein Lösegeld verlangt" (The rebels have not demanded any ransom); "verlangt" (demanded) is a past particple that is separated from the auxilary verb "haben" (have). The 2^{nd}-order model does not consider enough context and misclassifies "verlangt" as a finite verb form, while the 3^{rd}-order model tags it correctly.

We can also conclude that the improvements for higher-order models are always higher than the loss we estimated in the oracle experiments. More precisely we see that if a language has a low number of word form ambiguities (e.g., Hungarian) we observe a small loss during 0-order pruning but we also have to expect less of an improvement when increasing the order of the model. For languages with a high number of word form ambiguities (e.g., German) we must anticipate some loss during 0-order pruning, but we also see substantial benefits for higher-order models.

Surprisingly, we found that higher-order PCRF models can also avoid the pruning errors of lower-order models. Here is an example from the German data. The word "Januar" (January) is ambiguous: in the training set, it occurs 108 times as dative, 9 times as accusative and only 5 times as nominative. The development set contains 48 nominative instances of "Januar" in datelines at the end of news articles, e.g., "TEL AVIV, 3. Januar". For these 48 occurrences, (i) the oracle model in Table 3 selects the *correct* case nominative, (ii) the 1^{st}-order PCRF model selects the *incorrect* case accusative, and (iii) the 2^{nd}- and 3^{rd}-order models select – unlike the 1^{st}-order model – the *correct* case nominative. Our interpretation is that the correct nominative reading is pruned from the 0-order lattice. However, the higher-order models can put less weight on 0-order features as they have access to more context to disambiguate the sequence. The lower weights of order-0 result in a more uniform posterior distribution and the nominative reading is not pruned from the lattice.

4.6 Experiments with Morph. Analyzers

In this section we compare the improvements of higher-order models when used with MAs. The re-

	ar		cs		es		de		hu		en	
	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC
SVMTool	178	96.39	935	98.94	64	98.42	899	97.29	2653	97.42	253	97.09
Morfette	9	95.91	6	99.00	3	98.43	16	97.28	30	97.53	17	96.85
CRFSuite	4	96.20	2	99.02	2	98.40	8	97.57	15	97.48	8	96.80
Stanford	29	95.98	8	**99.08**	7	98.53	51	97.70	40	97.53	65	**97.24**
PCRF 1	5	96.21*	4	98.96*	3	98.52	7	97.70	12	97.64*	5	97.07*
PCRF 2	6	**96.43**	5	99.01*	3	98.65*	9	97.91*	13	**97.71***	6	97.21
PCRF 3	6	**96.43**	6	99.03	4	**98.66***	9	**97.94***	14	97.69*	6	97.19

Table 4: Development results for POS tagging. Given are training times in minutes (TT) and accuracies (ACC). Best baseline results are underlined and the overall best results bold. * indicates a significant difference (positive or negative) between the best baseline and a PCRF model.

	ar	cs	es	de	hu	en
SVMTool	**96.19**	98.82	98.44	96.44	97.32	97.12
Morfette	95.55	98.91	98.41	96.68	97.28	96.89
CRFSuite	95.97	98.91	98.40	96.82	97.32	96.94
Stanford	95.75	**98.99**	98.50	97.09	97.32	**97.28**
PCRF 1	96.03*	98.83*	98.46	97.11	97.44*	97.09*
PCRF 2	96.11	98.88*	**98.66***	97.36*	**97.50***	97.23
PCRF 3	96.14	98.87*	**98.66***	**97.44***	97.49*	97.19*

Table 5: Test results for POS tagging. Best baseline results are underlined and the overall best results bold. * indicates a significant difference between the best baseline and a PCRF model.

	ar		cs		es		de		hu	
	TT	ACC	TT	ACC	TT	ACC	TT	ACC	TT	ACC
SVMTool	454	89.91	2454	89.91	64	97.63	1649	85.98	3697	95.61
RFTagger	4	89.09	3	90.38	1	97.44	5	87.10	10	95.06
Morfette	132	89.97	539	90.37	63	97.71	286	85.90	540	95.99
CRFSuite	309	89.33	9274	91.10	69	97.53	1295	87.78	5467	95.95
PCRF 1	22	90.90*	301	92.45*	25	97.95*	32	88.96*	230	96.47*
PCRF 2	26	91.86*	318	**93.06***	32	**98.01***	37	90.27*	242	**96.57***
PCRF 3	26	**91.88***	318	92.97*	35	97.87*	37	**90.60***	241	96.50*

Table 6: Development results for POS+MORPH tagging. Given are training times in minutes (TT) and accuracies (ACC). Best baseline results are underlined and the overall best results bold. * indicates a significant difference between the best baseline and a PCRF model.

	ar	cs	es	de	hu
SVMTool	89.58	89.62	97.56	83.42	95.57
RFTagger	88.76	90.43	97.35	84.28	94.99
Morfette	89.62	90.01	97.58	83.48	95.79
CRFSuite	89.05	90.97	97.60	85.68	95.82
PCRF 1	90.32*	92.31*	97.82*	86.92*	96.22*
PCRF 2	**91.29***	92.94*	**97.93***	88.48*	**96.34***
PCRF 3	91.22*	**92.99***	97.82*	**88.58***	96.29*

Table 7: Test results for POS+MORPH tagging. Best baseline results are underlined and the overall best results bold. * indicates a significant difference between the best baseline and a PCRF model.

sults are given in the following table:

	n	ar	cs	es	de	hu
	1	90.90^-	92.45^-	97.95^-	88.96^-	96.47^-
	2	91.86^+	93.06	98.01^-	90.27^+	96.57^-
	3	91.88^+	92.97^-	97.87^-	90.60^+	96.50^-
MA	1	91.22	93.21	98.27	89.82	97.28
MA	2	92.16^+	93.87^+	98.37^+	91.31^+	97.51^+
MA	3	92.14^+	93.88^+	98.28	91.65^+	97.48^+

Plus and minus indicate models that are significantly better or worse than MA1. We can see that the improvements due to higher-order models are orthogonal to the improvements due to MAs for all languages. This was to be expected as MAs provide additional lexical knowledge while higher-order models provide additional information about the context. For Arabic and German the improvements of higher-order models are bigger than the improvements due to MAs.

4.7 Comparison with Baselines

We use the following baselines: SVMTool (Giménez and Màrquez, 2004), an SVM-based discriminative tagger; RFTagger (Schmid and Laws, 2008), an n-gram Hidden Markov Model (HMM) tagger developed for POS+MORPH tagging; Morfette (Chrupała et al., 2008), an averaged perceptron with beam search decoder; CRFSuite (Okazaki, 2007), a fast CRF implementation; and the Stanford Tagger (Toutanova et al., 2003), a bidirectional Maximum Entropy Markov Model. For POS+MORPH tagging, all baselines are trained on the concatenation of POS tag and MORPH tag. We run SVMTool with the standard feature set and the optimal c-values $\in \{0.1, 1, 10\}$. Morfette is run with the default options. For CRFSuite we use l_2-regularized SGD training. We use the optimal regularization parameter $\in \{0.01, 0.1, 1.0\}$ and stop after 30 iterations where we reach a relative improvement in regularized likelihood of at most 0.01 for all languages. The feature set is identical to our model except for some restrictions: we only use concatenations with the full tag and we do not use the binary feature that indicates whether a word-tag combination has been observed. We also had to restrict the combinations of tag and features to those observed in the training set[5]. Otherwise the memory requirements would exceed the memory of our test machine (144 GB) for Czech and Hungarian. The Stanford Tagger is used as a bidirectional 2^{nd}-order model and trained using OWL-BFGS. For Arabic, German and English we use the language specific feature sets and for the other languages the English feature set.

Development set results for POS tagging are shown in Table 4. We can observe that Morfette, CRFSuite and the PCRF models for different orders have training times in the same order of magnitude. For Arabic, Czech and English, the PCRF accuracy is comparable to the best baseline models. For the other languages we see improvements of 0.13 for Spanish, 0.18 for Hungarian and 0.24 for German. Evaluation on the test set confirms these results, see Table 5.[6]

The POS+MORPH tagging development set results are presented in Table 6. Morfette is the fastest discriminative baseline tagger. In comparison with Morfette the speed up for 3^{rd}-order PCRFs lies between 1.7 for Czech and 5 for Arabic. Morfette gives the best baseline results for Arabic, Spanish and Hungarian and CRFSuite for Czech and German. The accuracy improvements of the best PCRF models over the best baseline models range from 0.27 for Spanish over 0.58 for Hungarian, 1.91 for Arabic, 1.96 for Czech to 2.82 for German. The test set experiments in Table 7 confirm these results.

5 Conclusion

We presented the pruned CRF (PCRF) model for very large tagsets. The model is based on coarse-to-fine decoding and stochastic gradient descent training with early updating. We showed that for moderate tagset sizes of $\approx$ 50, the model gives significant speed-ups over a standard CRF with negligible losses in accuracy. Furthermore, we showed that training and tagging for approximated trigram and fourgram models is still faster than standard 1^{st}-order tagging, but yields significant improvements in accuracy.

In oracle experiments with POS+MORPH tagsets we demonstrated that the losses due to our approximation depend on the word level ambiguity of the respective language and are moderate (≤ 0.14) except for German where we observed a loss of 0.37.

[5] We set the CRFSuite option $\mathrm{possible\_states} = 0$

[6] Giménez and Màrquez (2004) report an accuracy of 97.16 instead of 97.12 for SVMTool for English and Manning (2011) an accuracy of 97.29 instead of 97.28 for the Stanford tagger.

We also showed that higher order tagging – which is prohibitive for standard CRF implementations – yields significant improvements over unpruned 1^{st}-order models. Analogous to the oracle experiments we observed big improvements for languages with a high level of POS+MORPH ambiguity such as German and smaller improvements for languages with less ambiguity such as Hungarian and Spanish.

Acknowledgments

The first author is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellowship. This research was also funded by DFG (grant SFB 732).

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. *Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.*

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL.*

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of LREC*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL.*

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged treebank. In *Proceedings of Text, Speech and Dialogue*.

Richárd Farkas, Veronika Vincze, and Helmut Schmid. 2012. Dependency parsing of Hungarian: Baseline results and challenges. In *Proceedings of EACL.*

Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze. 2013. Knowledge Sources for Constituent Parsing of German, a Morphologically Rich and Less-Configurational Language. *Computational Linguistics*.

Jesús Giménez and Lluis Màrquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of LREC*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL.*

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of NAACL.*

Jan Hajič. 2001. Czech "Free" Morphology. *URL http://ufal.mff.cuni.cz/pdt/Morphology_and_Tagging*.

Nobuhiro Kaji, Yasuhiro Fujiwara, Naoki Yoshinaga, and Masaru Kitsuregawa. 2010. Efficient staggered decoding for sequence labeling. In *Proceedings of ACL.*

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML.*

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of ACL.*

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *Proceedings of NEMLAR*.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*. Springer.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*.

Naoaki Okazaki. 2007. Crfsuite: A fast implementation of conditional random fields (CRFs). *URL http://www.chokkan.org/software/crfsuite*.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards Wider Multilinguality. In *Proceedings of LREC*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*.

Alexander M. Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of NAACL.*

Anne Schiller. 1995. DMOR Benutzerhandbuch. *Universität Stuttgart, Institut für maschinelle Sprachverarbeitung*.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of COLING*.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of NEMLP*.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In *Proceedings of ACL*.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash Kernels for Structured Data. *J. Mach. Learn. Res.*

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of EMNLP*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *Proceedings of ACL*.

David Weiss and Ben Taskar. 2010. Structured prediction cascades. In *In Proceedings of AISTATS*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING*.

János Zsibrita, Veronika Vincze, and Richárd Farkas. 2013. Magyarlanc 2.0: Szintaktikai elemzés és felgyorsított szófaji egyértelműsítés. In *IX. Magyar Számítógépes Nyelvészeti Konferencia*.

The Effects of Syntactic Features in Automatic Prediction of Morphology

Wolfgang Seeker and **Jonas Kuhn**
Institute for Natural Language Processing
University of Stuttgart
{seeker,jonas}@ims.uni-stuttgart.de

Abstract

Morphology and syntax interact considerably in many languages and language processing should pay attention to these interdependencies. We analyze the effect of syntactic features when used in automatic morphology prediction on four typologically different languages. We show that predicting morphology for languages with highly ambiguous word forms profits from taking the syntactic context of words into account and results in state-of-the-art models.

1 Introduction

In this paper, we investigate the interplay between syntax and morphology with respect to the task of assigning morphological descriptions (or tags) to each token of a sentence. Specifically, we examine the effect of syntactic information when it is integrated into the feature model of a morphological tagger. We test the effect of syntactic features on four languages – Czech, German, Hungarian, and Spanish – and find that syntactic features improve our tagger considerably for Czech and German, but not for Hungarian and Spanish. Our analysis of constructions that show morpho-syntactic agreement suggests that syntactic features are important if the language shows frequent word form syncretisms[1] that can be disambiguated by the syntactic context.

The meaning of a sentence is structurally encoded by morphological and syntactic means.[2] Different languages, however, use them to a different extent. Languages like English encode grammatical information (like the subject vs object status of an argument) via word order, whereas languages like Czech or Hungarian use different word forms. Automatic analysis of languages with rich morphology needs to pay attention to the interaction between morphology and syntax in order to arrive at suitable computational models. Linguistic theory (e. g., Bresnan (2001), Melčuk (2009)) suggests many interactions between morphology and syntax. For example, languages with a case system use different forms of the same word to mark different syntactic (or semantic) relations (Blake, 2001). In many languages, two words that participate in a syntactic relation show covariance in some or all of their morphological features (so-called agreement, Corbett (2006)).[3]

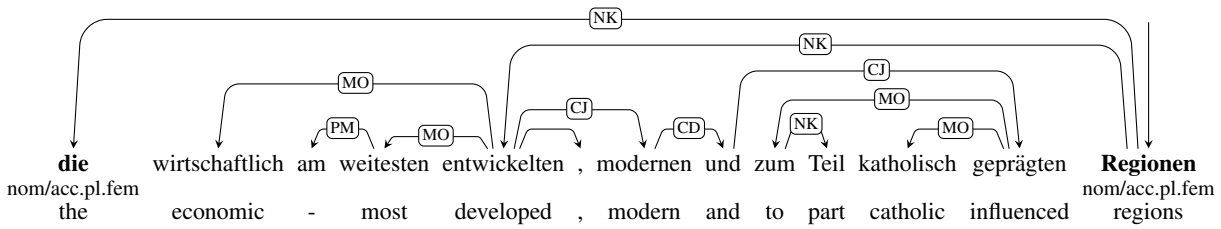
Automatic annotation of morphology assigns morphological descriptions (e. g., *nominative-singular-masculine*) to word forms. It is usually modeled as a sequence model, often in combination with part-of-speech tagging and lemmatization (Collins, 2002; Hajič, 2004; Smith et al., 2005; Chrupała et al., 2008, and others). Sequence models achieve high accuracy and coverage but since they only use linear context they only approximate some of the underlying hierarchical relationships. As an example for these hierarchical relationships,

[1] Syncretism describes the situation where a word form is ambiguous between several different morphological descriptions within its inflection paradigm.

[2] And also by prosodic means, which we will not discuss since text-based tools rarely have access to this information.

[3] For example, in English, the subject of a sentence and the finite verb agree with respect to their number and person feature.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 333–344,
Seattle, Washington, USA, 18-21 October 2013.



'the regions that are economically most developed, modern, and partly catholic'

Figure 1: Example of a German noun phrase. First and last word agree in number, gender, and case value.

Figure 1 shows a German noun phrase taken from the German TiGer corpus (Brants et al., 2002). The two bold-faced words are the determiner and the head noun of the phrase, and they agree in their gender, number, and case values. The word *Regionen* (regions) is four-way ambiguous for its case value, which is reduced to a two-way ambiguity between *nominative* and *accusative* by the determiner. Further disambiguation would require information about the syntactic role of the noun phrase in a sentence. There are 11 tokens between these two words, which would require a context window of at least 13 to capture the agreement relation within a sequence model. Syntactically, however, as indicated by the dependency tree, the determiner and the head are linked directly. The interdependency between morphology and syntax in the example thus manifests itself in the morphological disambiguation of a highly syncretic word form because of its government or agreement relation to its respective syntactic head/dependents.

Of course, the sequence model is most of the time a reasonable approximation, because the majority of noun phrases in the TiGer corpus are not as long as the example in Figure 1.[4] Furthermore, not all languages show this kind of relationship between morphological forms and syntactic relation as demonstrated for German. But taking advantage of the morphosyntactic dependencies in a language can give us better models that may even be capable of handling the more difficult or rare cases. We therefore advocate that models for predicting morphology should be designed with the typological characteristics of a language and its morphosyntactic properties in mind, and should, where appropriate, integrate syntactic information in order to better model the morphosyntactic interdependencies of the language.

In the remainder of the paper, we show empirically that taking syntactic information into account produces state-of-the-art models for languages with a high interdependency between morphology and syntax. We use a simple setup, where we combine a morphological tagger and a dependency parser in a bootstrapping architecture in order to analyze the effect of syntactic information on the performance of the morphological tagger (Section 2). Using syntactic features in morphology prediction requires a syntactically annotated corpus for training a statistical parser, which may not be available for languages with few resources. We show in Section 3 that only very little syntactically annotated data is required to achieve the improvements. We furthermore expect that the improved morphological information also improves parsing performance and present a preliminary experiment in Section 4.

2 Experiments

In this section, we present a series of experiments that investigate the effect of syntactic information on the prediction of morphological features. We start by describing our data sets and the system that we used for the experiments.

2.1 Languages and Data Sets

We test our hypotheses on four different languages: Czech, German, Hungarian, and Spanish.

Spanish, a Romance language, and German, a Germanic language, constitute inflecting languages that show verbal and nominal morphology, but not as sophisticated as Czech and Hungarian. As we will see in the experiments, it is relatively easy to

[4] We find 57,551 noun phrases with less than three tokens between determiner and noun and 4,670 with three or more.

predict the morphological information annotated in the Spanish data set.

Czech and Hungarian represent languages with very rich morphological systems both in verbal and nominal morphological paradigms. They differ significantly in the way in which morphological information is encoded in word forms. Czech, a Slavic language, is an inflecting language, where one suffix may signal several different morphological categories simultaneously (e. g., number, gender, case). In contrast, Hungarian, a Finno-Ugric language, is of the agglutinating type, where each morphological category is marked by its own morpheme.

Both German and Czech show various form syncretisms in their inflection paradigms. Form syncretisms emerge when the same word form is ambiguous between several different morphological descriptions, and they are a major challenge to automatic morphological analysis. Spanish shows syncretism in the verbal inflection paradigms. In Hungarian, form syncretisms are much less frequent. The case paradigm of Hungarian only shows one form syncretism between dative and genitive case (out of about 18 case suffixes).

All languages show agreement between subject and verb, and within the noun phrase. The word order in Czech and Hungarian is very variable whereas it is more restrictive in Spanish and German.

As our data, we use the CoNLL 2009 Shared Task data sets (Hajič et al., 2009) for Czech and Spanish. For German, we use the dependency conversion of the TiGer treebank by Seeker and Kuhn (2012), splitting it into 40k/5k/5k sentences for training/development/test. For Hungarian, we use the Szeged Dependency Treebank (Vincze et al., 2010), with the split of Farkas et al. (2012).

2.2 System Description

To test our hypotheses, we implemented a tagger that assigns full morphological descriptions to each token in a sentence. The system was inspired by the morphological tagger included in mate-tools.[5] Like the tagger provided with mate-tools, it is a classifier that tags each token using the surrounding tokens in its feature model. Models are trained using passive-aggressive online training (Crammer et al., 2003). The system makes two passes over each sentence: The first pass provides predicted tags that are used as features during the second pass. We also adopted the idea of a tag filter, which deterministically assigns tags for words that always occur with the same tag in the training data.

For all matters of syntactic annotation in this paper, we use the graph-based dependency parser by Bohnet (2010), also included in mate-tools. All data sets are annotated with gold syntactic information, which is used to train the parsing models.

For our experiments, we use a bootstrapping approach: the parser uses the output of the morphology in its feature set, and the morphological tagger we want to analyze uses the output of the parser as syntactic features. Since it is best to keep the training setting as similar as possible to the test setting, we use 10-fold jackknifing to annotate our training data with predicted morphology or syntax respectively.

Jackknifing differs from cross-validation only in its purpose. Cross-validation is used for evaluating data, jackknifing is used to annotate data. The data set is split into n parts, and n-1 parts are used to train a model for annotating the n^{th} part. This is then rotated n times such that each part is annotated by the automatic tool without training it on its own test data. Jackknifing is important for creating a realistic training scenario that provides automatic preprocessing. For annotating development and test sets, models are trained on the jackknifed training set.

2.3 The Effects of Syntactic Features

In the first experiment, we use the system described in Section 2.2 to predict morphological information on all four languages. We start with describing the general setup and the feature set, and continue with a discussion of the results.

The experimental setup is as follows: the German and Spanish data sets are annotated with lemma and part-of-speech information using 10-fold jackknifing. The annotation is done with mate-tools' lemmatizer and pos-tagger. For Czech and Hungarian, we keep the annotation provided with the data sets.

Note that our experimental setup does not include lemmas or part-of-speech tags as part of the prediction of the morphology but annotates them in a pre-

[5] A collection of language independent, data-driven analysis tools for lemmatization, pos-tagging, morphological analysis, and dependency parsing: http://code.google.com/p/mate-tools

processing step. It is not necessary to separate part-of-speech and lemma from the prediction of morphology and, in fact, many systems perform these steps simultaneously (e. g. Spoustová et al. (2009)). Doing morphology prediction as a separate step allows us to use lemma and part-of-speech information in the feature set.[6]

static features		
form	form1b	form2b
form3b	form1a	lemma2a
pos1b	pos2b	pos1a
form+pos	pos+s1	pos+s2
pos+s3	pos+s4	lemma+p2
lemma+p3	pos+number	form+form1b
pos+pos1a	pos+pos1b+pos2b	s1+s1_1b
s1+s1_1a	s2+s2_1a	last-verb-lemma
last-verb-pos	next-verb-lemma	next-verb-pos
dynamic features		
tag1b+tag2b	tag2b+tag3b	tag1a
tag1a+tag1b	tag1a+tag2a	tag2a+tag3a
pos1b+case1b	last-verb-tag	next-verb-tag
pos1b+case1b+pos2b+case2b		
Hungarian only features		
pos+uppercase		
Czech only features		
pos+p2		
Spanish only features		
s5	p1	p4
p5	s2_1a	s3_1a
s4_1a		

Table 1: Baseline feature set. *form* means word form, *lemma* is lemma, *pos* is part-of-speech, *s1*/*p1* stand for suffix and prefix of length 1 (characters), *tag* is the morphological tag predicted by the system, *1b*/*1a* means 1 token before/after the current token, and + marks feature conjunctions. *number* marks if the form contains a digit.

After preprocessing the data, our baseline system is trained using the feature set shown in Table 1. The baseline system does not make use of any syntactic information but predicts morphological information based solely on tokens and their linear context. The features are divided into static features, which can be computed on the input, and dynamic features, which are computed also on previous output of the system (cf. two passes in Section 2.2).

[6]Lemma and part-of-speech prediction may also profit from syntactic information, see e.g. Prins (2004) or Bohnet and Nivre (2012).

The feature sets in Table 1 were developed specifically for our experiments and are the result of an automatic forward/backward feature selection process. The purpose of the feature selection was to arrive at a baseline system that performs well without any syntactic information. With such an optimized baseline system, we can measure the contribution of syntactic features more reliably.

The *last-verb*/*next-verb* and *pos+case* features are variants of the features proposed in Votrubec (2006). They extract information about the first verb within the last 10/the next 30 tokens in the sentence. The *case* feature extracts the case value from previously assigned morphological tags. Note that the *verb* features are approximating syntactic information by making the assumption that the closest verbs are likely to be syntactic heads for many words.

static features				
h_lemma	h_s2	h_s3	pos+h_pos	s1+h_s1
h_dir	h_dir+h_pos			
ld_s1	ld_s2	ld_p1	ld_p4	
dynamic features				
h_tag	ld_tag			

Table 2: Syntactic features. *h_* and *ld_* mark features from the head and the left-most daughter, *dir* is a binary feature marking the direction of the head with respect to the current token.

After training the baseline models, we use them to annotate the whole data set with morphological information (using 10-fold jackknifing for the training portions). We then use 10-fold jackknifing again to annotate the data sets with the dependency parser.

At this point, all our data sets are annotated with predicted morphology from our baseline system and with syntactic information from the parser, which uses the morphological information from our baseline system in its feature set. We can now retrain our morphological tagger using features that are derived from the dependency trees provided by the parser. Note that this is not a stacking architecture, since the second system does not use the predicted morphology output from the baseline system. The loop simply ensures that we get the best possible syntactic features.

We extract two kinds of syntactic features: features of the syntactic head of the current token, and

	dev set		test set	
	all	oov	all	oov
Czech				
morfette	90.37	68.66	90.01	67.25
our baseline	92.51	73.12	92.29	72.58
pred syntax	*93.18	74.04	*92.82	73.11
gold syntax	*93.64	75.20	*93.30	74.96
German				
morfette	86.78	66.37	84.58	61.05
our baseline	90.92	72.52	89.11	69.67
pred syntax	*92.07	75.06	*90.10	71.18
gold syntax	*92.70	*76.29	*90.87	*73.20
Hungarian				
morfette	*96.19	*85.82	95.99	*85.43
our baseline	96.08	84.49	95.94	83.76
pred syntax	96.18	84.70	96.11	83.85
gold syntax	*96.46	85.30	*96.35	84.50
Spanish				
morfette	97.83	89.67	97.76	91.00
our baseline	97.83	89.05	97.59	90.88
pred syntax	97.84	89.08	97.67	90.91
gold syntax	98.11	90.34	97.88	91.61

Table 3: The effect of syntactic features when predicting morphological information. * mark statistically significantly better models compared to our baseline (sentence-based t-test with $\alpha = 0.05$).

	dev set		test set	
	all	oov	all	oov
Czech				
featurama	94.75	84.12	94.78	84.23
our baseline	93.80	80.47	93.57	80.53
pred syntax	*94.40	81.51	*94.24	81.61
gold syntax	*94.80	82.45	*94.64	82.80
German				
RFTagger	90.63	72.11	89.04	70.80
our baseline	92.59	80.73	91.48	78.83
pred syntax	*93.70	82.71	*92.51	80.20
gold syntax	*94.28	*84.12	*93.32	*82.35
Hungarian				
our baseline	97.27	92.61	97.03	91.28
pred syntax	97.38	92.39	97.19	91.50
gold syntax	*97.63	92.79	*97.45	91.92
Spanish				
our baseline	98.23	92.46	98.02	93.15
pred syntax	98.24	92.30	98.07	93.03
gold syntax	98.40	92.82	*98.22	93.64

Table 4: The effect of syntactic features when predicting morphology **using lexicons**. * mark statistically significantly better models compared to our baseline (sentence-based t-test with $\alpha = 0.05$).

features of the left-most daughter of the current token. We also experimented with other types, e. g. the right-most daughter, but these features did not improve the model. This is likely due to the way these languages encode morphological information and may be different for other languages. From the head and the left-most daughter, we construct features about form, lemma, affixes, and tags. Table 2 lists the syntactic features that we use in the model.

With the syntactic features available due to the parsing step, we train new models with the full system. For each language, we run four experiments. The first two are baseline experiments, where we use the off-the-shelf morphological tagger morfette (Chrupała et al., 2008) and our own baseline system, both of which do not use any syntactic features. In the third experiment, we evaluate our full system using the syntactic features provided by the dependency parser. As an oracle experiment, we also report results on the full system when using the gold standard syntax from the treebank. Table 3 presents all results in terms of accuracy on all tokens (all) and out-of-vocabulary tokens only (oov). Out-of-vocabulary tokens do not occur in the training data.

We find trends along several axes: Generally, the syntactic features work well for Czech and German, whereas for Hungarian and Spanish, they do not yield any significant improvement. The improvements for German and Czech are between 0.5 (Czech) and 1.0 (German) percentage points absolute in token accuracy, and between 0.2 (Czech test set) and 2.5 (German dev set) percentage points absolute in accuracy of unknown words. There are no obvious differences between the development and the test set in any of the languages.

Compared to the morfette baseline, we find our systems to be either superior or equal to morfette in terms of token accuracy. Regarding accuracy on unknown words, morfette outperforms our systems for Hungarian, but is outperformed on Czech and German. For Spanish, all systems yield similar results.

Looking at the oracle experiment, we see that for all languages, the system can learn something from syntax. For Czech and German, this is clearly the

case, for Hungarian and Spanish, the differences are small but visible. There are pronounced differences between the predicted and the gold syntax experiments in Czech and German. Clearly, the parser makes mistakes that propagate through to the prediction of the morphology.

2.4 Syntax vs Lexicon

The current state-of-the-art in predicting morphological features makes use of morphological lexicons (e.g. Hajič (2000), Hakkani-Tür et al. (2002), Hajič (2004)). Lexicons define the possible morphological descriptions of a word and a statistical model selects the most probable one among them. In the following experiment, we test whether the contribution of syntactic features is similar or different to the contribution of morphological lexicons.

Lexicons encode important knowledge that is difficult to pick up in a purely statistical system, e. g. the gender of nouns, which often cannot be deduced from the word form (Corbett, 1991).[7]

We extend our system from the previous experiment to include information from a morphological dictionaries. For Czech, we use the morphological analyzer distributed with the Prague Dependency Treebank 2 (Hajič et al., 2006). For German, we use DMor (Schiller, 1994). For Hungarian, we use (Trón et al., 2006), and for Spanish, we use the morphological analyzer included in Freeling (Carreras et al., 2004). The output of the analyzers is given to the system as features that simply record the presence of a particular morphological analysis for the current word. The system can thus use the output of any tool regardless of its annotation scheme, especially if the annotation scheme of the treebank is different from the one of the morphological analyzer.

Table 4 presents the results of experiments where we add the output of the morphological analyzers to our system. Again, we run experiments with and without syntactic features. For Czech, we also show results from featurama[8] with the feature set developed by Votrubec (2006). For German, we show results for RFTagger (Schmid and Laws, 2008).

As expected, the information from the morphological lexicon improves the overall performance considerably compared to the results in Table 3, especially on unknown tokens. This shows that even with the considerable amounts of training data available nowadays, rule-based morphological analyzers are important resources for morphological description (cf. Hajič (2000)). The contribution of syntactic features in German and Czech is almost the same as in the previous experiment, indicating that the syntactic features contribute information that is orthogonal to that of the morphological lexicon. The lexicon provides lexical knowledge about a word form, while the syntactic features provide the syntactic context that is needed in German and Czech to decide on the right morphological tag.

2.5 Language Differences

From the previous experiments, we conclude that syntactic features help in the prediction of morphology for Czech and German, but not for Hungarian and Spanish. To further investigate the difference between Czech and German on the one hand, and Hungarian and Spanish on the other, we take a closer look at the output of the tagger.

We find an interesting difference between the two pairs of languages, namely the performance with respect to agreement. Agreement is a phenomenon where morphology and syntax strongly interact. Morphological features co-vary between two items in the sentence, but the relation between these items can occur at various linguistic levels (Corbett, 2006). If the syntactic information helps with predicting morphological information, we expect this to be particularly helpful with getting agreement right. All languages show agreement to some extent. Specifically, all languages show agreement in number (and person) between the subject and the verb of a clause. Czech, German, and Spanish show agreement in number, gender, and case (not Spanish) within a noun phrase. Hungarian shows case agreement within the noun phrase only rarely, e.g. for attributively used demonstrative pronouns.

In order to test the effect on agreement, we measure the accuracy on tokens that are in an agreement relation with their syntactic head. We counted subject verb agreement as well as agreement with respect to number, gender, and case (where applicable) between a noun and its dependent adjective and determiner. Table 5 displays the counts from the devel-

[7] Lexicons are also often used to speed up processing considerably by restricting the search space of the statistical model.

[8] http://sourceforge.net/projects/featurama/

opment sets of each language. We compare the baseline system that does not use any syntactic information with the output of the morphological tagger that uses the gold syntax. We use the gold syntax rather than the predicted one in order to eliminate any influence from parsing errors. As can be seen from the results, the level of agreement relations in Czech and German improves when using syntactic information, whereas in Spanish and Hungarian, only very tiny changes occur.

agreement	baseline	gold syntax
	Czech	
sbj-verb	3199/4044 = 79.10	3264/4044 = 80.71
NP case	8719/9132 = 95.48	8821/9132 = 96.59
NP num	8933/9132 = 97.82	9016/9132 = 98.73
NP gen	8493/9132 = 93.00	8768/9132 = 96.01
	German	
sbj-verb	4412/4696 = 93.95	4562/4696 = 97.15
NP case	13340/13951 = 95.62	13510/13951 = 96.84
NP num	13631/13951 = 97.71	13788/13951 = 98.83
NP gen	13253/13951 = 95.00	13528/13951 = 96.97
	Hungarian	
sbj-verb	8653/10219 = 84.68	8655/10219 = 84.70
NP case	402/891 = 45.12	412/891 = 46.24
	Spanish	
sbj-verb	1930/2004 = 96.31	1932/2004 = 96.41
NP num	8810/8849 = 99.56	8816/8849 = 99.63
NP gen	8810/8849 = 99.56	8821/8849 = 99.68

Table 5: Agreement counts in morphological annotation compared between the baseline system and the oracle system using gold syntax.

For Czech and German, these results sugguest that syntactic information helps with agreement. We believe that the reasons why it does not help for Hungarian and Spanish are the following: for Spanish, we see that also the baseline model achieves very high accuracies (cf. Table 3) and also high rates of correct agreement. It seems that for Spanish, syntactic context is simply not necessary to make the correct prediction. For Hungarian, the reason lies within the inflectional paradigms of the language, which do not show any form syncretism, meaning that word forms in Hungarian are usually not ambiguous within one morphological category (e.g. case). Making a morphological tag prediction, however, is difficult only if the word form itself is ambiguous between several morphological tags. In this case, using the agreement relation between the word and its syntactic head can help the system making the proper prediction. This is the situation that we find in Czech and German, where form syncretism is pervasive in the inflectional paradigms.

2.6 Syntactic Features in Czech

In Section 2.4 we compared the performance of our system on Czech to another system, featurama (see Table 4). Featurama outperforms our baseline system by a percentage point in token accuracy (and even more for unknown tokens). Syntactic information closes that gap to a large extent but only using gold syntax gets our system on a par with featurama.

The question then arises whether the syntactic features actually contribute something new to the task, or whether the same effect could also be achieved with linear context features alone as in featurama. In order to test this we run an additional experiment, where we add some of the syntax features to the feature set of featurama. Specifically, we add the static features from Table 2 that do not use lemma or part-of-speech information. Due to the way featurama works, we cannot use features from the morphological tags (the dynamic features).

The results in Table 6 show that also featurama profits from syntactic features, which corroborates the findings from the previous experiments. We also note again that better syntax would improve results even more.

	dev set		test set	
	all	oov	all	oov
featurama	94.75	84.12	94.78	84.23
pred syntax	95.18	84.65	95.09	84.52
gold syntax	*95.39	84.62	*95.34	85.03

Table 6: Syntactic features for featurama (Czech). * mark statistically significantly better models compared to featurama (sentence-based t-test with $\alpha = 0.05$).

3 How Much Syntax is Needed?

Syntactic features require syntactically annotated corpora. Without a treebank to train the parser, the morphology cannot profit from syntactic features.[9] This may be problematic for languages for which there is no treebank, because creating a treebank is expensive. Fortunately, it turns out that very small amounts of syntactically annotated data are enough

[9] Which is of course only a problem for statistical parsers.

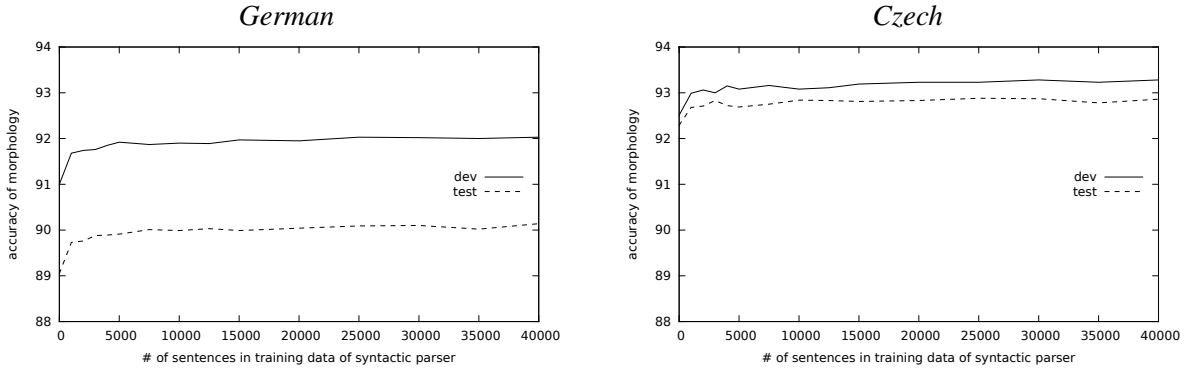


Figure 2: Dependency between amount of training data for syntactic parser and quality of morphological prediction.

to provide a parsing quality that is sufficient for the morphological tagger.

In order to test what amount of training data is needed, we train several parsing models on increasing amounts of syntactically annotated data. For example, the first experiment uses the first 1,000 sentences of the treebank. We perform 5-fold jackknifing with the parser on these sentences to annotate them with syntax. Then we train one parsing model on these 1,000 sentences and use it to annotate the rest of the training data as well as the development and the test set. This gives us the full data set annotated with syntax that was learned from the first 1,000 sentences of the treebank. The morphological tagger is then trained on the full training set and applied to development and test set.

Figure 2 shows the dependency between the amount of training data given to the parser and the quality of the morphological tagger using syntactic features provided by this parser. The left-most point corresponds to a model that does not use syntactic information. For both languages, German and Czech, we find that already 1,000 sentences are enough training data for the parser to provide useful syntactic information to the morphological tagger. After 5,000 sentences, both curves flatten out and stay on the same level. We conclude that using syntactic features for morphological prediction is viable even if there is only small amounts of syntactic data available to train the parser.

As a related experiment, we also test if we can get the same effect with a very simple and thus much faster parser. We use the brute-force algorithm described in Covington (2001), which selects for each token in the sentence another token as the head. It does not have any tree requirements, so it is not even guaranteed to yield a cycle-free tree structure. In Table 7, we compare the simple parser with the mate-parser, both trained on the first 5,000 sentences of the treebank. Evaluation is done in terms of labeled (LAS) and unlabeled attachment score (UAS).[10]

	dev set		test set	
	LAS	UAS	LAS	UAS
Czech				
simple parser (5k)	71.57	78.96	69.09	77.23
full parser (5k)	76.77	84.38	74.70	83.00
German				
simple parser (5k)	83.06	85.23	78.56	81.18
full parser (5k)	87.56	90.08	83.69	86.58

Table 7: Simple parser vs full parser – syntactic quality. Trained on first 5,000 sentences of the training set.

As expected, the simple parser performs much worse in terms of syntactic quality. Table 8 shows the performance of the morphological tagger when using the output of both parsers as syntactic features. For Czech, both parsers seem to supply similar information to the morphological tagger, while for German, using the full parser is clearly better. In both cases, the morphological tagger outperforms the models that do not use syntactic information (cf. Table 3). The performance on unknown words is however much worse for both languages. We conclude that even with a simple parser and little training data, the morphology can make use of syntactic information to some extent.

[10] LAS: $\frac{\text{correct edges with correct labels}}{\text{all edges}}$, UAS: $\frac{\text{correct edges}}{\text{all edges}}$

	dev set		test set	
	all	oov	all	oov
Czech				
no syntax	92.51	73.12	92.29	72.58
simple syntax	92.96	73.45	92.53	72.66
full syntax	93.08	73.64	92.69	73.39
German				
no syntax	90.92	72.52	89.11	69.67
simple syntax	91.52	73.34	89.66	70.52
full syntax	91.92	83.46	89.91	80.50

Table 8: Simple parser vs full parser – morphological quality. The parsing models were trained on the first 5,000 sentences of the training data, the morphological tagger was trained on the full training set.

4 Does Better Morphology lead to Better Parses?

In the previous sections, we show that syntactic information improves a model for predicting morphology for Czech and German, where syntax and morphology interact considerably. A natural question then is whether the improvement also occurs in the other direction, namely whether the improved morphology also leads to better parsing models.

In the previous experiments, we run a 10-fold jackknifing process to annotate the training data with morphological information using no syntactic features and afterwards use jackknifing with the parser to annotate syntax. The syntax is subsequently used as features for our predicted-syntax experiments. We can apply the same process once more with the morphology prediction in order to annotate the training data with morphological information that is predicted using the syntactic features. A parser trained on this data will then use the improved morphology as features. If the improved morphology has an impact on the parser, the quality of the second parsing model should then be superior to the first parsing model, which uses the morphology predicted without syntactic information. Note that for the following experiments, neither morphology model uses the morphological lexicon.

Table 9 presents the evaluation of the two parsing models (one using morphology without syntactic features, the other one using the improved morphology). The results show no improvement in parsing performance when using the improved morphology. Looking closer at the output, we find differences between the two parsing models with respect to grammatical functions that are morphologically marked. For example, in German, performance on subjects and accusative objects improves while performance for dative objects and genitives decreases. This suggests different strengths in the two parsing models. However, the question how to make use of the improved morphology in parsing clearly needs more research in the future. A promising avenue may be the approach by Hohensee and Bender (2012).

	dev set		test set	
	LAS	UAS	LAS	UAS
Czech				
baseline morph	81.73	88.45	81.02	87.77
morph w/ syntax	81.63	88.37	80.83	87.61
German				
baseline morph	91.16	92.97	88.06	90.24
morph w/ syntax	91.20	92.97	88.15	90.34

Table 9: Impact of the improved morphology on the quality of the dependency parser for Czech and German.

5 Related Work

Morphological taggers have been developed for many languages. The most common approach is the combination of a morphological lexicon with a statistical disambiguation model (Hakkani-Tür et al., 2002; Hajič, 2004; Smith et al., 2005; Spoustová et al., 2009; Zsibrita et al., 2013).

Our work has been inspired by Versley et al. (2010), who annotate a treebank with morphological information after the syntax had been annotated already. The system used a finite-state morphology to propose a set of candidate tags for each word, which is then further restricted using hand-crafted rules over the already available syntax tree.

Lee et al. (2011) pursue the idea of jointly predicting syntax and morphology, out of the motivation that joint models should model the problem more faithfully. They demonstrate that both sides can use information from each other. However, their model is computationally quite demanding and its overall performance falls far behind the standard pipeline approach where both tasks are done in sequence.

The problem of modeling the interaction between morphology and syntax has recently attracted some attention in the SPMRL workshops (Tsarfaty et al.,

2010). Modeling morphosyntactic relations explicitly has been shown to improve statistical parsing models (Tsarfaty and Sima'an, 2010; Goldberg and Elhadad, 2010; Seeker and Kuhn, 2013), but the co-dependency between morphology and syntax makes it a difficult problem, and linguistic intuition is often contradicted by the empirical findings. For example, Marton et al. (2013) show that case information is the most helpful morphological feature for parsing Arabic, but only if it is given as gold information, whereas using case information from an automatic system may even harm the performance.

Morphologically rich languages pose different challenges for automatic systems. In this paper, we work with European languages, where the problem of predicting morphology can be reduced to a tagging problem. In languages like Arabic, Hebrew, or Turkish, widespread ambiguity in segmentation of single words into meaningful morphemes adds an additional complexity. Given a good segmentation tool that takes care of this, our approach is applicable to these languages as well. For Hebrew, this problem has also been addressed by jointly modeling segmentation, morphological prediction, and syntax (Cohen and Smith, 2007; Goldberg and Tsarfaty, 2008; Goldberg and Elhadad, 2013).

6 Conclusion

In this paper, we have demonstrated that using syntactic information for predicting morphological information is helpful if the language shows form syncretism in combination with morphosyntactic phenomena like agreement. A model that uses syntactic information is superior to a sequence model because it leverages the syntactic dependencies that may hold between morphologically dependent words as suggested by linguistic theory. We also showed that only small amounts of training data for a statistical parser would be needed to improve the morphological tagger. Making use of the improved morphology in the dependency parser is not straight-forward and requires more investigation in the future.

Modeling the interaction between morphology and syntax is important for building successful parsing pipelines for languages with free word order and rich morphology. Moreover, our experiments show that paying attention to the individual properties of a language can help us explain and predict the behavior of automatic tools. Thus, the term "morphologically rich language" should be viewed as a broad term that covers many different languages, whose differences among each other may be as important as the difference with languages with a less rich morphology.

Acknowledgments

We would like to thank Jan Hajič and Jan Štěpánek for their kind help with the Czech morphology and featurama. We would also like to thank Thomas Müller for sharing resources and thoughts with us, and Anders Björkelund for commenting on earlier versions of this paper. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) via SFB 732 "Incremental Specification in Context", project D8.

References

Barry J. Blake. 2001. *Case*. Cambridge University Press, Cambridge, New York, 2nd edition.

Bernd Bohnet and Joakim Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju, South Korea. Association for Computational Linguistics.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China. International Committee on Computational Linguistics.

Sabine Brants, Stefanie Dipper, Silvia Hansen-Shirra, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 24–41, Sozopol, Bulgaria.

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell Publishers.

Xavier Carreras, Isaac Chao, Llus Padr, and Muntsa Padr. 2004. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, pages 239–242. European Language Resources Association (ELRA).

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 2362–2367, Marrakech, Morocco. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 208–217, Prague, Czech Republic. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Greville G. Corbett. 1991. *Gender*. Cambridge Textbooks in Linguistics. Cambridge University Press.

Greville G. Corbett. 2006. *Agreement*. Cambridge Textbooks in Linguistics. Cambridge University Press.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing (with corrections). In *Proceedings of the 39th Annual ACM Southeast Conference*, Athens, Gorgia. ACM.

Koby Crammer, Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. 2003. Online passive-aggressive algorithms. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, volume 7, pages 1217–1224, Cambridge, Massachusetts, USA. MIT Press.

Richárd Farkas, Veronika Vincze, and Helmut Schmid. 2012. Dependency parsing of hungarian: Baseline results and challenges. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 55–65, Avignon, France. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. Easy first dependency parsing of modern Hebrew. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 103–107, Los Angeles, California, USA. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2013. Word segmentation, unknown-word resolution, and morphological agreement in a hebrew parsing system. *Computational Linguistics*, 39(1):121–160.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 371–379, Columbus, Ohio. Association for Computational Linguistics.

Jan Hajič. 2000. Morphological Tagging: Data vs. Dictionaries. In *Proceedings of the 6th ANLP Conference / 1st NAACL Meeting*, pages 94—101, Seattle, Washington. Association for Computational Linguistics.

Jan Hajič. 2004. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, Prague, Czech Republic.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jií Havelka, and Marie Mikulová. 2006. Prague Dependency Treebank 2.0.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Stepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and Semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task*, pages 1–18, Boulder, Colorado, USA. Association for Computational Linguistics.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410.

Matt Hohensee and Emily M. Bender. 2012. Getting more from morphology in multilingual dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 315–326, Montréal, Canada. Association for Computational Linguistics.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th annual meeting of the Association for Computational Linguistics*, pages 885–894, Portland, USA. Association for Computational Linguistics.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Igor Melčuk. 2009. *Dependency in linguistic description*.

Robbert Prins. 2004. Beyond N in N-gram tagging. In Leonoor Van Der Beek, Dmitriy Genzel, and Daniel Midgley, editors, *Proceedings of the ACL 2004 Student Research Workshop*, pages 61–66, Barcelona, Spain. Association for Computational Linguistics.

Anne Schiller. 1994. Dmor - user's guide. Technical report, University of Stuttgart.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 777–784, Morristown, NJ, USA. Association for Computational Linguistics.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3132–3139, Istanbul, Turkey. European Language Resources Association (ELRA).

Wolfgang Seeker and Jonas Kuhn. 2013. Morphological and syntactic case in statistical dependency parsing. *Computational Linguistics*, 39(1):23–55.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 475–482, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Drahomíra "Johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771, Athens, Greece. Association for Computational Linguistics.

Viktor Trón, Péter Halácsy, Péter Rebrus, András Rung, Péter Vajda, and Eszter Simon. 2006. Morphdb.hu: Hungarian lexical database and morphological grammar. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1670–1673, Genoa, Italy.

Reut Tsarfaty and Khalil Sima'an. 2010. Modeling morphosyntactic agreement in constituency-based parsing of Modern Hebrew. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 40–48, Los Angeles, California, USA. Association for Computational Linguistics.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, California, USA. Association for Computational Linguistics.

Yannick Versley, Kathrin Beck, Erhard Hinrichs, and Heike Telljohann. 2010. A syntax-first approach to high-quality morphological analysis and lemma disambiguation for the tba-d/z treebank. In *9th Conference on Treebanks and Linguistic Theories (TLT9)*, pages 233–244.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. In *Proceedings of the 7th Conference on International Language Resources and Evaluation*, pages 1855–1862, Valletta, Malta. European Language Resources Association (ELRA).

Jan Votrubec. 2006. Morphological tagging based on averaged perceptron. In *WDS'06 Proceedings of Contributed Papers*, pages 191–195, Praha, Czechia. Matfyzpress, Charles University.

János Zsibrita, Veronika Vincze, and Richárd Farkas. 2013. magyarlanc 2.0: szintaktikai elemzés és felgyorsított szófaji egyértelmsítés. In Attila Tanács and Veronika Vincze, editors, *IX. Magyar Számítógépes Nyelvészeti Konferencia*, pages 368–374, Szeged, Hungary.

Adaptor Grammars for Learning Non-Concatenative Morphology

Jan A. Botha and **Phil Blunsom**
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
{jan.botha,phil.blunsom}@cs.ox.ac.uk

Abstract

This paper contributes an approach for expressing non-concatenative morphological phenomena, such as stem derivation in Semitic languages, in terms of a mildly context-sensitive grammar formalism. This offers a convenient level of modelling abstraction while remaining computationally tractable. The nonparametric Bayesian framework of adaptor grammars is extended to this richer grammar formalism to propose a probabilistic model that can learn word segmentation and morpheme lexicons, including ones with discontiguous strings as elements, from unannotated data. Our experiments on Hebrew and three variants of Arabic data find that the additional expressiveness to capture roots and templates as atomic units improves the quality of concatenative segmentation and stem identification. We obtain 74% accuracy in identifying triliteral Hebrew roots, while performing morphological segmentation with an F1-score of 78.1.

1 Introduction

Unsupervised learning of morphology is the task of acquiring, from unannotated data, the intra-word building blocks of a language and the rules by which they combine to form words. This task is of interest both as a gateway for studying language acquisition in humans and as a way of producing morphological analyses that are of practical use in a variety of natural language processing tasks, including machine translation, parsing and information retrieval.

A particularly interesting version of the morphology learning problem comes from languages that use *templatic* morphology, such as Arabic, Hebrew and Amharic. These Semitic languages derive verb and noun stems by interspersing abstract *root* morphemes into *templatic* structures in a non-concatenative way. For example, the Arabic root k·t·b can combine with the template (i-a) to derive the noun stem *kitab* (book). Established morphological analysers typically ignore this process and simply view the derived stems as elementary units (Buckwalter, 2002), or their account of it coincides with a requirement for extensive linguistic knowledge and hand-crafting of rules (Finkel and Stump, 2002; Schneider, 2010; Altantawy et al., 2010). The former approach is bound to suffer from vocabulary coverage issues, while the latter clearly does not transfer easily across languages. The practical appeal of unsupervised learning of templatic morphology is that it can overcome these shortcomings.

Unsupervised learning of *concatenative* morphology has received extensive attention, partly driven by the MorphoChallenge (Kurimo et al., 2010) in recent years, but that is not the case for root-templatic morphology (Hammarström and Borin, 2011).

In this paper we present a model-based method that learns concatenative and root-templatic morphology in a unified framework. We build on two disparate strands of work from the literature: Firstly, we apply *simple Range Concatenating Grammars* (SRCGs) (Boullier, 2000) to parse contiguous and discontiguous morphemes from an input string. These grammars are *mildly-context sensitive* (Joshi, 1985), a superset of context-free grammars that retains polynomial parsing time-complexity. Secondly, we generalise the nonparametric Bayesian learning framework of *adaptor grammars* (Johnson et al., 2007) to SRCGs.[1] This should also be rel-

[1]Our formulation is in terms of SRCGs, which are equivalent in power to linear context-free rewrite systems (Vijay-Shanker et al., 1987) and multiple context-free grammars (Seki et al., 1991), all of which are weaker than (non-simple) range concatenating grammars (Boullier, 2000).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 345–356,
Seattle, Washington, USA, 18-21 October 2013.

evant to other applications of probabilistic SRCGs, e.g. in parsing (Maier, 2010), translation (Kaeshammer, 2013) and genetics (Kato et al., 2006).

In addition to unannotated data, our method requires as input a minimal set of high-level grammar rules that encode basic intuitions of the morphology. This is where there would be room to become very language specific. Our aim, however, is not to obtain a best-published result in a particular language, but rather to create a method that is applicable across a variety of morphological processes. The specific rules used in our empirical evaluation on Arabic and Hebrew therefore contain hardly any explicit linguistic knowledge about the languages and are applicable across the family of Semitic languages.

2 A powerful grammar for morphology

Concatenative morphology lends itself well to an analysis in terms of finite-state transducers (FSTs) (Koskenniemi, 1984). With some additional effort, FSTs can also encode non-concatenative morphology (Kiraz, 2000; Beesley and Karttunen, 2003; Cohen-Sygal and Wintner, 2006; Gasser, 2009). Despite this seeming adequacy of regular languages to describe morphology, we see two main shortcomings that motivate moving further up the Chomsky hierarchy of formal languages: first is the issue of learning. We are not aware of successful attempts at inducing FST-based morphological analysers in an unsupervised way, and believe the challenge lies in the fact that FSTs do not offer a *convenient* way of expressing prior linguistic intuitions to guide the learning process. Secondly, an FST composed of multiple machines might capture morphological processes well and excel at analysis, but interpretability of its internal operations are limited.

These shortcomings are overcome for concatenative morphology by context-free adaptor grammars, which allowed diverse segmentation models to be formulated and investigated within a single framework (Johnson et al., 2007; Johnson, 2008; Sirts and Goldwater, 2013). In principle, that covers a wide range of phenomena (typical example language in parentheses): affixal inflection (Czech) and derivation (English), agglutinative derivation (Turkish, Finnish), compounding (German). Our agenda here is to extend that approach to include non-concatenative processes such as root-templatic derivation (Arabic), infixation (Tagalog) and circumfixation (Indonesian). In this pursuit, an abstraction that permits discontiguous constituents is a highly useful modelling tool, but requires looking beyond context-free grammars.

An idealised generative grammar that would capture all the aforementioned phenomena could look like this:

$$\text{Word} \rightarrow (\text{Pre}^* \ \text{Stem} \ \text{Suf}^*)^+ \quad (1)$$

e.g. English *un+accept+able*

$$\text{Stem} \mid \text{Pre} \mid \text{Suf} \rightarrow \text{Morph} \quad (2)$$

$$\text{Stem} \rightarrow \mathbf{intercal}(\text{Root}, \text{Template}) \quad (3)$$

e.g. Arabic derivation $k{\cdot}t{\cdot}b + i{\cdot}a \Rightarrow$ *kitab* (book)

$$\text{Stem} \rightarrow \mathbf{infix}(\text{Stem}, \text{Infix}) \quad (4)$$

e.g. Tagalog *sulat* (write) $\Rightarrow$ ***sum****ulat* (wrote)

$$\text{Stem} \rightarrow \mathbf{circfix}(\text{Stem}, \text{Circumfix}) \quad (5)$$

e.g. Indonesian *percaya* (to trust)

$\Rightarrow$ ***ke****percaya****an*** (belief)

where the symbols (excluding Word and Stem) implicitly expand to the relevant terminal strings. The bold-faced "functions" combine the potentially discontiguous yields of the argument symbols into single contiguous strings, e.g. $\mathbf{infix}$(s·ulat, um) produces stem *sumulat*.

Taken by themselves, the first two rules are simply a CFG that describes word formation as the concatenation of stems and affixes, a formulation that matches the underlying grammar of Morfessor (Creutz and Lagus, 2007), a well-studied unsupervised model.

The key aim of our extension is that we want the grammar to capture a discontiguous string like k·t·b as a single constituent in a parse tree. This leads to well-understood problems in probabilistic grammars (e.g. what is this rule's probability?), but also corresponds to the linguistic consideration that k·t·b is a proper morpheme of the language (Prunet, 2006).

3 Simple range concatenating grammars

In this section we define SRCGs formally and illustrate how they can be used to model non-concatenative morphology. SRCGs define languages that are recognisable in polynomial time, yet can capture discontiguous elements of a string under a single category (Boullier, 2000). An SRCG-

rule operates on vectors of ranges in contrast to the way a CFG-rule operates on single ranges (spans). In other words, a non-terminal symbol in an SRCG (CFG) derivation can dominate a subset (substring) of terminals in an input string.

3.1 Formalism

An SRCG $\mathcal{G}$ is a tuple (N, T, V, P, S), with finite sets of non-terminals (N), terminals (T) and variables (V), with a start symbol $S \in N$. A rewrite rule $p \in P$ of rank $r = \rho(p) \geq 0$ has the form $A(\alpha_1, \ldots, \alpha_{\psi(A)}) \rightarrow B_1(\beta_{1,1}, \ldots, \beta_{1,\psi(B_1)}) \ldots B_r(\beta_{r,1}, \ldots, \beta_{r,\psi(B_r)})$, where each $\alpha, \beta \in (T \cup V)^*$, and $\psi(A)$ is the number of arguments a non-terminal A has, called its *arity*. By definition, the start symbol has arity 1. Any variable $v \in V$ appearing in a given rule must be used exactly once on each side of the rule. Terminating rules are written with ϵ as the right-hand side and thus have rank 0.

A *range* is a pair of integers (i, j) denoting the substring $w_{i+1} \ldots w_j$ of a string $w = w_1 \ldots w_n$. A non-terminal becomes *instantiated* when its variables are bound to ranges through substitution. Variables within an argument imply concatenation and therefore have to bind to adjacent ranges.

An instantiated non-terminal A' is said to derive ϵ if the consecutive application of a sequence of instantiated rules rewrite it as ϵ. A string w is within the language defined by a particular SRCG iff the start symbol S, instantiated with the exhaustive range $(0, w_n)$, derives ϵ.

An important distinction with regard to CFGs is that, due to the instantiation mechanism, the ordering of non-terminals on the right-hand side of an SRCG rule is irrelevant, i.e. $A(ab) \rightarrow B(a)C(b)$ and $A(ab) \rightarrow C(b)B(a)$ are the same rule.[2] Consequently, the isomorphisms of any given SRCG derivation tree all encode the same string, which is uniquely defined through the instantiation process.

3.2 Application to morphological analysis

A fragment of the idealised grammar schema from the previous section (§2) can be rephrased as an SRCG by writing the rules in the newly introduced notation, and supplying a definition of the **intercal** function as simply another rule of the grammar, with instantiation for $w =$ kitab shown below:

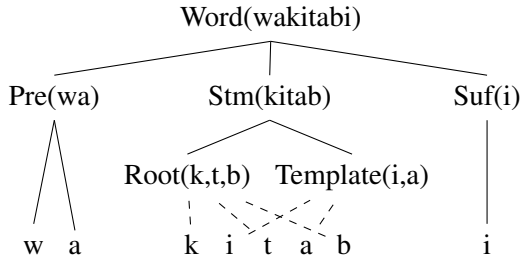


Figure 1: Example derivation for *wakitabi* (and my book) using the SRCG fragment from §3.2. CFGs cannot capture such crossing branches.

$$\text{Word}(abc) \rightarrow \text{Pre}(a)\ \text{Stem}(b)\ \text{Suf}(c)$$
$$\text{Stem}(abcde) \rightarrow \text{Root}(a, c, e)\ \text{Template}(b, d),$$

$$\text{Stem}(\langle 0..1\rangle, \langle 1..2\rangle, \langle 2..3\rangle, \langle 3..4\rangle, \langle 4..5\rangle) \rightarrow \text{Root}(\langle 0..1\rangle, \langle 2..3\rangle, \langle 4..5\rangle)\ \text{Template}(\langle 1..2\rangle, \langle 3..4\rangle)$$

Given an appropriate set of grammar rules (as we present in §5), we can parse an input string to obtain a tree as shown in Figure 1. The overlapping branches of the tree demonstrate that this grammar captures something a CFG could not. From the parse tree one can read off the word's root morpheme and the template used.

Although SRCGs specify mildly context-sensitive grammars, each step in a derivation is context-free – a node's expansion does not depend on other parts of the tree. This property implies that a recognition/parsing algorithm can have a worst-case time complexity that is polynomial in the input length n, $O(n^{(\rho+1)\psi})$ for arity ψ and rank ρ, which reduces to $O(n^{3\psi})$ for a binarised grammar. To capture the maximal case of a root with $k-1$ characters and k discontiguous templatic characters forming a stem would require a grammar that has arity $\psi = k$. For Arabic, which has up to quadriliteral roots ($k = 5$), the time complexity would be $O(n^{15})$.[3] This is a daunting proposition for parsing, but we are careful

[2] Certain ordering restrictions over the variables *within* an argument need to hold for an SRCG to indeed be a *simple* RCG (Boullier, 2000).

[3] The trade-off between arity and rank with respect to parsing complexity has been characterised (Gildea, 2010), and the appropriate refactoring may bring down the complexity for our grammars too.

to set up our application of SRCGs in such a way that this is not too big an obstacle:

Firstly, our grammars are defined over the characters that make up a word, and not over words that make up a sentence. As such, the input length n would tend to be shorter than when parsing full sentences from a corpus.

Secondly, we do type-based morphological analysis, a view supported by evidence from Goldwater et al. (2006), so each unique word in a dataset is only ever parsed once with a given grammar. The set of word types attested in the data sources of interest here is fairly limited, typically in the tens of thousands. For these reasons, our parsing and inference tasks turn out to be tractable despite the high time complexity.

4 Learning

4.1 Probabilistic SRCG

The probabilistic extension of SRCGs is similar to the probabilistic extension of CFGs, and has been used in other guises (Kato et al., 2006; Maier, 2010). Each rule $r \in P$ has an associated probability θ_r such that $\sum_{r \in P_A} \theta_r = 1$. A random string in the language of the grammar can then be obtained through a generative procedure that begins with the start symbol S and iteratively expands it until deriving ϵ: At each step for some current symbol A, a rewrite rule r is sampled randomly from P_A in accordance with the distribution over rules and used to expand A. This procedure terminates when no further expansions are possible. Of course, expansions need to respect the range concatenating and ordering constraints imposed by the variables in rules. The expansions imply a chain of variable bindings going down the tree, and instantiation happens only when rewriting into ϵs but then propagates back up the tree.

The probability $P(w, t)$ of the resulting tree t and terminal string w is the product $\prod_r \theta_r$ over the sequence of rewrite rules used. This generative procedure is a conceptual device; in practice, one would care about parsing some input string under this probabilistic grammar.

4.2 PYSRCAG

A central property of the generative procedure underlying probabilistic SRCGs is the fact that each expansion happens independently, both of the other expansions in the tree under construction and of any other trees. To some extent, this flies in the face of the reality of estimating a grammar from text, where one would expect certain sub-trees to be used repeatedly across different input strings.

Adaptor grammars weaken this independence assumption by allowing whole subtrees to be reused during expansion. Informally, they act as a cache of tree fragments whose tendency to be reused during expansion is governed by the choice of adaptor function. Following earlier applications of adaptor grammars (Johnson et al., 2007; Huang et al., 2011), we employ the Pitman-Yor process (Pitman, 1995; Pitman and Yor, 1997) as adaptor function.

A Pitman-Yor Simple Range Concatenating Adaptor Grammar (PYSRCAG) is a tuple $\mathcal{G} = (\mathcal{G}_\mathcal{S}, M, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha})$, where $\mathcal{G}_\mathcal{S}$ is a probabilistic SRCG as defined before and $M \subseteq N$ is a set of *adapted non-terminals*. The vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, indexed by the elements of M, are the discount and concentration parameters for each adapted non-terminal, with $a \in [0, 1], b \geq 0$. $\boldsymbol{\alpha}$ are parameters to Dirichlet priors on the rule probabilities θ.

PYSRCAG defines a generative process over a set of trees $\boldsymbol{T}$. Unadapted non-terminals $A' \in N \setminus M$ are expanded as before (§4.1). For each adapted non-terminal $A \in M$, a cache C_A is maintained for storing the terminating tree fragments expanded from A earlier in the process, and we denote the fragment corresponding to the i-th expansion of A as z_i. In other words, the sequence of indices z_i is the assignment of a sequence of expansions of A to particular tree fragments. Given a cache C_A that has n previously generated trees comprising m unique trees each used $n_1, \ldots, n_m$ times (where $n = \sum_k n_k$), the tree fragment for the next expansion of A, z_{n+1}, is sampled conditional on the previous assignments $\boldsymbol{z}_<$ according to

$$z_{n+1}|\boldsymbol{z}_< \sim \begin{cases} \frac{n_k - a}{n+b} & \text{if } z_{n+1} = k \in [1, m] \\ \frac{ma+b}{n+b} & \text{if } z_{n+1} = m+1, \end{cases}$$

where a and b are those elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ corresponding to A. The first case denotes the situation where a previously cached tree is reused for this $n+1$-th expansion of A; to be clear, this expands A with a fully terminating tree fragment, meaning that none of the nodes descending from A in the

tree being generated are subject to further expansion. The second case by-passes the cache and expands A according to the rules P_A and rule probabilities θ_A of the underlying SRCG $\mathcal{G}_S$. Other caches $C_B(B \in M)$ may come into play during those expansions of the descendants of A; thus a PYSRCAG can define a hierarchical stochastic process. Both cases eventually result in a terminating tree-fragment for A, which is then added to the cache, updating the counts n, n_{z_n+1} and potentially m.

The adaptation does not affect the string language of $\mathcal{G}_S$, but it maps the distribution over trees to one that is distributed according to the PYP.

The invariance of SRCGs trees under isomorphism would make the probabilistic model deficient, but we side-step this issue by requiring that grammar rules are specified in a canonical way that ensures a one-to-one correspondence between the order of nodes in a tree and of terminals in the yield.

4.3 Inference under PYSRCAG

The inference procedure under our model is very similar to that of CFG PY-adaptor grammars, so we restate the central aspects here but refer the reader to the original article by Johnson et al. (2007) for further details. First, one may integrate out the adaptors to obtain a single distribution over the set of trees generated from a particular non-terminal. Thus, the joint probability of a particular sequence $\boldsymbol{z}$ for the adapted non-terminal A with cached counts $(n_1, \ldots, n_m)$ is

$$PY(\boldsymbol{z}|a,b) = \frac{\prod_{k=1}^{m}(a(k-1)+b)\ \prod_{j=1}^{n_k-1}(j-a)}{\prod_{i=0}^{n-1}(i+b)}. \tag{6}$$

Taking all the adapted non-terminals into account, the joint probability of a set of full trees $\boldsymbol{T}$ under the grammar $\mathcal{G}$ is

$$P(\boldsymbol{T}|\boldsymbol{a},\boldsymbol{b},\boldsymbol{\alpha}) = \prod_{A\in M} \frac{\mathrm{B}(\boldsymbol{\alpha}_A + \boldsymbol{f}_A)}{\mathrm{B}(\boldsymbol{\alpha}_A)} PY(\boldsymbol{z}(\boldsymbol{T})|\boldsymbol{a},\boldsymbol{b}), \tag{7}$$

where $\boldsymbol{f}_A$ is a vector of the usage counts of rules $r \in P_A$ across $\boldsymbol{T}$, and B is the Euler beta function.

The posterior distribution over a set of *strings* $\boldsymbol{w}$ is obtained by marginalising (7) over all trees that have $\boldsymbol{w}$ as their yields. This is intractable to compute directly, so instead we use MCMC techniques to obtain samples from that posterior using a component-wise Metropolis-Hastings sampler. The sampler works by visiting each string w in turn and drawing a new tree for it under a proposal grammar $\mathcal{G}_Q$ and randomly accepting that as the new analysis for w according to the Metropolis-Hastings accept-reject probability. As proposal grammar, we use the analogous approximation of our $\mathcal{G}$ as Johnson et al. used for PCFGs, namely by taking a static snapshot $\mathcal{G}_Q$ of the adaptor grammar where additional rules rewrite adapted non-terminals as the terminal strings of their cached trees. Drawing a sample from the proposal distribution is then a matter of drawing a random tree from the parse chart of w under G_Q.

Lastly, the adaptor hyperparameters $\boldsymbol{a}$ and $\boldsymbol{b}$ are modelled by placing flat $\mathrm{Beta}(1,1)$ and vague $\mathrm{Gamma}(10, 0.1)$ priors on them, respectively, and inferring their values using slice sampling (Johnson and Goldwater, 2009).

5 Modelling root-templatic morphology

We start with a CFG-based adaptor grammar[4] that models words as a stem and any number of prefixes and suffixes:

$$\text{Word} \rightarrow \underline{\text{Pre}}^* \ \underline{\text{Stem}} \ \underline{\text{Suf}}^* \tag{8}$$

$$\underline{\text{Pre}} \mid \underline{\text{Stem}} \mid \underline{\text{Suf}} \rightarrow \text{Char}^+ \tag{9}$$

This fragment can be seen as building on the stem-and-affix adaptor grammar presented in (Johnson et al., 2007) for morphological analysis of English, of which a later version also covers multiple affixes (Sirts and Goldwater, 2013). In the particular case of Arabic, multiple affixes are required to handle the attachment of particles and proclitics onto base words.

To extend this to *complex stems* consisting of a root with three radicals we have rules like the following:

$$\underline{\text{Stem}}(abcdefg) \rightarrow \underline{\text{R3}}(b,d,e)\ \text{T4}(a,c,e,g) \tag{10}$$

$$\underline{\text{Stem}}(abcdef) \rightarrow \underline{\text{R3}}(a,c,e)\ \text{T3}(b,d,f) \tag{11}$$

$$\underline{\text{Stem}}(abcde) \rightarrow \underline{\text{R3}}(a,c,e)\ \text{T2}(b,d) \tag{12}$$

$$\underline{\text{Stem}}(abcd) \rightarrow \underline{\text{R3}}(a,c,d)\ \text{T1}(b) \tag{13}$$

$$\underline{\text{Stem}}(abc) \rightarrow \underline{\text{R3}}(a,b,c) \tag{14}$$

[4] Adapted non-terminals are indicated by underlining and we use the following abbreviations: X $\rightarrow$ Y$^+$ means one or more instances of Y and encodes the rules X $\rightarrow$ Ys and Ys $\rightarrow$ Ys Y $\mid$ Y. Similarly, X $\rightarrow$ Y* Z allows zero or more instances of Y and encodes the rules X $\rightarrow$ Z and X $\rightarrow$ Y$^+$ Z. Further relabelling is added as necessary to avoid cycles among adapted non-terminals.

The actual rules include certain permutations of these, e.g. rule (13) has a variant $\underline{\text{R3}}(a, b, d)\text{T1}(c)$. In unvocalised text, the standard written form of Modern Standard Arabic (MSA), it may happen that the stem and the root of a word form are one and the same. So while rule (14) may look trivial, it ensures that in such cases the radicals are still captured as descendants of the non-terminal category $\underline{\text{R3}}$, thereby making their appearance in the cache.

A discontiguous non-terminal $\text{A}n$ is rewritten through recursion on its arity down to 1, i.e. $\text{A}n(v_1, \ldots, v_n) \rightarrow \text{A}l(v_1, \ldots, v_{n-1})\ \text{Char}(v_n)$ with base case $\text{A1}(v) \rightarrow \text{Char}(v)$, where Char rewrites all individual terminals as ϵ, v_i are variables and $l = n-1$.[5] Note that although we provide the model with two sets of discontiguous non-terminals R and T, we do not specify their mapping onto the actual terminal strings; no subdivision of the alphabet into vowels and consonants is hard-wired.

6 Experiments

We evaluate our model on standard Arabic, Quranic Arabic and Hebrew in terms of segmentation quality and lexicon induction ability. These languages share various properties, including morphology and lexical cognates, but are sufficiently different so as to require manual intervention when transferring rule-based morphological analysers across languages. A key question in this evaluation is therefore whether an appropriate instantiation of our model successfully generalises across related languages.

6.1 Data sets

Our models are unsupervised and therefore learn from raw text, but their evaluation requires annotated data as a gold-standard and these were derived[6] as follows:

Arabic (MSA) We created the dataset BW by synthesising 50k morphotactically correct word types from the morpheme lexicons and consistency rules supplied with the Buckwalter Arabic Morphological Analyser (BAMA).[7] This allowed control over the word shapes, which is important to focus the evaluation, while yielding reliable segmentation and root annotations. BW has no vocalisation; we denote the corresponding *vocalised* dataset as BW′.

	Types	Stems	Roots	m/w	c/w
BW	48428	24197	4717	2.3	6.4
BW′	48428	30891	4707	2.3	10.7
QU′	18808	12021	1270	1.9	9.9
HEB	5231	3164	492	2.1	6.7

Table 1: Corpus statistics, including average number of morphemes (m/w) and characters (c/w) per word, and total surface-realised roots of length 3 or 4.

Quranic Arabic We extracted the roughly 18k word types from a morphologically analysed version of the Quran (Dukes and Habash, 2010). As an additional challenge, we left all given diacritics intact for this dataset, QU′.

Hebrew We leveraged the Hebrew CHILDES database as an annotated resource (Albert et al., 2013) and were able to extract 5k word types that feature at least one affix to use as dataset HEB. The corrected versions of words marked as non-standard child language were used, diacritics were dropped, and we conflated stressed and unstressed vowels to overcome inconsistencies in the source data.

6.2 Models

We consider two classes of models. The first is the strictly context-free adaptor grammar for morphemes as sequences of characters using rules (8)-(9), which we denote as **Concat** and **MConcat**, where the latter allows multiple prefixes/suffixes in a word. These serve as baselines for the second class in which non-concatenative rules are added. **MTpl** and **Tpl** denote the canonical ver-

[5] Including the arity as part of the non-terminal symbol names forms part of our convention here to ensure that the grammar contains no cycles, a situation which would complicate inference under PYSRCAG.

[6] Our data preprocessing scripts are obtainable from `http://github.com/bothameister/pysrcag-data`.

[7] We used version 2.0, LDC2004L02, and sampled word types having a single stem and at most one prefix, suffix or both, according to the following random procedure: Sample a shape (stem: 0.1, pre+stem: 0.25 stem+suf: 0.25, pre+stem+suf: 0.4). Sample uniformly at random (with replacement) a stem from the BAMA stem lexicon, and affix(es) from the ones consistent with the chosen stem. The BAMA lexicons contain affixes and their legitimate concatenations, so some of the generated words would permit a linguistic segmentation into multiple prefixes/suffixes. Nonetheless, we take as gold-standard segmentation precisely the items used by our procedure.

sions with stems as shown in the set of rules above, and we experiment with a variant **Tpl3Ch** that allows the non-terminal T1 to be rewritten as up to three Char symbols, since the data indicate there are cases where multiple characters intervene between the radicals of a root.

These models exclude rule (10), which we include only in the variant **Tpl+T4**. Lastly, **TplR4** is the extension of **Tpl+T4** to include a stem-forming rule that uses $R4$.

As external baseline model we used **Morfessor** (Creutz and Lagus, 2007), which performs decently in morphological segmentation of a variety of languages, but only handles concatenation.

6.3 Method

The MCMC samplers converged within a few hundred iterations and we collected 100 posterior samples after 900 iterations of burn-in. Collected samples, each of which is a set of parse trees of the input word types, are used in two ways:

First, by averaging over the samples we can estimate the joint probability of a word type w and a parse tree t under the adaptor grammar, conditional on the data and the model's hyperparameters. We take the most probable parse of each word type and evaluate the implied segmentation against the gold standard segmentation. Likewise, we evaluate the implied lexicon of stems, affixes and roots against the corresponding reference sets. It should be emphasised that using this maximally probable analysis is aimed at simplifying the evaluation set-up; one could also extract multiple analyses of a word since the model defines a distribution over them.

The second method abstracts away from individual word-types and instead averages over the union of all samples to obtain an estimate of the probability of a string s being generated by a certain category (non-terminal) of the grammar. In this way we can obtain a lexicon of the morphemes in each category, ranked by their probability under the model.

6.4 Inducing Morpheme Lexicons

The quality of each induced lexicon is measured with standard set-based precision and recall with respect to the corresponding gold lexicon. The results are summarised by balanced F-scores in Table 2.

The main result is that all our models capable of forming complex stems obtain a marked improvement in F-scores over the baseline concatenative adaptor grammar, and the margin of improvement grows along with the expressivity of the complex-stem models tested. This applies across prefix, stem and suffix categories and across our datasets, with the exception of QU′, which we elaborate on in §6.5.

Stem lexicons of Arabic were learnt with relatively constant precision (∼70%), but modelling complex stems broadened the coverage by about 3000 stems over the concatenative model (against a reference set of 24k stems). On vocalised Arabic, the improvements for stems are along both dimensions. In contrast, affix lexicons for both BW and BW′ are noisy and the models all generate greedily to obtain near perfect recall but low precision.

On our Hebrew data, which comprises only 5k words, the gains in lexicon quality from modelling complex stems tend to be larger than on Arabic. This is consistent with our intuition that an appropriate, richer Bayesian prior helps overcome data sparsity.

Extracting a lexicon of roots is rendered challenging by the unsupervised nature of the model as the labelling of grammar symbols is ultimately arbitrary. Our simple approach was to regard a character tuple parsed under category R3 as a root. This had mixed success, as demonstrated by the outlier scores in Table 2. In the one case where it was obvious that T3 had been been co-opted for the role, we report the F-score obtained on the union of R3 and T3 strings.

Soft decisions The preceding set-based evaluation imposes hard decisions about category membership. But adaptor grammars are probabilistic by definition and should thus also be evaluated in terms of probabilistic ability. One method is to turn the model predictions into a binary classifier of strings using Receiver-Operator-Characteristic (ROC) theory. We plot the true positive rate versus the false positive rate for each prediction lexicon L_τ containing strings that have probability greater than τ under the model (for a grammar category of interest). A perfect classifier would rank all true positives (e.g. stem strings) above false positives (e.g. non-stem strings), corresponding to a curve in the upper left corner of the ROC plot. A random guesser would trace a diagonal line. The area under the curves (AUC) is the probability that the classifier would discriminate correctly.

	Vocalised Arabic (BW′)				**Unvocalised Arabic** (BW)				**Hebrew** (HEB)			
	Pre	Stem	Suf	R3	Pre	Stem	Suf	R3	Pre	Stem	Suf	R3
Concat	15.0	20.2	25.4	-	32.8	44.1	40.3	-	18.7	20.9	29.2	-
Tpl	24.7	39.4	35.2	†42.4	45.9	54.7	47.9	62.7	35.1	59.6	52.9	34.8
Tpl3Ch	28.4	36.0	36.5	5.2	50.3	55.1	48.5	62.4	**38.6**	61.5	**56.6**	7.1
Tpl+T4	29.0	44.8	41.0	3.9	46.2	54.2	47.7	62.3	32.5	59.6	53.0	**36.4**
TplR4	**37.8**	**60.3**	**47.0**	5.2	**53.0**	**57.7**	**51.9**	62.4	38.0	**62.4**	55.2	34.7

Table 2: Morpheme lexicon induction quality. F1-scores for lexicons induced from the most probable parse of each different dataset under each models. †42.4 was obtained by taking the union of R3 and T3 items to match the way the model used them (see §6.4).

	BW′	BW	QU′	HEB
Morfessor	55.57	40.04	**44.34**	24.20
Concat	47.36	64.22	19.64	60.05
Tpl	60.42	71.91	22.53	77.26
Tpl3Ch	60.52	72.20	25.72	77.41
Tpl+T4	64.49	71.59	24.81	77.14
TplR4	**74.54**	**73.66**	-	**78.14**

Table 3: Segmentation quality in SBF1. The QU′ results are for the corresponding **M*** models .

Our models with complex stem formation improve over the baseline on the AUC metric too. We include the ROC plots for Hebrew stem and root induction in Figure 2, along with the roots the model was most confident about (Table 4).

6.5 Morphological Analysis per Word Type

In this section we turn to the analyses our models assign to each word type. Two aspects of interest are the segmentation into sequential morphemes and the identification of the root.

Our intercalating adaptor grammars consistently obtain large gains in segmentation accuracy over the baseline concatenative model, across all our datasets (Table 3). We measure segmentation quality as *segment border F1-score* (SBF) (Sirts and Goldwater, 2013), which is the F-score over word-internal segmentation points of the predicted analysis with respect to the gold segmentation.

Of the two MSA datasets, the vocalised version BW′ presents a more difficult segmentation task as its words are on average longer and feature 31k unique contiguous morphemes, compared to the 24k in BW for the same number of words. It should thus benefit more from additional model expressivity, as is reflected in the increase of 10 SBF when adding the **TplR4** rule to the other triliteral ones.

The best triliteral root identification accuracy (on a per-word basis) was found for HEB (74%) and BW (67%).[8,9] Refer to Figure 3 for example analyses.

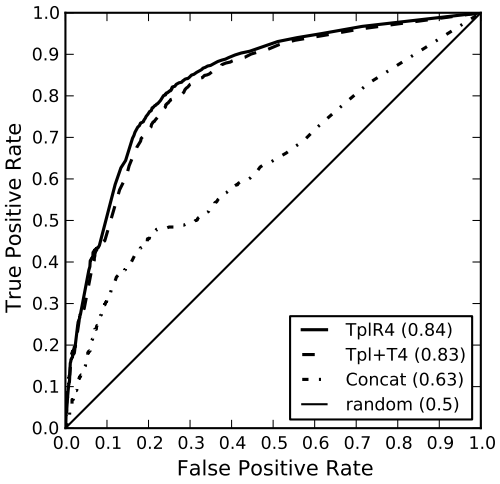
An interesting aspect of these results is that templatic rules may aid segmentation quality without necessarily giving perfect root identification. Modelling stem substructure allows any regularities that give rise to a higher data likelihood to be picked up.

The low performance on the Quran demands further explanation. All our adaptor grammars severely oversegmented this data, although the mistakes were not uniformly distributed. Most of the performance loss is on the 79% of words that have 1-2 morphemes. On the remaining words (having 3-5 morphemes), our models recover and approach the Morfessor baseline (**MConcat**: 32.7 , **MTpl3Ch**: 38.6).

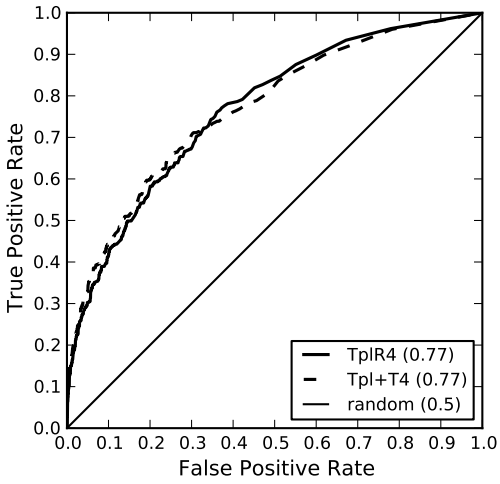
Preliminary experiments on BW had indicated that adaptation of (single) affix categories is crucial for good performance. Our multi-affixing models used on QU′ lacked a further level of adaptation for composite affixes, which we suspect as a contributing factor to the lower performance on that dataset. This remains to be confirmed in future experiments, but would be consistent with other observations on the role of hierarchical adaptation in adaptor grammars (Sirts and Goldwater, 2013). The trend that intercalated rules improve segmentation (compared to the concatenative grammar) remains consistent

[8]When excluding cases where root equals stem, root identification on BW is 55%. Those cases are still not trivial, since words without roots also exist.

[9]By way of comparison, Rodrigues and Ćavar (2007) presented an unsupervised statistics-based root identification method that obtained precision ranging between 50-75%, the higher requiring vocalised words.



(a) Stems



(b) Triliteral roots

Figure 2: ROC curves for predicting the stem and root lexicons for the HEB dataset. The area under each curve (AUC), as computed with the trapezium rule, is given in parentheses.

across datasets, despite the lower absolute performance on QU′.

The performance of the Morfessor baseline was quite mixed. Contrary to our expectations, it performs best on the "harder" BW′, worst on the arguably simpler HEB and struggled less than the adaptor grammars on QU′.

One factor here is that it learns according to a grammar with multiple consecutive affixes and stems, whereas all our experiments (except on QU′) presupposed single affixes. This biases the evaluation slightly in our favour, but works in Morfessor's favour on the QU′ data which is annotated with multiple affixes.

7 Related work

The distinctive feature of our morphological model is that it jointly addresses root identification and morpheme segmentation, and our results demonstrate the mutual benefit of this.

In contrast, earlier unsupervised approaches tend to focus on these tasks in isolation.

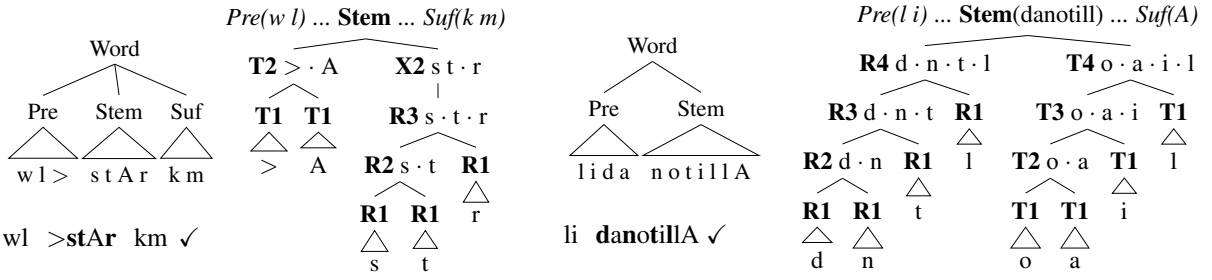
In unsupervised Arabic segmentation, the parametric Bayesian model of (Lee et al., 2011) achieves F1-scores in the high eighties by incorporating sentential context and inferred syntactic categories, both of which our model forgoes, although theirs has no account of discontiguous root morphemes.

Root		Example instances
1. spr ✓	G	**šaǒař**.ti te.**šaǒř** ye.**šaǒř**.u
	B	**sipur**.im hi$_x$**štaǒ**$_x$ař_t
2. lbs ✓	G	**ǐaǐbaŠ**.t li.**ǐǐb**o**Š** ti.**ǐǐb**e**Š**.i
	B	le_ha$_x$**ǐǐb**i**Š** ti_t$_x$**ǐaǐbŠ**.i
3. ptx ✓	G	**ǒaťax**.ti ti.**ǒťex**.i
	B	li.**ǒťoax** ni$_x$**ǒťax**.at
5. !al ×	B	ya.**!al**.u ḿa$_x$**!aǐ**.a **!ač**ǐ$_x$an_it

Table 4: Top Hebrew roots hypothesised by **Tpl+T4**. Numbers indicate position when ranked by model probability. (G)ood and (B)ad instances from the corpus are given with morpheme boundaries marked: true positive (.), false negative (_) and false positive ($_x$). Hypothesised root characters are boldfaced, while accent (ˇ) marks gold root characters.

Previous approaches to Arabic root identification that sought to use little supervision typically constrain the search space of candidate characters within a word, leveraging pre-existing dictionaries (Darwish, 2002; Boudlal et al., 2009) or rule constraints (Elghamry, 2005; Rodrigues and Ćavar, 2007; Daya et al., 2008).

In contrast to these approaches, our model requires no dictionary, and while our grammar rules effect some constraints on what could be a root, they are specified in a convenient and flexible manner that



(a) **Concat** & **TplT4**, "wl>stArkm" (BW) (b) **Concat** & **TplR4**, "lidanotillA" (BW′)

Figure 3: Parse trees produced for words in the two standard Arabic datasets that were incorrectly segmented by the baseline grammar. The templatic grammars correctly identified the triliteral and quadriliteral roots, also fixing the segmentation of (a). In (b), the templatic grammar improved over the baseline by finding the correct prefix but falsely posited a suffix. Unimportant subtrees are elided for space, while the yields of discontiguous constituents are indicated next to their symbols, with dots marking gaps. Crossing branches are not drawn but should be inferrable. Root characters are bold-faced in the reference analysis ✓. The nonterminal X2 in (a) is part of a number of implementation-specific helper rules that ensure the appropriate handling of partly contiguous roots.

makes experimentation with other phenomena easy.

Recent work by Fullwood and O'Donnell (2013) goes some way toward jointly dealing with non-concatenative and concatenative morphology in the unsupervised setting, but their focus is limited to inflected stems and does not handle multiple consecutive affixes. They analyse the Arabic verb stem (e.g. kataba "he wrote") into a templatic bit-string denoting root and non-root characters (e.g. r-r-r-) along with a root morpheme (e.g. ktb) and a so-called residue morpheme (e.g. aaa). Their nonparametric Bayesian model induces lexicons of these entities and achieves very high performance on templates. The explicit formulation of templates alleviates the labelling ambiguity that hampered our evaluation (§6.4), but we believe their method of analysis can be simulated in our framework using the appropriate SRCG-rules.

Learning root-templatic morphology is loosely related to morphological paradigm induction (Clark, 2001; Dreyer and Eisner, 2011; Durrett and DeNero, 2013). Our models do not represent templatic paradigms explicitly, but it is interesting to note that preliminary experiments with German indicate that our adaptor grammars pick up on the past participle forming circumfix in ab+**ge**+spiel+**t** (*played back*).

8 Conclusion and Outlook

We presented a new approach to modelling non-concatenative phenomena in morphology using simple range concatenating grammars and extended adaptor grammars to this formalism. Our experiments show that this richer model improves morphological segmentation and morpheme lexicon induction on different languages in the Semitic family.

Various avenues for future work present themselves. Firstly, the lightly-supervised, meta-grammar approach to adaptor grammars (Sirts and Goldwater, 2013) can be extended to this more powerful formalism to lessen the burden of defining the "right" grammar rules by hand, and possibly boost performance. Secondly, the discontiguous constituents learnt with our framework can be used as features in other downstream applications. Especially in low-resource languages, the ability to model non-concatenative phenomena (e.g. circumfixing, ablaut, etc.) can play an important role in reducing data sparsity for tasks like word alignment and language modelling. Finally, the PYSRCAG presents another way of learning SRCGs in general, which can thus be employed in other applications of SRCGs, including syntactic parsing and translation.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. Our PYSRCAG implementation leveraged the adaptor grammar code released by Mark Johnson, whom we thank, along with the individuals who contributed to the public data sources that enabled the empirical elements of this paper.

References

Aviad Albert, Brian MacWhinney, Bracha Nir, and Shuly Wintner. 2013. The Hebrew CHILDES corpus: transcription and morphological analysis. *Language Resources and Evaluation*, pages 1–33.

Mohamed Altantawy, Nizar Habash, Owen Rambow, and Ibrahim Saleh. 2010. Morphological Analysis and Generation of Arabic Nouns: A Morphemic Functional Approach. In *Proceedings of LREC*, pages 851–858.

Kenneth R Beesley and Lauri Karttunen. 2003. *Finite state morphology*, volume 18. CSLI publications Stanford.

Abderrahim Boudlal, Rachid Belahbib, Abdelhak Lakhouaja, Azzeddine Mazroui, Abdelouafi Meziane, and Mohamed Bebah. 2009. A Markovian approach for Arabic Root Extraction. *The International Arab Journal of Information Technology*, 8(1):91–98.

Pierre Boullier. 2000. A cubic time extension of context-free grammars. *Grammars*, 3(2-3):111–131.

Tim Buckwalter. 2002. Arabic Morphological Analyzer. Technical report, Linguistic Data Consortium, Philedelphia.

Alexander Clark. 2001. Learning Morphology with Pair Hidden Markov Models. In *Proceedings of the ACL Student Workshop*, pages 55–60.

Yael Cohen-Sygal and Shuly Wintner. 2006. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, 32(1):49–82.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):1–34.

Kareem Darwish. 2002. Building a shallow Arabic morphological analyzer in one day. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 47–54. Association for Computational Linguistics.

Ezra Daya, Dan Roth, and Shuly Wintner. 2008. Identifying Semitic Roots: Machine Learning with Linguistic Constraints. *Computational Linguistics*, 34(3):429–448.

Markus Dreyer and Jason Eisner. 2011. Discovering Morphological Paradigms from Plain Text Using a Dirichlet Process Mixture Model. In *Proceedings of EMNLP*, pages 616–627, Edinburgh, Scotland.

Kais Dukes and Nizar Habash. 2010. Morphological Annotation of Quranic Arabic. In *Proceedings of LREC*.

Greg Durrett and John DeNero. 2013. Supervised Learning of Complete Morphological Paradigms. In *Proceedings of NAACL-HLT*, pages 1185–1195, Atlanta, Georgia, June. Association for Computational Linguistics.

Khaled Elghamry. 2005. A Constraint-based Algorithm for the Identification of Arabic Roots. In *Proceedings of the Midwest Computational Linguistics Colloquium. Indiana University. Bloomington, IN*.

Raphael Finkel and Gregory Stump. 2002. Generating Hebrew verb morphology by default inheritance hierarchies. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. Association for Computational Linguistics.

Michelle A. Fullwood and Timothy J. O'Donnell. 2013. Learning non-concatenative morphology. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 21–27, Sofia, Bulgaria. Association for Computational Linguistics.

Michael Gasser. 2009. Semitic morphological analysis and generation using finite state transducers with feature structures. In *Proceedings of EACL*, pages 309–317. Association for Computational Linguistics.

Daniel Gildea. 2010. Optimal Parsing Strategies for Linear Context-Free Rewriting Systems. In *Proceedings of NAACL*, pages 769–776. Association for Computational Linguistics.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Interpolating Between Types and Tokens by Estimating Power-Law Generators. In *Advances in Neural Information Processing Systems, Volume 18*.

Harald Hammarström and Lars Borin. 2011. Unsupervised Learning of Morphology. *Computational Linguistics*, 37(2):309–350.

Yun Huang, Min Zhang, and Chew Lim Tan. 2011. Nonparametric Bayesian Machine Transliteration with Synchronous Adaptor Grammars. In *Proceedings of ACL (Short papers)*, pages 534–539.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of NAACL-HLT*, pages 317–325. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models. In *Advances in Neural Information Processing Systems*, volume 19, page 641. MIT.

Mark Johnson. 2008. Unsupervised word segmentation for Sesotho using Adaptor Grammars. In *Proceedings of ACL Special Interest Group on Computational Morphology and Phonology (SigMorPhon)*, pages 20–27. Association for Computational Linguistics.

Aravind K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D.R. Dowty, L. Karttunen, and A.M. Zwicky, editors, *Natural Language Parsing*, chapter 6, pages 206–250. Cambridge University Press.

Miriam Kaeshammer. 2013. Synchronous Linear Context-Free Rewriting Systems for Machine Translation. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 68–77, Atlanta, Georgia. Association for Computational Linguistics.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic Multiple Context-Free Grammar for RNA Pseudoknot Modeling. In *Proceedings of the International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 57–64.

George Anton Kiraz. 2000. Multitiered Nonlinear Morphology Using Multitape Finite Automata: A Case Study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105, March.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th international conference on Computational Linguistics*, pages 178–181. Association for Computational Linguistics.

Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood, and William Byrne. 2010. Overview and Results of Morpho Challenge 2009. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, volume 6241 of *Lecture Notes in Computer Science*, pages 578–597. Springer Berlin / Heidelberg.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of CoNLL*.

Wolfgang Maier. 2010. Direct Parsing of Discontinuous Constituents in German. In *Proceedings of the NAACL-HLT Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66. Association for Computational Linguistics.

Jim Pitman and Marc Yor. 1997. The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *The Annals of Probability*, 25(2):855–900.

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158.

Jean-François Prunet. 2006. External Evidence and the Semitic Root. *Morphology*, 16(1):41–67.

Paul Rodrigues and Damir Ćavar. 2007. Learning Arabic Morphology Using Statistical Constraint-Satisfaction Models. In Elabbas Benmamoun, editor, *Perspectives on Arabic Linguistics: Proceedings of the 19th Arabic Linguistics Symposium*, pages 63–75, Urbana, IL, USA. John Benjamins Publishing Company.

Nathan Schneider. 2010. Computational Cognitive Morphosemantics: Modeling Morphological Compositionality in Hebrew Verbs with Embodied Construction Grammar. In *Proceedings of the Annual Meeting of the Berkeley Linguistics Society*, Berkeley, CA.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-Supervised Morphological Segmentation using Adaptor Grammars. *Transactions of the ACL*.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, pages 104–111.

Grounding Strategic Conversation: Using negotiation dialogues to predict trades in a win-lose game

Anaïs Cadilhac
IRIT
Univ. Toulouse, France
cadilhac@irit.fr

Nicholas Asher
IRIT, CNRS
Toulouse, France
asher@irit.fr

Farah Benamara
IRIT
Univ. Toulouse, France
benamara@irit.fr

Alex Lascarides
School of Informatics
Univ. Edinburgh, UK
alex@inf.ed.ac.uk

Abstract

This paper describes a method that predicts which trades players execute during a win-lose game. Our method uses data collected from chat negotiations of the game *The Settlers of Catan* and exploits the conversation to construct dynamically a partial model of each player's preferences. This in turn yields equilibrium trading moves via principles from game theory. We compare our method against four baselines and show that tracking how preferences evolve through the dialogue and reasoning about equilibrium moves are both crucial to success.

1 Introduction

Rational agents act so as to maximise their *expected utilities*—an optimal trade off between what they *prefer* and what they *believe* they can achieve (Savage, 1954). Solving a game problem involves finding *equilibrium strategies*: an optimal action for each player that maximises his expected utility, assuming that the other players perform their specified action (Shoham and Leyton-Brown, 2009). Calculating equilibria thus requires knowledge of the other players' preferences but almost all *bargaining* games occur under the handicap of imperfect information about this (Osborne and Rubinstein, 1994). Players therefore try to extract their opponents' preferences from what they say, likewise revealing their own preferences in their own utterances. These elicited preferences guide an agent's decisions, like choosing to make such and such a bargain with such and such a person. Tracking preferences through dialogue is thus crucial for analyzing the agents' strategic reasoning in real game scenarios.

In this paper, we design a model that maps what people say in a win-lose game into a prediction of exactly which players, if any, trade with each other, and exactly what resources they exchange. We use both statistics and logic: we use a corpus of negotiation dialogues to learn classifiers that map each utterance to its speech act and to other acts pertinent to bargaining; and we develop a symbolic algorithm that, from the classifiers' output, dynamically constructs a model of each player's preferences as the conversation proceeds (for instance, the preference to receive a certain resource, or to accept a certain trade). This preference model uses CP-nets (Boutilier et al., 2004), a representation of preferences for which algorithms for computing equilibrium strategies exist. We adapt those algorithms to predict the trades executed in the game.

The algorithm for construcing CP-nets uses *only* the output of our classifiers, which in turn rely entirely on shallow features in the raw text and robust parsers. Together they provide an end to end model, from raw text to a prediction of which trade, if any, occurred. We evaluate the various components of this (pipeline) algorithm separately, as well as the end to end model.

Our study exploits a corpus of negotiation dialogues from an online version of the win lose game *The Settlers of Catan*. Sections 2 and 3 describe the corpus and its annotation. Section 4 introduces our method for constructing the agents' preferences from the dialogues. We use this in Section 5 to predict whether a trade is executed as a result of the

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 357–368,
Seattle, Washington, USA, 18-21 October 2013.

players' negotiations, and if so we predict who took part in the trade, and what they exchanged. Our method shows promising results, beating baselines that don't adequately track or reason about preferences. We compare our model to related work in Section 6 and point to future work in Section 7.

2 The game

The Settlers of Catan (`www.catan.com`) is a win-lose game that involves negotiations over restricted resources. Each player (three or more) acquires resources (of 5 types: ore, wood, wheat, clay, sheep), which they use in different combinations to build roads, settlements and cities, which in turn earns them points towards winning. The first player to 10 points wins. Players acquire resources in several ways, in particular through agreed trades with other players. Some methods (e.g., robbing) are hidden from view, so players lack complete information about their opponents' resources.

Our corpus contains conversations of humans playing an online version of *Settlers* (Afantenos et al., 2012). Players must converse in a chat interface to carry out trades. Each game contains several dozen self-contained bargaining dialogues. Our experiments use 10 *Settlers* games, consisting of more than 2000 individual dialogue turns (see Section 3).

Table 1 is a sample dialogue from the corpus. The sentences in the corpus have a relatively simple syntax, though many also exhibit long distance dependencies. However, these conversations are pragmatically complex. They exhibit complex anaphoric dependences (e.g., utterance ID 4 in Table 1). Other pragmatic inferences, which are dependent on reasoning about intentions, speech acts and discourse structure, are also ubiquitous. For example, the question *Have you got any ore?* implies an offer for the speaker to receive ore in exchange for something from someone unspecified, and its response *I've got wheat* not only implies a willingness to exchange wheat for something, but as a response to the question it also implies a refusal to give any ore.

More generally, a dialogue turn in our corpus can express an offer, a counteroffer, an acceptance or rejection of an offer, or a commentary on the above or on moves in the game. All except the last provide clues about preferences: e.g., which players a speaker wants to execute a trade with; or what resources to exchange. For instance, the utterance *Anybody have any sheep for wheat?* conveys several preferences. First, it conveys the speaker's preference to trade with someone unspecified. Other informative but underspecified preferences include: the speaker's preference to acquire some sheep over alternatives; and in a context where she receives sheep, a preference to give away some of her wheat over the alternatives. Crucially, it does not convey a preference to give away wheat in a context where she receives nothing or something other than sheep.

In line with a non-cooperative bargaining game, the preferences and offers that a speaker reveals are less specific than an executable trade requires, where the trading partners and the type of resources offered and received must all be defined. Such general dialogue moves are essentially information seeking—evidence that humans playing *Settlers* have imperfect information about their opponents' preferences. In fact, many offers to trade result in no trade being agreed to and executed. While observed negotiation failure would be puzzling in a bargaining game with perfect information (Osborne and Rubinstein, 1994), it occurs relatively frequently in *Settlers*.

3 Annotation

We have a multi-layered dialogue annotation scheme that includes: (1) a pre-annotation that segments the dialogue into turns which are further segmented into Elementary Discourse Units (EDUs) with the author of each turn automatically given; (2) a characterization of each EDU in terms of basic speech acts (assertion, question, request) as well as dialogue acts that are specific to bargaining (offers, counteroffers, etc.); and (3) associated information about the givable and/or receivable resources that EDUs express.

Two annotators received training on 77 dialogues, totaling 699 EDUs. They then both annotated the remaining dialogues independently (2741 EDUs and 511 dialogues in total). Kappas for inter-annotator agreement are given below.

3.1 Dialogue act annotation (Kappa=0.79)

Each turn logs what a player enters in the chat window and also aspects of the game state at the time:

ID	Dialogue Act	Text	Speaker	Addressee	Resource
1	Offer	i need clay, any1 have?	Rainbow	All	Receivable (clay, ?)
2	Refusal	Nope, sorry	inca	Rainbow	
3	Refusal	Not at the moment, unfortunately.	ariachiba	Rainbow	
4	Refusal	need mine sorry	Kittles	Rainbow	Not givable (Anaphoric, ?) Anaphora Link:(mine , clay)
5	Offer	no one has ore to giv?	Rainbow	All	Receivable (ore, ?)
6	Accept	oh yeah me	Kittles	Rainbow	
7	Counteroffer	ore for wheat again?	Kittles	Rainbow	Givable (ore, ?) Receivable (wheat, ?)
8	Accept	ya	Rainbow	Kittles	
9	Accept	ok	Kittles	Rainbow	

Table 1: Example of an annotated negotiation dialogue.

his resources, the state of the game board and a time stamp. The pre-annotation divides each turn into EDUs. The annotators then have to specify the *dialogue act* of each EDU: *Offer*, *Counteroffer*, *Accept* or *Refusal* (of an offer addressed to the emitter), and *Other*. *Other* labels units that either comment on strategic moves in the game or are not directly pertinent to bargaining. Annotators also specify the *addressee* of the EDU and its *surface type*: *Question*, *Request* or *Assertion*.

3.2 Resource type annotation (Kappa=0.80)

Annotators also specify for each EDU and its dialogue act an associated *feature structure*, which captures (partial) information that the EDU expresses about the type and quantity of resources that are of the following four attributes: *Givable*, *Not Givable*, *Receivable* or *Not Receivable*. These attributes can take Boolean combinations of resources as values via two operators AND and OR, that respectively stand for conjunction (the agent expresses two preferences and he prefers to achieve one of them if he cannot have both, such as *I need clay and wood*) and disjunction (free choice) of preferences (e.g., *I can give you clay or wood*). We allow attributes to have unknown values: the annotation tool inserts a ? in these cases. We also insist that the annotators resolve anaphoric dependencies when specifying values to attributes, as shown in EDU (4) in Table 1.

4 Dialogue act and resource prediction

Predicting the executed trades from the dialogues starts with three sub-tasks: automatically identifying each EDU's dialogue act; detecting the EDU's resources; and specifying the attributes of those resources (i.e., *Givable*, *Receivable*, etc.).

4.1 Identifying dialogue acts

As is well established, one EDU's dialogue act depends on previous dialogue acts (Stolcke et al., 2000). In our corpus, *Accept* or *Reject* frequently follow *Offer* and *Counteroffer*. Since labeling is sequential, we use Conditional Random Fields (CRFs) to learn dialogue acts. CRFs have been shown to yield better results in dialogue act classification on online chat than HMM-SVN and Naive Bayes (Kim et al., 2012).

We use three types of features: lexical, syntactic and semantic. And we exploit them as unigrams and bigrams: unigrams associate the value of the feature with the current output class (level 0); bigrams take account of the value of the feature associated with a combination of the current output class and previous output class (level -1). 6 features were used exclusively as unigrams: the EDU's position in the dialogue, its first and last words, its subject lemma, a boolean feature to indicate if the current speaker is the one that initiates the dialogue and the position of the speaker's first turn in the dialogue.

We have 15 unigram and bigram features (at levels 0 and -1), as well as templates that combine feature values for the two levels. These include 14 boolean features that indicate if the EDU contains: bargaining verbs (e.g. *trade, offer*), references to another player (e.g. *you*), resource tokens as encoded in a task dedicated lexicon (e.g. *wheat, clay*), quantifiers (e.g. *one, none*), anaphoric pronouns, occurrences of "for" prepositional phrases (e.g. *wheat for clay*), acceptance words (e.g. *OK*), negation words, emoticons, opinion words (from (Benamara et al., 2011)), words of politeness, exclamation marks, questions, and finally whether the EDU's speaker has talked previously in the dialogue.

The last feature gives the EDU speaker lemma. In addition, 3 unigram and bigram booleans indicate whether the current EDU contains the most frequent tokens, couple of tokens and syntactic patterns in our corpus. Finally, we use 2 composed bigram features that encode whether the EDU contains an acceptance or refusal word, given that the previous EDU is a question.

To assign sequential tags of dialogue acts within a negotiation dialogue, we use the CRF++ tool (`crfpp.googlecode.com`). Our data consists of 2741 EDUs in 511 dialogues. Each EDU is associated with a dialogue act resulting in 410 *Offer*, 197 *Counteroffer*, 179 *Accept*, 398 *Refusal* and 1557 *Other*. We use 10-fold cross-validation to evaluate our model, computing precision, recall and F-score for each class and global accuracy from the total number of true positives, false positives, false negatives and true negatives obtained by summing over all fold decisions. The results (in percent) are given in Table 2 (MaF is the average of F-scores of all the classes). Our model significantly outperforms the frequency-based baseline (MaF=14.5; Accuracy=56.8), with the best F-score achieved for *Other*. The least good results are for the two least frequent classes in our data. In addition to the frequency problem, the lower score for *Counteroffer* is mainly due to the model confusing it with *Offer*. Errors in the *Accept* class were often due to misspelling or to chat style conversation; e.g., *kk*, *yup*.

Dialogue act	Precision	Recall	F-score
Other	87.4	93.1	90.1
Offer	80.0	81.0	80.5
Counterof.	64.8	53.3	58.5
Accept	65.1	53.1	58.5
Refusal	81.7	73.9	77.6
Macro-averaged F-score (MaF)			**73.0**
Accuracy			**83.0**

Table 2: Results for dialogue act classification.

4.2 Finding resource text spans

Since the resource vocabulary in *The Settlers of Catan* is a closed set composed of words denoting specific resources (e.g., *clay*, *wood*) and their synonyms (*brick*), we use a simple rule to detect them: a noun phrase (NP) is a resource text span if and only if it contains a lemma from our resource lexicon. A closed set resource vocabulary is common to many different types of negotiation dialogues. We used the Stanford parser (Klein and Manning, 2003) to obtain the NPs: there are 4361 NPs, where (by the gold standard annotations) 21% are resources and 79% are not. We obtain an F-score of 96.9% and accuracy of 97.9%, clearly beating both the frequency and random baselines for this task.

4.3 Recognizing the type of resources

Recall that each resource within an EDU can be the value of four types of attributes: *Givable, Receivable, Not Givable* or *Not Receivable* (cf. Section 3.2). We predict these attributes using CRFs with the following features. 8 features are used as unigram at the current and the previous EDU level: the speaker, the EDU's subject, the dialogue act, and (if present) the lemma of a bargaining verb, and 4 boolean features indicate if the EDU contains an opinion word, a reference to another speaker, if the resource comes after a "for" and if it contains a refusal word. These features also serve as bigrams at the current EDU level. Additionally, we have a set of unigram and bigram boolean features that indicate if the current EDU contains the most frequent verbs in the corpus. And finally, we use a feature that encodes the combination subject/bargaining verb in the current EDU.

We used CRF++ to implement our classifier. Our corpus data consists of 1077 Resources, split into 510 *Receivable*, 432 *Givable*, 116 *Not Givable* and 19 *Not Receivable*. We use again 10-fold cross-validation to evaluate our model and compute the results by summing over all fold decisions. We present them (in percent) in Table 3. They beat the frequency-based baseline (MaF=16.1; Accuracy=47.4), although performance on the *Not Receivable* class is poor probably due to its low frequency in the data.

Ambiguities make this task challenging. For instance, *anyone wheat for clay?* can mean that the speaker wants to receive wheat and give clay or the opposite, and resolving which meaning is intended involves reasoning not only with the previous and/or the following EDU, but also sometimes EDUs with long distance attachments, which are not supported by our classifier and require a full discourse parser.

Res. type	Precision	Recall	F-score
Receivable	66.8	71.4	69.0
Givable	62.6	59.7	61.1
Not Giv.	88.1	89.7	88.9
Not Rec.	0	0	0
Macro-averaged F-score (MaF)			**54.8**
Accuracy			**67.4**

Table 3: Results for resource type classification.

5 Predicting Players' Strategic Actions

We aim to capture the evolution of commitments to certain preferences as the dialogue proceeds so as to predict the agents' bargaining behavior. In other words, we wish to predict which of the 61 possible trade actions is executed at the end of each dialogue. The possible trades vary over which partner the player whose turn it is trades with (3 options in a 4 player game), the resources exchanged (assuming each partner gives one type of resource and receives another type yields $5 \times 4 = 20$ possibilities), or there is no trade; i.e., $(3 \times 20) + 1 = 61$ possible actions in the hypothesis space (we predict the types of resources that are exchanged, but not their quantity).

We predict the executed action by identifying the equilibrium trade entailed by the model of the players' preferences, which in turn we construct dynamically from the output of the classifiers in Section 4. We use the attributes of resources in the EDUs (*Givable*, etc.) to identify the preference that a speaker conveys in the EDU, and we use the dialogue acts (*Offer*, *Accept*, etc.) to update a model of the preferences expressed so far in the dialogue with this new preference (see Section 5.2). Our model of preferences consists of a set of partial CP-nets, one for each player (see Section 5.1 for details). The resulting CP-nets are then used to infer the executed trading action (if any) automatically, via well-understood principles from game theory for identifying rational behavior (Bonzon, 2007).

5.1 CP-Nets

Following Cadilhac et al. (2011), we use CP-nets (Boutilier et al., 2004) to model preferences and their dependencies. CP-nets are compatible with the kind of partial information about preferences that utterances reveal, and inference with CP-nets is computationally efficient.

Just as Bayesian nets are a graphical model that exploits probabilistic conditional independence to provide a compact representation of a joint probability distribution (Pearl, 1988), CP-nets are a graphical model that exploits *conditional preferential independence* to provide a compact representation of the preference order over all outcomes. The CP-net structures the decision maker's preferences under a *ceteris paribus* assumption: outcomes are compared, other things being equal.

More formally, let V be a finite set of variables whose combination of values determine all outcomes O. Then a *preference relation* $\succeq$ over O is a reflexive and transitive binary relation with strict preference $\succ$ defined as: $o \succeq o'$ and $o' \not\succeq o$. Indifference, written $o \sim o'$, means $o \succeq o'$ and $o' \succeq o$. Definition 1 defines *conditional preference independence* and Definition 2 defines *CP-nets*: the graphical component $\mathcal{G}$ of a CP-net specifies for each variable $X \in V$ its *parent variables* $Pa(X)$ that affect the agent's preferences over the values of X, such that X is conditionally preferentially independent of $V \setminus (\{X\} \cup Pa(X))$ given $Pa(X)$.

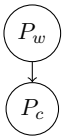
Definition 1 *Let V be a set of variables, each variable X_i with a domain $D(X_i)$. Let $\{X, Y, Z\}$ be a partition of V. X is* **conditionally preferentially independent** *of Y given Z if and only if $\forall z \in D(Z)$, $\forall x_1, x_2 \in D(X)$ and $\forall y_1, y_2 \in D(Y)$, $x_1 y_1 z \succeq x_2 y_1 z$ iff $x_1 y_2 z \succeq x_2 y_2 z$.*

Definition 2 *$\mathcal{N}_\mathcal{V} = \langle \mathcal{G}, \mathcal{T} \rangle$ is a* **CP-net** *on variables V, where $\mathcal{G}$ is a directed graph over V, and $\mathcal{T}$ is a set of Conditional Preference Tables (CPTs). That is, $\mathcal{T} = \{\text{CPT}(X_j)\text{: } X_j \in V\}$, where $\text{CPT}(X_j)$ specifies for each combination p of values of the parent variables $Pa(X_j)$ either $p : x_j \succ \overline{x_j}$, $p : \overline{x_j} \succ x_j$ or $p : x_j \sim \overline{x_j}$ where the $\overline{\ }$ symbol sets the variable to false.*

We discuss below how a CP-net predicts rational action, but first we describe how CP-nets are constructed from the dialogues. In the *Settlers* corpus, preferences involve a quadruplet *(o, a, <r,q>)* where: *o* is the preference owner, *a* is the addressee, *r* is the resource and *q* is its quantity. So each variable in the CP-nets we construct is such a quadruplet, and for each variable the possibles values are *Givable* (Giv), *Not Givable* ($\overline{Giv}$), *Receiv-*

able (Rcv) and *Not Receivable* ($\overline{Rcv}$).

For example, the utterance *Anyone want to give me a wheat for a clay?* expresses two preferences: one for receiving wheat, represented by the variable P_w = (A,All,<wheat,1>); and given this preference, another for giving clay, represented by P_c = (A,All,<clay,1>) (where A is the name of the speaker). The corresponding CP-Net is Figure 1.



CPT(P_w) = $Rcv \succ \overline{Rcv}$

CPT(P_c) = $Rcv\_P_w : Giv \succ \overline{Giv}$

Figure 1: An example CP-net

5.2 Modeling players' preferences

As stated above, we first automatically acquire a CP-net from each EDU by using the EDU's dialogue act and the attributes (*Givable*, etc.) of its resources. We then apply the rules presented in (Cadilhac et al., 2011) to dynamically construct a preference model of the dialogue overall: this uses an equivalence between their coherence relations and our dialogue acts. Our CP-nets reasoning model handles uncertain information and noise because it use as input only the outputs of the statistical models described in Section 4, and these prior models handle uncertain information and noise. The symbolic rules for constructing CP-nets have complete coverage over any possible combination of classes that are output by the statistical models, and so they are robust. We give our rules below where π_i stands for EDU ID i.

Offers. Because an *Offer* may specify or refine an existing preference or offer, we must model how the preferences expressed in an EDU that's an *Offer* updates the prior declared preferences. So, while our annotations treat *Offer* as a property of EDUs, we treat them here as *binary relations*: *Offer(π_1, π_2)*, where the second term, π_2, is the actual EDU whose dialogue act is *Offer* and π_1 is the set of EDUs occurring between π_2 and the last EDU uttered by the same speaker. *Offers* then have a similar effect on the CP-net as the coherence relation *Elaboration* presented in (Cadilhac et al., 2011). That is, to automatically update the CP-net constructed so far with a current EDU that's an *Offer*, the two step rule for *Offer(π_1, π_2)* is:

1. to update the speaker's CP-net according to the preferences expressed in π_1, and
2. if π_2 expresses preferences, to enrich the CP-net with these new preferences so that each variable in π_2 depends on each variable in π_1.

Counteroffers. They specify or modify the terms of a previous *Offer* or *Counteroffer*. Their purpose is to give new information to refine the negotiation. Like *Offers* they must also receive a contextually dependent interpretation. The rule is quite similar to that for *Offer*; however, *Counteroffer* can modify or *correct* elements in a previously introduced offer. So for *Counteroffer(π_1, π_2)*, the rule is :

1. to *partially* update the speaker's CP-net according to the preferences expressed in π_1 which do not have the same resource type (*Givable, Receivable*) than the ones in π_2.
2. same as step 2 *Offer* rule.

Accepts and Refusals. As they are answers to *Offers* and *Counteroffers*, they behave like question answer pairs (*QAPs*) presented in (Cadilhac et al., 2011). Because we are not doing full discourse parsing, we once again approximate its effects by making *Accepts* and *Refusals* respond to the set of EDUs between the current EDU and the speaker's last turn.

Accepts are positive responses to *Offers* or *Counteroffers* and are *de facto* similar to *QAP(π_1, π_2)* where π_2 is *Yes*. Thus, the rule is, as for *Offer*, to update and enrich the CP-net.

Refusals are instead negative responses and behave like *QAP(π_1, π_2)* where π_2 is *No*. For *Refusal(π_1, π_2)*, there is no update of the preferences expressed in π_1. Instead, we enrich the CP-net with the *Non Givable* and *Non Receivable* information obtained from the negation of the preferences expressed in the previous *Offer* or *Counteroffer*. We then enrich the CP-net based on any new preferences expressed in π_2. If there is a conflict between the value of a variable to be updated and the current value in the CP-net, we apply the *Correction* rule: all occurrences of the old value are replaced by the new value in π_2.

Other. This category pertains to content that does not directly relate to trading in the game, and so we choose to ignore resources expressed in the EDUs with this dialogue act.

At the end of the negotiation dialogue, to predict exactly what trade is executed (if any), the method

checks if there are complete and reciprocal preferences expressed in the CP-nets that respectively represent the declared preferences of two agents A and B. This is done in two steps. First, we use the logic of CP-nets to determine each agent's best outcome $bestO_A$ and $bestO_B$ from their respective CP-nets (we'll discuss how shortly). Secondly, we compare these best outcomes: if they correspond to the same trade, we predict that this trade was executed; if not, we predict no trade is executed. Specifically, $bestO_A$ (resp. $bestO_B$) corresponds to a preference for receiving a resource r_1 from an agent B (or from all the agents indifferently) and for giving a resource r_2 to this (or these) agent(s). We predict that A gives B r_2 and B gives A r_1 if and only if: $bestO_A = Rcv(A,\ B,\ r_1) \wedge Giv(A,\ B,\ r_2)$ and $bestO_B = Rcv(B,\ A,\ r_2) \wedge Giv(B,\ A,\ r_1)$.

The first step—computing each agent's best outcome from his CP-net—can be found in linear time using the *forward sweep algorithm* (Boutilier et al., 2004): sweep through the CP-net's graph from top to bottom, instantiating each variable with its preferred value, given the values that are (already) assigned to its parents. This algorithm is *sound* with respect to the semantics of CP-nets.

Example. We apply this method for constructing CP-nets and determining the executed trade to the negotiation dialogue presented in Table 1.

π_1 The EDU is an *Offer*, so Rainbow's CP-net is updated according to π_1's content.
CPT(R,All,<clay,?>) = $Rcv \succ \overline{Rcv}$

π_2 It's a *Refusal*, so we update inca's CP-net with the negation of the preferences expressed in Rainbow's offer.
CPT(I,R,<clay,?>) = $\overline{Giv} \succ Giv$

π_3 Idem for ariachiba.
CPT(A,R,<clay,?>) = $\overline{Giv} \succ Giv$

π_4 Idem for Kittles where the preferences expressed in this EDU are redundant with the negation of the preferences in Rainbow's offer.
CPT(K,R,<clay,?>) = $\overline{Giv} \succ Giv$

π_5 It's an *Offer*, so Rainbow's CP-net is first updated according to previous EDUs (π_2 to π_4 until his last speaking), then according to the content of π_5.
CPT(R,All,<clay,?>) = $Rcv \succ \overline{Rcv}$ ***(inactive)***
CPT(R,I,<clay,?>) = $\overline{Rcv} \succ Rcv$
CPT(R,A,<clay,?>) = $\overline{Rcv} \succ Rcv$
CPT(R,K,<clay,?>) = $\overline{Rcv} \succ Rcv$
CPT(R,All,<ore,?>) = $\overline{Rcv}$*(R,I,<clay,?>)* $\wedge$ $\overline{Rcv}$*(R,A,<clay,?>)* $\wedge$ $\overline{Rcv}$*(R,K,<clay,?>)*: $Rcv \succ \overline{Rcv}$
The introduction of the preference to receive ore conflicts with the prior one for receiving clay. So the method adds to the associated CPT the label "inactive" to indicate that this is older and should be ignored if the preference about ore is satisfied.

π_6 The EDU is an *Accept*, so Kittles's CP-net is updated according to previous EDUs (only π_5).[1]
CPT(K,R,<ore,?>) = $\overline{Giv}$*(K,R,<clay,?>)*: $Giv \succ \overline{Giv}$

π_7 The EDU is a *Counteroffer*. Since she is the last speaker, her CP-net gets updated only according to the content of the current EDU, to obtain:
CPT(K,R,<ore,?>) = $\overline{Giv}$*(K,R,<clay,?>)*: $Giv \succ \overline{Giv}$
CPT(K,R,<wheat,?>) = $\overline{Giv}$*(K,R,<clay,?>)* $\wedge$ *Giv(K,R, <ore,?>)* : $Rcv \succ \overline{Rcv}$

π_8 The EDU is an *Accept*, so Rainbow's CP-net is updated according to previous EDUs (π_6 and π_7):
CPT(R,K,<ore,?>) = $\overline{Rcv}$*(R,I,<clay,?>)* $\wedge$ $\overline{Rcv}$*(R,A,<clay,?>)* $\wedge$ $\overline{Rcv}$*(R,K,<clay,?>)* : $Rcv \succ \overline{Rcv}$
CPT(R,K,<wheat,?>) = $\overline{Rcv}$*(R,I,<clay,?>)* $\wedge$ $\overline{Rcv}$*(R,A,<clay,?>)* $\wedge$ $\overline{Rcv}$*(R,K,<clay,?>)* $\wedge$ *Rcv(R,K,<ore,?>)* : $Giv \succ \overline{Giv}$

π_9 It's an *Accept* with nothing new to update.

At the end of the dialogue, these agents' CP-nets (correctly) predict that Kittles gave ore to Rainbow in exchange for wheat.

5.3 Evaluation and results

We compare our model against four baselines. Since none of these baselines support reasoning about equilibrium moves, they all rely on the presence of an *Accept* act to predict there was a trade, and its absence to predict there wasn't. The baselines differ, however, in how they identify the trading partners and resources in an executed trade. The **first baseline** predicts a trade according to the first *Offer* and the last person to *Accept*, and if the *Offer* doesn't specify one of the resources then it is chosen randomly (similar random choices complete all partial predictions in all the models we consider here): e.g., for Table 1 this would predict that Kittles gave clay to Rainbow (which is incorrect) in exchange for

[1] Due to lack of space, in the following CP-nets, we do not copy the inactive CPTs and CPTs about *Not Givable* or *Not Receivable* resources.

something that's chosen randomly (which will probably be incorrect). The **second baseline** uses the last *Offer* and the last person to *Accept*: e.g., for Table 1 this predicts that Kittles gave ore to Rainbow (correct) for something random (probably incorrect). The **third baseline** uses the last *Offer* or *Counteroffer*, whichever is latest, and the last person to *Accept*: e.g., for Table 1 this correctly predicts that Kittles gave ore to Rainbow in exchange for wheat. And the **fourth baseline**, uses *default unification* between the prior *Offers* or *Counteroffers* and the current one to resolve any of the current offer's elided parts and to replace specific values in prior offers with conflicting specific values in the current offer (Ehlen and Johnston, 2013). One then takes the executed trade to be the result of this unification process at the point where the last *Accept* occurs. This makes the same predictions as the third baseline for Table 1, but outperforms it in the corpus example (1) by predicting the correct and *complete* trade (i.e., Rainbow gave Kittles sheep for wheat, rather than for something random):

(1) Rainbow: i need clay ore or wheat
Kittles: i got wheat
Rainbow: i cn giv sheep
Kittles: ok

We performed the evaluation on the data presented in Sections 3 and 4: 254 dialogues in total since we ignore dialogues that contain only *Other*s. 90 of these dialogues end with a trade being executed and 2 of them end with 2 trades. A random baseline would give 1.6% accuracy (given the 61 possible trading actions) and a frequency baseline (always choose no trade) gives 64.1% accuracy.

Table 4 presents the accuracy figures for all the models when calculated from the gold standard labels rather than the classifiers' predicted labels from Section 4, so that we can compare the models in isolation of the classifiers' errors. McNemar's test shows that our model significantly outperforms all the baselines ($p < 0.05$). A predicted trade counts as correct only if it specifies the right participants and the correct type of resources offered and received (we ignore their quantity). True Positives (TP) are thus examples where the model correctly predicts not only that a trade happened, but also the correct partners and resources; Wrong Positives (WP), on the other hand, constitute a correct prediction that there was a trade but errors on the partners and/or resources involved (so WPs undermine accuracy). True Negatives (TN) are examples where the model correctly predicts there was no trade (so TPs and TNs contribute to accuracy). False Positives (FP) and False Negatives (FN) are respectively incorrect predictions that there was a trade, or that there was no trade.

While Table 4 does not reflect this, the first three baselines tend to predict *incomplete* information about the trade even when what they do predict is correct: that is, they predict the correct addressee and the owner but resort to random choice for a resource that's missing from the *Offer* or *Counteroffer* that predicts which trade occurred. For the first baseline 34 examples are like this; for the second and third baselines it's 32. In contrast, this problem occurs only once with the fourth baseline, and all the trades predicted by our method are complete, making random choice unnecessary. Moreover, the first three baselines often make incorrect predictions about the addressee or resources exchanged because in contrast to our model and the fourth baseline, they don't track how potential trades *evolve* through a *sequence* of offers and counteroffers.

Even though the fourth baseline, which uses default unification to track the content of the current offer, is smart and gives good results, it has statistically significant lower accuracy than our model. One major problem with the fourth baseline is that, in contrast to our model, it does not track each player's *attitude* towards the current offer. Instead, like all our baselines, it relies on the presence of an *Accept* act to predict that there's a trade.[2] But several corpus examples are like (2), in which a trade is executed but there's no *Accept* act, thus yielding a False Negative (FN) for all four baselines:

(2) Joel: anyone have sheep or wheat
Cardlinger: neither :(
Joel: will give clay or ore
Euan: not just now
Jon: got a wheat for a clay
(*Joel gives clay to Jon and receives wheat*)

[2]We tried a baseline that doesn't rely on the presence of an *Accept* act, but rather predicts a trade whenever default unification yields a complete offer. It performed worse than the fourth baseline.

So overall, our analysis shows that using CP-nets significantly outperforms all baselines that don't model how preferences evolve in the dialogue, and error analysis yields evidence that our model outperforms the fourth baseline because our model supports *reasoning* about player preferences, rational behavior and equilibrium strategies.

1st baseline: first *Offer*/last *Accept*					
TP	FP	FN	TN	WP	Accuracy
24	14	30	150	38	**68.0**
2nd baseline: last *Offer*/last *Accept*					
TP	FP	FN	TN	WP	Accuracy
29	6	32	158	31	**73.0**
3rd baseline: last *(Counter)Offer*/last *Accept*					
TP	FP	FN	TN	WP	Accuracy
39	4	23	160	30	**77.7**
4th baseline: *default unification*					
TP	FP	FN	TN	WP	Accuracy
64	4	23	160	5	**87.5**
Our method					
TP	FP	FN	TN	WP	Accuracy
75	4	15	160	2	**91.8**

Table 4: Results for trade prediction. TP, FP, FN, TN and WP are the True and False Positives, False and True Negatives and Wrong Positives.

Table 5 presents the results for the end to end evaluation, where trade predictions are made from the classifiers' output from Section 4 rather than the gold standard labels. As expected, performance decreases due to the classifiers' errors, mainly on the type of resources (*Givable*, etc.). But our method still significantly outperforms all the baselines with an accuracy of 73.4% when the baselines obtain values between 60.9% and 68.4%.

4th baseline: *default unification*					
TP	FP	FN	TN	WP	Accuracy
23	12	37	152	32	**68.4**
Our method					
TP	FP	FN	TN	WP	Accuracy
34	10	43	154	15	**73.4**

Table 5: Results for the end to end trade prediction.

6 Related Work

6.1 Dialogue act modeling

Most work on dialogue act modeling focuses on spoken dialogue (Stolcke et al., 2000; Fernández et al., 2005; Keizer et al., 2002). But live chats introduce specific complications (Kim et al., 2012): ill-formed data, abbreviations and acronyms, emotional indicators and entanglement (especially for multi-party chat). Among related work in this emerging field, Joty et al. (2011) use unsupervised learning to model dialogue acts in Twitter, Ivanovic (2008) and Kim et al. (2010) analyze one-to-one online chat in a customer service domain, and Wu et al. (2002) and Kim et al. (2012) predict dialogue acts in a multi-party setting. We used a similar classifier to predict dialogue acts as the one reported in (Kim et al., 2012) and evaluation yields similar results.

This paper proposes an approach to dialogue act identification in online chat that aims to predict strategic actions like bargaining. Compared to (Sidner, 1994) and DAMSL (Core and Allen, 1997), our domain level annotation is much more detailed: we not only predict moves like *Accept* but also features like the *Givable* and *Receivable* resources. Our general speech act typology of EDUs lacks intentional descriptions of speech acts, however. This reflects a conscious choice to specify the semantics of each act purely by the public commitments made to offer or to receive goods.

6.2 Preference extraction

While preference extraction from non-linguistic actions is well studied (Chen and Pu, 2004; Fürnkranz and Hüllermeier, 2011), their extraction from spontaneous conversation has received little attention. To our knowledge, the only existing work is (Asher et al., 2010; Cadilhac et al., 2011; Cadilhac et al., 2012) which we build on. Cadilhac et al. (2011) compute CP-nets from *coherence relations*, found in the annotation of the Verbmobil corpus (Baldridge and Lascarides, 2005). Here we adapt their algorithm from coherence relations to *unary* dialogue acts. Further, while they assume that preferences are given, here we apply versions of the NLP techniques from Cadilhac et al (2012) to *estimate* the preferences of EDUs automatically. And we go further than any of these works by using the elicited pref-

erences to infer the domain-level actions that result from information exchanged in the conversation.

In this respect, our work relates to models for *grounding language*, where semantic parsing techniques are used to automatically map linguistic instructions to domain-level actions (Artzi and Zettlemoyer, 2013; Kim and Mooney, 2013). Our domain of application is more challenging, however: to our knowledge, this is the first attempt to map *non-cooperative* dialogues into predictions about domain-level actions. We can tackle these strategic scenarios because we exploit a logic of preferences as part of our model, yielding inferences about rational action even when agents' preferences conflict.

Compared to previous work, our task is new. Our aim is not to predict what dialogue act to perform next, but what non verbal action should be performed, mapping dialogue acts to non verbal actions. The difference between our work and other work on grounding is that we are grounding non-cooperative dialogue rather than instructions in a cooperative setting. There is no prior work of which we're aware that maps a non-cooperative dialogue into a prediction about which joint non-verbal action the agents will do as a result of what they've learned about their opponent through conversation. Furthermore, both the CP-net and the fourth baseline, whose accuracy is quite high (making it a hard baseline to beat), use the dialogue history as they incrementally build up the preference model.

6.3 Predicting strategic actions

Modeling player behavior in real-time strategy games is a growing research area in AI. These models can be used to identify common strategic states, discover new strategies as they emerge or predict an opponents future actions and so help players to optimize their choices. For example, Schadd et al. (2007) develop a hierarchical opponent model in the game *Spring*, Dereszynski et al. (2011) reason about strategic behavior in *StarCraft* using hidden Markov models and Amato and Shani (2010) use reinforcement learning to acquire a policy for switching among high-level strategies in *Civilization IV*.

In comparison, we propose a novel approach for predicting strategic action based on the *symbolically* formalized preferences that each agent commits to in spontaneous conversation. Our approach thus deals with imperfect information by exploiting the agents' declared preferences. By predicting what bargain (if any) will take place, we are able to verify the correctness of our preference descriptions. Our task is a subtask of learning a strategy over an entire game space, but our approach yields good predictive results on relatively little data—an advantage of exploiting CP-nets and the symbolic rules that guide their evolution from observable evidence.

7 Conclusion

We have proposed a linguistic approach to strategy prediction in spontaneous conversation, exploiting dialogue acts to build a partial model of the agents' declared preferences. Our method tracks how preferences evolve during the dialogue, which we use to infer their bargaining behavior, i.e. what resources, if any, are exchanged, and by whom.

We based our study on a corpus collected using an online version of *The Settlers of Catan*. Negotiations in this game mirror complex real life negotiations and provide a fruitful arena to study strategic conversation. Evaluation shows that our approach provides more accurate and complete information about trades than baselines that don't track how an offer evolves through the dialogue, and we also argued that game-theoretic reasoning about rational behavior has advantages over relying on the presence or absence of an *Accept* act to make predictions.

Our approach, however, does not exploit discourse structure, which is needed to properly handle long distance dependencies of offers on prior material. We will exploit this in future work to improve our results. We also plan to investigate other aspects of strategic reasoning on a larger dataset.

We have proposed a method that relies on a typology of dialogue acts that is domain sensitive. However, in other work we have shown how to adapt our algorithms to several domains (Cadilhac et al., 2012). In future work, we plan to link our preference extraction algorithms to an automatically acquired discourse structure for a given text. This will provide a domain independent means for extracting preferences from dialogue.

Acknowledgments

This work is supported by ERC grant 269427 STAC.

References

Stergos Afantenos, Nicholas Asher, Farah Benamara, Anaïs Cadilhac, Cédric Dégremont, Pascal Denis, Markus Guhe, Simon Keizer, Alex Lascarides, Oliver Lemon, Philippe Muller, Soumya Paul, Verena Rieser, and Laure Vieu. 2012. Developing a corpus of strategic conversation in the settlers of catan. In *Proceedings of the 1st Workshop on Games and NLP (GAMNLP-12)*.

Christopher Amato and Guy Shani. 2010. High-level reinforcement learning in strategy games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 75–82.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Nicholas Asher, Elise Bonzon, and Alex Lascarides. 2010. Extracting and modelling preferences from dialogue. In *IPMU*, pages 542–553.

Jason Baldridge and Alex Lascarides. 2005. Annotating discourse structures for robust semantic interpretation. In *Proceedings of the 6th IWCS*.

Farah Benamara, Baptiste Chardon, Yannick Mathieu, and Vladimir Popescu. 2011. Towards context-based subjectivity analysis. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1180–1188, Chiang Mai, Thailand.

Elise Bonzon. 2007. *Modélisation des interactions entre agents rationnels : les jeux booléens*. PhD thesis, Université Paul Sabatier, Toulouse.

Craig Boutilier, Craig Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. 2004. Cp-nets: A tool for representing and reasoning with conditional *ceteris paribus* preference statements. *Journal of Artificial Intelligence Research*, 21:135–191.

Anaïs Cadilhac, Nicholas Asher, Farah Benamara, and Alex Lascarides. 2011. Commitments to preferences in dialogue. In *Proceedings of SIGDIAL*, pages 204–215. ACL.

Anaïs Cadilhac, Nicholas Asher, Farah Benamara, Vladimir Popescu, and Mohamadou Seck. 2012. Preference extraction from negotiation dialogues. In *European Conference on Artificial Intelligence (ECAI)*, pages 211–216. IOS Press.

Li Chen and Pearl Pu. 2004. Survey of preference elicitation methods. Technical report.

Mark G. Core and James F. Allen. 1997. Coding dialogs with the DAMSL annotation scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*.

Ethan W. Dereszynski, Jesse Hostetler, Alan Fern, Thomas G. Dietterich, Thao-Trang Hoang, and Mark Udarbe. 2011. Learning probabilistic behavior models in real-time strategy games. In *AIIDE*.

Patrick Ehlen and Michael Johnston. 2013. A multimodal dialogue interface for mobile local search. In *IUI Companion*, pages 63–64.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2005. Using machine learning for non-sentential utterance classification. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 77–86.

Johannes Fürnkranz and Eyke Hüllermeier, editors. 2011. *Preference Learning*. Springer.

Edward Ivanovic. 2008. Automatic instant messaging dialogue using statistical models and dialogue acts. In *Masters thesis, The University of Melbourne*.

Shafiq R. Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised modeling of dialog acts in asynchronous conversations. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1807–1813.

Simon Keizer, Rieks op den Akker, and Anton Nijholt. 2002. Dialogue act recognition with bayesian networks for dutch dialogues. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 88–94. Association for Computational Linguistics.

Joohyun Kim and Raymond J. Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-2013)*, Sofia, Bulgaria.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialogue acts in 1-to-1 live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 862–871.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2012. Classifying dialogue acts in multi-party live chats. In *26th Pacific Asia Conference on Language,Information and Computation*, pages 463–472.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430.

Martin Osborne and Ariel Rubinstein. 1994. *A Course in Game Theory*. MIT Press.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffmann.

Leonard Savage. 1954. *The Foundations of Statistics*. John Wiley.

Frederik Schadd, Sander Bakkes, and Pieter Spronck. 2007. Opponent modeling in real-time strategy games. In *Games and Simulation GAMEON*, pages 61–68.

Yoav Shoham and Kevin Leyton-Brown. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic and Logical Foundations*. Cambridge University Press.

Candace Sidner. 1994. An artificial discourse language for collaborative negotiation. In *AAAI*, volume 1, pages 814–819. MIT Press, Cambridge.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol V. Ess-dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. In *Computational Linguistics*, pages 26:339–373.

Tianhao Wu, Faisal M. Khan, Todd A. Fisher, Lori A. Shuler, and William M. Pottenger. 2002. Posting act tagging using transformation-based learning. In *Foundations of Data Mining and knowledge Discovery*, pages 319–331. Springer.

Unsupervised Induction of Contingent Event Pairs from Film Scenes

Zhichao Hu, Elahe Rahimtoroghi, Larissa Munishkina, Reid Swanson and Marilyn A. Walker
Natural Language and Dialogue Systems Lab
Department of Computer Science, University of California, Santa Cruz
Santa Cruz, CA, 95064
{zhu, elahe, mlarissa, reid, maw}@soe.ucsc.edu

Abstract

Human engagement in narrative is partially driven by reasoning about discourse relations between narrative events, and the expectations about what is likely to happen next that results from such reasoning. Researchers in NLP have tackled modeling such expectations from a range of perspectives, including treating it as the inference of the CONTINGENT discourse relation, or as a type of common-sense causal reasoning. Our approach is to model likelihood between events by drawing on several of these lines of previous work. We implement and evaluate different unsupervised methods for learning event pairs that are likely to be CONTINGENT on one another. We refine event pairs that we learn from a corpus of film scene descriptions utilizing web search counts, and evaluate our results by collecting human judgments of contingency. Our results indicate that the use of web search counts increases the average accuracy of our best method to 85.64% over a baseline of 50%, as compared to an average accuracy of 75.15% without web search.

1 Introduction

Human engagement in narrative is partially driven by reasoning about discourse relations between narrative events, and the expectations about what is likely to happen next that results from such reasoning (Gerrig, 1993; Graesser et al., 1994; Lehnert, 1981; Goyal et al., 2010). Thus discourse relations are one of the primary means to structure narrative in genres as diverse as weblogs, search queries, stories, film scripts and news articles (Chambers and Jurafsky, 2009; Manshadi et al., 2008; Gordon and Swanson, 2009; Gordon et al., 2011; Beamer and Girju, 2009; Riaz and Girju, 2010; Do et al., 2011).

DOUGLAS QUAIL and his wife KRISTEN, are asleep in bed.
Gradually the room lights brighten. the clock chimes and begins speaking in a soft, feminine voice.
They don't budge. Shortly, the clock chimes again.
Quail's wife stirs. Maddeningly, the clock chimes a third time.
CLOCK (continuing)Tick, tock –.
Quail reaches out and shuts the clock off. Then he sits up in bed.
He swings his legs out from under the covers and sits on the edge of the bed. He puts on his glasses and sits, lost in thought.
He is a good-looking but conventional man in his early thirties. He seems rather in awe of his wife, who is attractive and rather off-hand towards him.
Kirsten pulls on her robe, lights a cigarette, sits fishing for her slippers.

Figure 1: Opening Scene from Total Recall

Recent work in NLP has tackled the inference of relations between events from a broad range of perspectives: (1) as inference of a discourse relations (e.g. the Penn Discourse Treebank (PDTB) CONTINGENT relation and its specializations); (2) as a type of common sense reasoning; (3) as part of text understanding to support question-answering; and (4) as way of learning script-like or plot-like knowledge structures. All these lines of work aim to model narrative understanding, i.e. to enable systems to infer which events are likely to **have happened** even though they have not been mentioned in the text (Schank et al., 1977), and which events are likely **to happen** in the future. Such knowledge has practical applications in commonsense reasoning, infor-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 369–379,
Seattle, Washington, USA, 18-21 October 2013.

mation retrieval, question answering, narrative understanding and inferring discourse relations.

We model this likelihood between events by drawing on the PTDB's general definition of the CONTINGENT relation, which encapsulates relations elsewhere called CAUSE, CONDITION and ENABLEMENT (Prasad et al., 2008a; Lin et al., 2010; Pitler et al., 2009; Louis et al., 2010). Our aim in this paper is to implement and evaluate a range of different unsupervised methods for learning event pairs that are likely to be CONTINGENT on one another.

We first utilize a corpus of scene descriptions from films because they are guaranteed to have an explicit narrative structure. Moreover, screenplay scene descriptions tend to be told in temporal order (Beamer and Girju, 2009; Gordon and Swanson, 2009), which makes them a good resource for learning about contingencies between events. In addition, scenes in film represent many typical sequences from real life, while providing a rich source of event clusters related to battles, love and mystery. We carry out separate experiments for the action movie genre and the romance movie genre. For example, in the scene from *Total Recall*, from the action movie genre (See Fig. 1), we might learn that the event of `sits up` is CONTINGENT on the event of `clock chimes`. The subset of the corpus we use comprises 123,869 total unique event pairs.

We produce initial scalar estimates of potential CONTINGENCY between events using four previously defined measures of distributional co-occurrence. We then refine these estimates through web searches that explicitly model the patterns of narrative event sequences that were previously observed to be likely within a particular genre. There are several advantages of this method: (1) events in the same genre tend to be more similar than events across genres, so less data is needed to estimate co-occurrence; (2) film scenes are typically narrated via simple tenses in the correct temporal order, which allows the ordering of events to contribute to the estimation of the CONTINGENCY relation; (3) The web counts focus on validating event pairs already deemed to be likely to be CONTINGENT in the smaller, more controlled, film scene corpus. To test our method, we conduct perceptual experiments with human subjects on Mechanical Turk by asking them to select which of two pairs of events are the most likely. For example, given the scene from *Total Recall* in Fig. 1, Mechanical Turkers are asked to select whether the sequential event pair `clock chimes, sits up` is more likely than `clock chimes` followed by a randomly selected event from the **action** film genre. Our experimental data and annotations are available at `http://nlds.soe.ucsc.edu/data/EventPairs`.

Sec. 2 describes our experimental method in detail. Sec. 3 describes how we set up our evaluation experiments and the results. We show that none of the methods from previous work perform better on our data than 75.15% average accuracy as measured by human perceptions of CONTINGENCY. But after web search refinement, we achieve an average accuracy of 85.64%. We delay a more detailed comparison to previous work to Sec. 4 where we summarize our results and compare previous work to our own.

2 Experimental Method

Our method uses a combination of estimating the likelihood of a CONTINGENT relation between events in a corpus of film scenes (Walker et al., 2012b), with estimates then revised through web search. Our experiments are based on two subsets of 862 film screen plays collected from the IMSDb website using its ontology of film genres (Walker et al., 2012b): a set of **action** movies of 115 screenplays totalling 748 MB, and a set of **romance** movies of 71 screenplays totalling 390 MB. Fig. 1 provided an example scene from the action movie genre from the IMSDb corpus.

We assume that the relation we are aiming to learn is the PDTB CONTINGENT relation, which is defined as a relation that exists when one of the situations described in the text spans that are identified as the two arguments of the relation, i.e. Arg1 and Arg2, causally influences the other (Prasad et al., 2008b). As Girju notes, it is notoriously difficult to define causality without making the definition circular, but we follow Beamer and Girju's work in assuming that if events A, B are causally related then B should occur less frequently when it is not preceded by A and that B $\rightarrow$A should be much less frequent than A $\rightarrow$ B. We assume that both the CAUSE and CONDITION subtypes of the CONTINGENCY relation will result in pairs of events that are likely to occur together and in a particular order. In particular we assume that the subtypes of the PDTB taxonomy of Contingency.Cause.Reason and Contingency.Cause.Result are the most likely to occur together as noted in previous work. Other related

work has made use of discourse connectives or discourse taggers (implicit discourse relations) to provide additional evidence of CONTINGENCY (Do et al., 2011; Gordon et al., 2011; Chiarcos, 2012; Pitler et al., 2009; Lin et al., 2010), but we do not because the results have been mixed. In particular these discourse taggers are trained on The Wall Street Journal (WSJ) and are unlikely to work well on our data.

We define an event as a verb lemma with its subject and object. Two events are considered equal if they have the same verb. We do not believe word ambiguities to be a primary concern, and previous work also defines events to be the same if they have the same surface verb, in some cases with a restriction that the dependency relations should also be the same (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Do et al., 2011; Riaz and Girju, 2010; Manshadi et al., 2008). Word sense ambiguities are also reduced in specific genres (Action and Romance) of film scenes.

Our method for estimating the likelihood of a CONTINGENT relations between events consists of four steps:

1. TEXT PROCESSING: We use Stanford CoreNLP to annotate the corpus document by document and store the annotated text in XML format (Sec. 2.1);

2. COMPUTE EVENT REPRESENTATIONS: We form intermediate artifacts such as events, protagonists and event pairs from the annotated documents. Each event has its arguments (subject and object). We calculate the frequency of the event across the relevant genre (Sec. 2.2);

3. CALCULATE CONTINGENCY MEASURES: We define 4 different measures of contingency and calculate each one separately using the results from Steps 1 and 2 above. We call each result a PREDICTED CONTINGENT EVENT PAIR (**PCEP**). All measures return scalar values that we use to rank the PCEPs (Sec. 2.3);

4. WEB SEARCH REFINEMENT: We select the top 100 event pairs calculated by each contingency measure, and construct a RANDOM EVENT PAIR (**REP**) for each PCEP that preserves the first element of the PCEP, and replaces the second element with another event selected randomly from within the same genre. We then define web search patterns for both PCEP and REPs and compare the counts (Sec. 2.4).

2.1 Text Processing

We first separate our screen plays into two sets of documents, one for the action genre and one for the romance genre. Because we are interested in the event descriptions that are part of the scene descriptions, we excise the dialog from each screen play. Then using the Stanford CoreNLP pipeline, we annotate the film scene files. Annotations include tokenization, lemmatization, named entity recognition, parsing and coreference resolution.

We extract the events by keeping all tokens whose POS tags begin with `VB`. We then use the dependency parse to find the subject and object of each verb (if any), considering only `nsubj, agent, dobj, iobj, nsubjpass`. We keep the original tokens of the subject and the object for further processing.

2.2 Compute Event Representations

Given the results of Step 1 we start by generalizing the subject and object stored with each event by substituting tokens with named entities if there are any named entities tagged. Otherwise we generalize the subjects and the objects using their lemmas. For example, `person UNLOCK door`, as illustrated in Table 1.

We then integrate all the subjects and objects across all film scene files, keeping a record of the frequency of each subject and object. For example, `[person (115), organization (14), door (3)] UNLOCK [door (127), person (5), bars (2)]`. The most frequent subject and object are selected as representative arguments for the event. We then count the frequency of each event across all the film scene files.

Within each film scene file, we count adjacent events as potential CONTINGENT event pairs. Two event pairs are defined as equal if they have the same verbs in the same order. We also count the frequency of each event pair.

2.3 Calculate Contingency Measures

We calculate four different measures of CONTINGENCY based on previous work using the results of Steps 1 and 2 (Sec. 2.1 and Sec. 2.2). These measures are pointwise mutual information, causal potential, bigram probability and protagonist-based

causal potential as described in detail below. We calculate each measure separately by genre for the action and romance genres of the film corpus.

Pointwise Mutual Information. The majority of related work uses pointwise mutual information (**PMI**) in some form or another (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Riaz and Girju, 2010; Do et al., 2011). Given a set of events (a verb and its collected set of subjects and objects), we calculate the PMI using the standard definition:

$$pmi(e_1, e_2) = \log \frac{P(e_1, e_2)}{P(e_1)P(e_2)} \quad (1)$$

in which e_1 and e_2 are two events. $P(e_1)$ is the probability that event e_1 occur in the corpus:

$$P(e_1) = \frac{count(e_1)}{\sum_x count(e_x)} \quad (2)$$

where $count(e_1)$ is the count of how many times event e_1 occurs in the corpus, and $\sum_x count(e_x)$ is the count of all the events in the corpus. The numerator is the probability that the two events occur together in the corpus:

$$P(e_1, e_2) = \frac{count(e_1, e_2)}{\sum_x \sum_y count(e_x, e_y)} \quad (3)$$

in which $count(e_1, e_2)$ is the number of times the two events e_1 and e_2 occur together in the corpus regardless of their order. Only adjacent events in each document are paired up. PMI is a symmetric measurement for the relationship between two events. The order of the events does not matter.

Causal Potential. Beamer and Girju proposed a measure called Causal Potential (**CP**) based on previous work in philosophy and logic, along with an annotation test for causality. An annotator deciding whether event A causes event B asks herself the following questions, where answering yes to both means the two events are causally related:

- Does event A occur before (or simultaneously) with event B?
- Keeping constant as many other states of affairs of the world in the given text context as possible, does modifying event A entail predictably modifying event B?

As Beamer & Girju note, this annotation test is objective, and it is simple to execute mentally. It only assumes that the average person knows a lot about how things work in the world and can reliably answer these questions. **CP** is then defined below, where the arrow notation means ordered bigrams, i.e. event e_1 occurs before event e_2:

$$\phi(e_1, e_2) = pmi(e_1, e_2) + \log \frac{P(e_1 \rightarrow e_2)}{P(e_2 \rightarrow e_1)} \quad (4)$$

$$\text{where } pmi(e_1, e_2) = \log \frac{P(e_1, e_2)}{P(e_1)P(e_2)}$$

The causal potential consists of two terms: the first is pair-wise mutual information (PMI) and the second is relative ordering of bigrams. PMI measures how often events occur as a pair; whereas relative ordering counts how often event order occurs in the bigram. If there is no ordering of events, the relative ordering is zero. We smooth unseen event pairs by setting their frequency equal to 1 to avoid zero probabilities. For **CP** as with PMI, we restrict these calculations to adjacent events. Column CP of Table 1 below provides sample values for the CP measure.

Probabilistic Language Models. Our third method models event sequences using statistical language models (Manshadi et al., 2008). A language model estimates the probability of a sequence of words using a sample corpus. To identify contingent event sequences, we apply a bigram model which estimates the probability of observing the sequence of two words w_1 and w_2 as follows:

$$P(w_1, w_2) \cong P(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)} \quad (5)$$

Here, the words are events. Each verb is a single event and each film scene is treated as a sequence of verbs. For example, consider the following sentence from *Total Recall*:

> *Quail and Kirsten sit at a small table, eating breakfast.*

This sentence is represented as the sequence of its two verbs: `sit, eat`. We estimate the probability of verb bigrams using Equation 5 and hypothesize that the verb sequences with higher probability are

Row #	Causal Potential Pair	CP	PCEP Search pattern	NumHits	Random Pair	REP Search pattern	NumHits
1	person KNOW person - person MEAN what	2.18	he knows * means	415M	person KNOW person - person PEDDLE papers	he knows * peddles	2
2	person COME - person REST head	2.12	he comes * rests	158M	person COME - person GLANCE window	he comes * glances	41
3	person SLAM person - person SHUT door	2.11	he slams * shuts	11	person SLAM person - person CHUCKLE	he slams * chuckles	0
4	person UNLOCK door - person ENTER room	2.11	he unlocks * enters	80	person UNLOCK door - person ACT shot	he unlocks * acts	0
5	person SLOW person - person STOP person	2.10	he slows * stops	697K	person SLOW person - eyes RIVET eyes	he slows * rivets	0
6	person LOOK window - person WONDER thing	2.06	he looks * wonders	342M	person LOOK window - person EDGE hardness	he looks * edges	98
7	person TAKE person - person LOOK window	2.01	he takes * looks	163M	person TAKE person - person CATCH person	he takes * catches	311M
8	person MANAGE smile - person GET person	2.01	he manages * gets	80M	person MANAGE smile - person APPROACH person	he manages * approaches	16
9	person DIVE escape - person SWIM way	2.00	he dives * swims	1.5M	person DIVE escape - gun JAM person	he dives * jams	6
10	person STAGGER person - person DROP person	2.00	he staggers * drops	33	person STAGGER person - plain WHEEL person	he staggers * wheels	1
11	person SHOOT person - person FALL feet	1.99	he shoots * falls	55.7M	person SHOOT person - person PREVENT person	he shoots * prevents	6
12	person SQUEEZE person - person SHUT door	1.87	he squeezes * shuts	5	person SQUEEZE person - person MARK person	he squeezes * marks	1
13	person SEE person - person GO	1.87	he sees * goes	184M	person SEE person - image QUIVER hips	he sees * quivers	2

Table 1: Sample web search patterns and values used in web search refinement algorithm from action genre

more likely to be contingent. We apply a threshold of 20 for $count(w_1, w_2)$ to avoid infrequent and uncommon bigrams.

Protagonist-based Models. We also used a method of generating event pairs based not only on the consecutive events in text but on their protagonist. This is based on the assumption that the agent, or protagonist, will tend to perform actions that further her own goals, and are thus causally related. We called this method protagonist-based because all events were partitioned into multiple sets where each set of events has one protagonist. This method is roughly based on previous work using chains of discourse entities to induce narrative schemas (Chambers and Jurafsky, 2009).

Events that share one protagonist were extracted from text according to co-referring mentions provided by the Stanford CoreNLP toolkit.[1] A manual examination of coreference results on a sample of movie scripts suggests that the accuracy is only around 60%: most of the time the same entity (in its nominal and pronominal forms) was not recognized and was assigned as a new entity.

We preserve the order of events based on their textual order assuming as above that film scripts tend to preserve temporal order. An ordered event pair is generated if both events share a protagonist. We further filter event pairs by eliminating those whose frequency is less than 5 to filter insignificant and rare event pairs. This also tends to catch errors generated by the Stanford parser.

CP was then calculated accordingly to Equation 4. To calculate the PMI part of **CP**, we combine the frequencies of event pairs in both orders.

2.4 Web Search Refinement

The final step of our method is WEB SEARCH REFINEMENT. Our hypothesis is that using the film corpus within a particular genre to do the initial estimates of contingency takes advantage of genre properties such as similar events and narration of scenes in chronological order. However the film corpus is necessarily small, and we can augment the evidence for a particular contingent relation by defining specific narrative sequence patterns and collecting web

[1] http://nlp.stanford.edu/software/corenlp.shtml

counts.

Recall that **PCEP** stands for predicted contingent event pair and that **REP** stands for random event pair. We first select the top 100 event pairs calculated by each CONTINGENCY measure, and construct a RANDOM EVENT PAIR (**REP**) for each PCEP that preserves the first element of the PCEP, and replaces the second element with another event selected randomly from within the same genre. We then define web search patterns for both PCEP and REPs and compare the counts. PCEPs should be frequent in web search and REPs should be infrequent.

Our web refinement procedure is:

- For each event pair, PCEPs and REPs, create a Google search pattern as illustrated by Table 1, and described in more detail below.
- Search for the exact match in Google general web search using incognito browsing and record the estimated count of results returned;
- Remove all the PCEP/REP pairs with PCEP Google search count less than 100: highly contingent events should be frequent in a general web search;
- Remove all PCEP/REP pairs with REP Google search count greater than 100: events that are not contingent on one another should not be frequent in a general web search.

The motivation for this step is to provide additional evidence **for** or **against** the contingency of a pair of events. Table 1 shows a selection of the top 100 PCEPs learned using the Causal potential (**CP**) Metric, the web search patterns that are automatically derived from the PCEPs (Column 4), the REPs that were constructed for each PCEP (Column 6), the web search patterns that were automatically derived from the REPs (Column 7). Column 5 shows the results of web search hits for the PCEP patterns and Column 8 shows the results of web search hits for the REP patterns. These hit counts were then used in refining our estimates of CONTINGENCY for the learned patterns as described above.

Note that the web search patterns do not aim to find every possible match of the targeted CONTINGENT relation that could possibly occur. Instead, they are **generalizations** of the instances of PCEPs that we found in the films corpus that are targeted at finding hits that are the most likely to occur in **narrative sequences**. Narrative sequences are most reliably signalled by use of the historical present tense, e.g. as instantiated in the search patterns *He knows* in Row 1 and *He comes* in Row 2 of Table 1 (Swanson and Gordon, 2012; Beamer and Girju, 2009; Labov and Waletzky, 1997). In addition, we use the "*" operator in Google Search to limit search to pairs of events reported in the historical present tense, that are "near" one another, and in a particular sequence. We don't care whether the events are in the same utterance or in sequential utterances, thus for the second verb (event) we do not include a subject pronoun *he*. These search patterns are not intended to match the original instances in the film corpus and in general they are unlikely to match those instances.

For example, consider the search patterns and results shown in Row 1 of Table 1. The PCEP is (`person KNOW person, person MEAN what`). The REP is (`person KNOW person, person PEDDLE papers`). Our prediction is that the REP should be much less likely in web search counts and the results validate that predication. A paired t-test over the 100 top PCEP pairs for the CP measure comparing the hit counts for the PCEP pairs vs. the REP pairs was highly significant ($p < .00001$). However, consider Row 7. Even though in general the PCEP pairs are more likely (as measured by the paired t-test comparing web search counts for PCEPs vs REPs), there are cases where the REP is highly likely as shown by the REP (`person take person, person CATCH person`) in Row 7. Alternatively there are cases where the web search counts provide evidence against one of the PCEPs. Consider Rows 3, 4, 10 and 12. In all of these cases the web counts NumHits for the PCEP number in the tens.

After the web search refinement, we retain the PCEP/REP pairs with initially high PCEP estimates, for which we found good evidence for contingency and for randomness, e.g. Row 1 and 2 in Table 1. We use 100 as a threshold because most of the time the estimate result count from Google is either a very large number (millions) or a very small number (tens), as illustrated by the NumHits columns in Table 1.

We experimented with different types of patterns with a development set of PCEPs before we settled on the search pattern template shown in Table 1. We

decided to use third person rather than first person patterns, because first person patterns are only one type of narrative (Swanson and Gordon, 2012). We also decided to utilize event patterns without typical objects, such as *head* in `person REST head` in Row 2 of Table 1. We do not have any evidence that this is the optimal search pattern template because we did not systematically try other types of search patterns.

3 Evaluation and Results

While other work uses a range of methods for evaluating accuracy, to our knowledge our work is the first to use human judgments from Mechanical Turk to evaluate the accuracy of the learned PCEPs. We first describe the evaluation setup in Sec. 3.1 and then report the results in Sec. 3.2

3.1 Mechanical Turk Contingent Pair Evaluations

We used three different types of HITs (Human Intelligence Tasks) on Mechanical Turk for our evaluation. Two of the HITS are in Fig. 2 and Fig. 3. The differences in the different types of HITS involve: (1) whether the arguments of events were given in the HIT, as in Fig. 2 and (2): whether the Turkers were told that the order of the events mattered, as in Fig. 3. We initially thought that providing the arguments to the events as shown in Fig. 2 would help Turkers to reason about which event was more likely. We tested this hypothesis only in the action genre for the Causal Potential Measure. For CP, Bigram and Protag the order of events always matters. For the PMI task, the order of the events doesn't matter because PMI is a symmetric measure. Fig. 2 illustrates the instructions that were given with the HIT when the event order doesn't matter. In all the other cases, the instructions that were given with the HIT are those in Fig. 3 where the Turkers are instructed to pay attention to the order of the events.

For all types of HITS, for all measures of CONTINGENCY, we set up the task as a choice over two alternatives, where for each predicted contingent pair (PCEP), we generate a random event pair (REP), with the first event the same and the second one randomly chosen from all the events in the same film genre. The REPs are constructed the same way as we construct REPs for web search refinement, as illustrated by Table 1. This is illustrated in both Fig. 2 and Fig. 3. For all types of HITS, we ask 15 Turkers from a pre-qualified group to select which pair (the PCEP or the REP) is more likely to occur together. Thus, the framing of these Mechanical Turk tasks only assumes that the average person knows how the world works; we do not ask them to explicitly reason about causality as other work does (Beamer and Girju, 2009; Gordon et al., 2011; Do et al., 2011).

For each measure of CONTINGENCY, we take 100 event pairs with highest PCEP scores, and put them in 5 HITs with twenty items per HIT. Previous work has shown that for many common NLP tasks, 7 Turkers' average score can match expert annotations (Snow et al., 2008). However, we use 15 Turkers because we had no gold-standard data and because we were not sure how difficult the task is. It is clearly subjective. To calculate the accuracy of each method, we computed the average correlation coefficient between each pair of raters and eliminate the 5 lowest scoring workers. We then used the perceptions of the 10 remaining workers to calculate accuracy as # of correct answers / total # of answers.

In general, deciding when a MTurk worker is unreliable when the data is subjective is a difficult problem. In the future we plan to test other solutions to measuring annotator reliability as proposed in related work (Callison-Burch, 2009; Snow et al., 2008; Karger et al., 2011; Dawid and Skene, 1979; Welinder et al., 2010; Liu et al., 2012).

3.2 Results

We report our results in terms of overall accuracy. Because the Mechanical Turk task is a choose-one question rather than a binary classification, Precision = Recall in our experimental results:

$$\text{True Positive} = \text{Number of Correct Answers}$$
$$\text{True Negative} = \text{Number of Correct Answers}$$
$$\text{False Positive} = \text{Number of Incorrect Answers}$$
$$\text{False Positive} = \text{Number of Incorrect Answers}$$
$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$
$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

The accuracies of all the methods are shown in Table 2. The results of using event arguments (`person KNOW person`) in the Mechanical Turk evaluation task (i.e. Fig. 2) is given in Rows 1 and 2 of Table 2. The accuracies for Rows 1 and 2 are

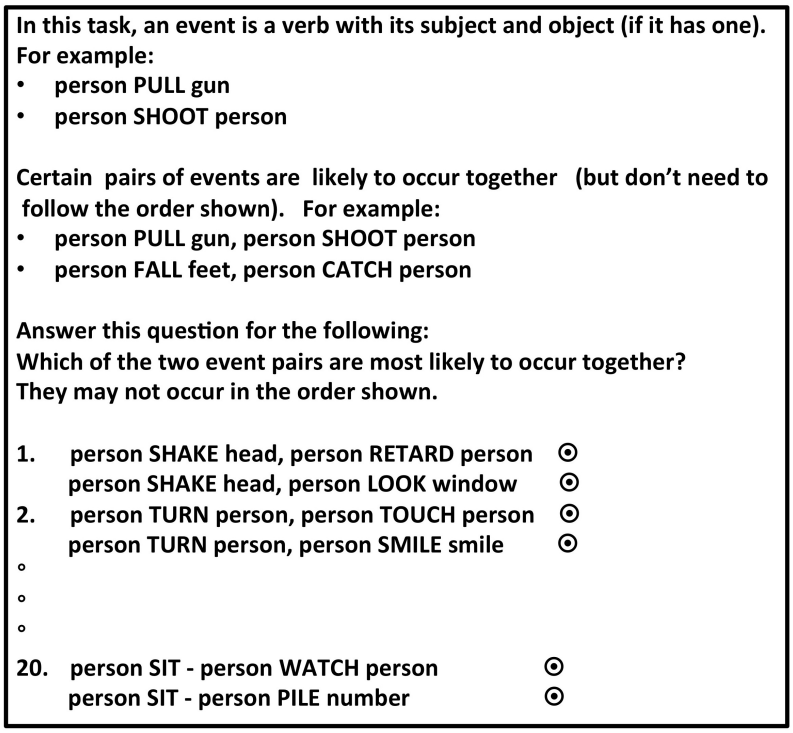


Figure 2: Mechanical Turk HIT with event arguments provided. This HIT also illustrates instructions where Turkers are told that the order of the events **does not** matter.

Row #	Contingency Estimation Method	Action Acc%	Romance Acc%	Average Acc%
1	CP with event arguments	69.30	NA	69.30
2	CP with event arguments + Web search	77.57	NA	77.57
3	CP no args	75.20	75.10	75.15
4	CP no args +Web Search	**87.67**	83.61	**85.64**
5	PMI no args	68.70	79.60	74.15
6	PMI no args +Web Search	72.11	**88.52**	80.32
7	Bigram no args	67.10	66.50	66.80
8	Bigram no args +Web Search	72.40	70.14	71.27
9	Protag CP no args	65.40	68.20	66.80
10	Protag CP no args +Web Search	76.59	64.10	70.35

Table 2: Evaluation results for the top 100 event pairs using all methods.

considerably lower than when the PCEPs are tested without arguments. Comparing Rows 1 and 2 with Rows 3 and 4 suggests that even if the arguments provide extra information that help to ground the type of event, in some cases these constraints on events may mislead the Turkers or make the evaluation task more difficult. There is an over 10% increase in CP + Web search accuracy for the task that omits the event arguments (i.e. Fig. 3 as can be seen by comparing Row 2 with Row 4. Thus omitting the arguments of events in evaluations actually appears to allow Turkers to make better judgments.

In addition, Table 2 shows clearly that for every single method, accuracy is improved by refining the initial estimates of contingency using the narrative-based web search patterns. Web search in-

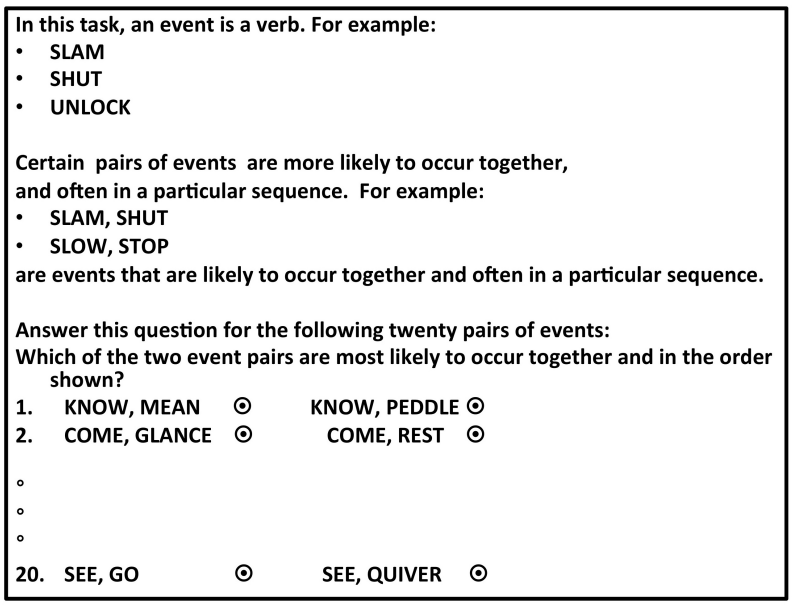


Figure 3: Mechanical Turk HIT for evaluation with no event arguments provided. This HIT also illustrates instructions where Turkers are told that the order of the events **does** matter.

creases the accuracy of almost all evaluation tasks, with increases ranging from 3.45% to 12.5% when averaged over both film genres (Column 5 Average Acc%). The best performing method for the Action genre is CP+Web Search at 87.67%, while the best performing method for the Romance genre is PMI+Web search at 88.52%. However PMI+Web Search does not beat CP+Web Search on average over both genres we tested, even though the Mechanical Turk HIT for CP specifies that the order of the events matters: a more stringent criterion. Also overall the CP+WebSearch method achieves a very high 85.64% average accuracy.

It is also interesting to note the variation across the different methods. For example, while it is well known that PMI typically requires very large corpora to make good estimates, the PMI method without web search refinement has an initially high accuracy of 79.60% for the romance genre, while only achieving 68.70% for action. Perhaps this difference arises because the romance genre is more highly causal, or because situations are more structured in romance, providing better estimates with a small corpus. However even in this case of romance with PMI, adding web search refinement provides an almost 10% increase in absolute accuracy to the highest accuracy of any combination, i.e. 88.52%. There is also an interesting case of Protag CP for the romance genre where web search refinement actually decreases accuracy by 4.1%. In future work we plan to examine more genres from the film corpus and also examine the role of corpus size in more detail.

4 Discussion and Future Work

We induced event pairs using several methods from previous work with similar aims but widely different problem formulations and evaluation methods. We used a verb-rich film scene corpus where events are normally narrated in temporal order. We used Mechanical Turk to evaluate the learned pairs of CONTINGENT events using human perceptions. In the first stage drawing on previous measures of distributional co-occurrence, we achieved an overall average accuracy of around 70%, over a 50% baseline. We then implemented a novel method of defining narrative sequence patterns using the Google Search API, and used web counts to further refine our estimates of the contingency of the learned event pairs. This increased the overall average accuracy to around 77%, which is 27% above the baseline. Our results indicate that the use of web search counts increases the average accuracy of our Causal Potential-based method to 85.64% as compared to an average accuracy of 75.15% without web search. To our knowledge this is the highest accuracy achieved in tasks of

this kind to date.

Previous work on recognition of the PDTB CONTINGENT relation has used both supervised and unsupervised learning, and evaluation typically measures precision and recall against a PDTB annotated corpus (Do et al., 2011; Pitler et al., 2009; Zhou et al., 2010; Chiarcos, 2012; Louis et al., 2010). We use an unsupervised approach and measure accuracy using human perceptions. Other work by Girju and her students defined a measure called causal potential and then used film screen plays to learn a knowledge base of causal pairs of events. They evaluate the pairs by asking two trained human annotators to label whether occurrences of those pairs in their corpus are causally related (Beamer and Girju, 2009; Riaz and Girju, 2010). We also make use of their causal potential measure. Work on commonsense causal reasoning aims to learn causal relations beween pairs of events using a range of methods applied to a large corpus of weblog narratives (Gordon et al., 2011; Gordon and Swanson, 2009; Manshadi et al., 2008). One form of evaluation aimed to predict the last event in a sequence (Manshadi et al., 2008), while more recent work uses the learned pairs to improve performance on the COPA SEMEVAL task (Gordon et al., 2011).

Related work on SCRIPT LEARNING induces *likely sequences of temporally ordered events* in news, rather than CONTINGENCY or CAUSALITY (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009). Chambers & Jurafsky also evaluate against a corpus of existing documents, by leaving one event out of a document (news story), and then testing the system's ability to predict the missing event. To our knowledge, our method is the first to augment distributional semantics measures from a corpus with web search data. We are also the first to evaluate the learned event pairs with a human perceptual evaluation with native speakers.

We hypothesize that there are several advantages to our method: (1) events in the same genre tend to be more similar than events across genres, so less data is needed to estimate co-occurrence; (2) film scenes are typically narrated via simple tenses in the correct temporal order, which allows the ordering of events to contribute to estimates of the CONTINGENCY relation; (3) The web counts focus on validating event pairs already deemed to be likely to be CONTINGENT in the smaller, more controlled, film scence corpus.

Our work capitalizes on event sequences narrated in temporal order as a cue to causality. We expect this approach to generalize to other domains where these properties hold, such as fables, personal stories and news articles. We do not expect this technique to generalize without further refinements to genres frequently told out of temporal order or when events are not mentioned consecutively in the text, for example in certain types of fiction.

In future work we want to explore in more detail the differences in performance of the different contingency measures. For example, previous work would suggest that the the higher the measure is, the more likely the two events are to be contingent on one another. To date, while we have only tested the top 100, we have not found that the bottom set of 20 are less accurate than the top set of 20. This could be due to corpus size, or the measures themselves, or noise from parser accuracy etc. As shown in Table 2, web search refinement is able to eliminate most noise in event pairs, but we would still aim to achieve a better understanding of the circumstances which lead particular methods to work better.

In future work we also want to explore ways of inducing larger event structures than event pairs, such as the causal chains, scripts, or narrative schemas of previous work.

Acknowledgments

We would like to thank Yan Li for setting up automatic search query. We also thank members of NLDS for their discussions and suggestions, especially Stephanie Lukin, Rob Abbort, and Grace Lin.

References

B. Beamer and R. Girju. 2009. Using a bigram event model to predict causal potential. In *Computational Linguistics and Intelligent Text Processing*, p. 430–441. Springer.

C. Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, p. 286–295. Association for Computational Linguistics.

N. Chambers and D. Jurafsky. 2008. Unsupervised learning of narrative event chains. *Proc. of ACL-08: HLT*, p. 789–797.

N. Chambers and D. Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In

Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, p. 602–610.

C. Chiarcos. 2012. Towards the unsupervised acquisition of discourse relations. In *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, p. 213–217. Association for Computational Linguistics.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Q. X. Do, Y. S. Chan, and D. Roth. 2011. Minimally supervised event causality identification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, p. 294–303. Association for Computational Linguistics.

R.J. Gerrig. 1993. *Experiencing narrative worlds: On the psychological activities of reading*. Yale Univ Pr.

A. Gordon and R. Swanson. 2009. Identifying personal stories in millions of weblog entries. In *Third International Conference on Weblogs and Social Media, Data Challenge Workshop*.

A. Gordon, Cosmin Bejan, and Kenji Sagae. 2011. Commonsense causal reasoning using millions of personal stories. In *Twenty-Fifth Conference on Artificial Intelligence (AAAI-11)*.

A. Goyal, E. Riloff, and H. Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *Proc. of the 2010 Conference on Empirical Methods in Natural Language Processing*, p. 77–86. Association for Computational Linguistics.

A. C. Graesser, M. Singer, and T. Trabasso. 1994. Constructing inferences during narrative text comprehension. *Psychological review*, 101(3):371.

D. R. Karger, S. Oh, and D. Shah. 2011. Iterative learning for reliable crowdsourcing systems. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, p. 1953–1961.

W. Labov and J. Waletzky. 1997. Narrative analysis: Oral versions of personal experience.

W. G. Lehnert. 1981. Plot units and narrative summarization. *Cognitive Science*, 5(4):293–331.

Z. Lin, M.-Y. Kan, and H. T Ng. 2010. A pdtb-styled end-to-end discourse parser. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*.

Q. Liu, J. Peng, and A. Ihler. 2012. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems 25*, p. 701–709.

A. Louis, A. Joshi, R. Prasad, and A. Nenkova. 2010. Using entity features to classify implicit relations. In *Proc. of the 11th Annual SIGdial Meeting on Discourse and Dialogue*, Tokyo, Japan.

M. Manshadi, R. Swanson, and A. S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proc. of the 21st FLAIRS Conference*.

E. Pitler, A. Louis, and A. Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proc. of the 47th Meeting of the Association for Computational Linguistics*.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008a. The penn discourse treebank 2.0. In *Proc. of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, p. 2961–2968.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008b. The Penn Discourse TreeBank 2.0. In *Proc. of 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

M. Riaz and R. Girju. 2010. Another look at causality: Discovering scenario-specific contingency relationships with no supervision. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, p. 361–368. IEEE.

R. Schank and R. Abelson. 1977. *Scripts Plans Goals*. Lea.

R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, p. 254–263. Association for Computational Linguistics.

R. Swanson and A. S. Gordon. 2012. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3):16.

M. A. Walker, G. Lin, and J. Sawyer. 2012b. An annotated corpus of film dialogue for learning and characterizing character style. In *Language Resources and Evaluation Conference, LREC2012*.

P. Welinder, S. Branson, S. Belongie, and P. Perona. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23*, p. 2424–2432.

Z.-M. Zhou, Y. Xu, Z.Y. Niu, M. Lan, J. Su, , and C. L. Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *In Coling 2010: Posters*, p. 1507–1514.

Latent Anaphora Resolution for Cross-Lingual Pronoun Prediction

Christian Hardmeier Jörg Tiedemann Joakim Nivre
Uppsala University
Department of Linguistics and Philology
Box 635, 751 26 Uppsala, Sweden
firstname.lastname@lingfil.uu.se

Abstract

This paper addresses the task of predicting the correct French translations of third-person subject pronouns in English discourse, a problem that is relevant as a prerequisite for machine translation and that requires anaphora resolution. We present an approach based on neural networks that models anaphoric links as latent variables and show that its performance is competitive with that of a system with separate anaphora resolution while not requiring any coreference-annotated training data. This demonstrates that the information contained in parallel bitexts can successfully be used to acquire knowledge about pronominal anaphora in an unsupervised way.

1 Motivation

When texts are translated from one language into another, the translation reconstructs the meaning or function of the source text with the means of the target language. Generally, this has the effect that the entities occurring in the translation and their mutual relations will display similar patterns as the entities in the source text. In particular, coreference patterns tend to be very similar in translations of a text, and this fact has been exploited with good results to project coreference annotations from one language into another by using word alignments (Postolache et al., 2006; Rahman and Ng, 2012).

On the other hand, what is true in general need not be true for all types of linguistic elements. For instance, a substantial percentage of the English third-person subject pronouns *he*, *she*, *it* and *they* does not get realised as pronouns in French translations (Hardmeier, 2012). Moreover, it has been recognised by various authors in the statistical machine translation (SMT) community (Le Nagard and Koehn, 2010; Hardmeier and Federico, 2010; Guillou, 2012) that pronoun translation is a difficult problem because, even when a pronoun does get translated as a pronoun, it may require choosing the correct word form based on agreement features that are not easily predictable from the source text.

The work presented in this paper investigates the problem of cross-lingual pronoun prediction for English-French. Given an English pronoun and its discourse context as well as a French translation of the same discourse and word alignments between the two languages, we attempt to predict the French word aligned to the English pronoun. As far as we know, this task has not been addressed in the literature before. In our opinion, it is interesting for several reasons. By studying pronoun prediction as a task in its own right, we hope to contribute towards a better understanding of pronoun translation with a long-term view to improving the performance of SMT systems. Moreover, we believe that this task can lead to interesting insights about anaphora resolution in a multi-lingual context. In particular, we show in this paper that the pronoun prediction task makes it possible to model the resolution of pronominal anaphora as a latent variable and opens up a way to solve a task relying on anaphora resolution without using any data annotated for anaphora. This is what we consider the main contribution of our present work.

We start by modelling cross-lingual pronoun prediction as an independent machine learning task after doing anaphora resolution in the source language (English) using the BART software (Broscheit et al., 2010). We show that it is difficult to achieve satisfactory performance with standard maximum-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 380–391,
Seattle, Washington, USA, 18-21 October 2013.

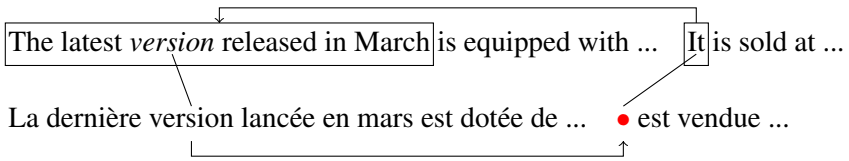


Figure 1: Task setup

entropy classifiers especially for low-frequency pronouns such as the French feminine plural pronoun *elles*. We propose a neural network classifier that achieves better precision and recall and manages to make reasonable predictions for all pronoun categories in many cases.

We then go on to extend our neural network architecture to include anaphoric links as latent variables. We demonstrate that our classifier, now with its own source language anaphora resolver, can be trained successfully with backpropagation. In this setup, we no longer use the machine learning component included in the external coreference resolution system (BART) to predict anaphoric links. Anaphora resolution is done by our neural network classifier and requires only some quantity of word-aligned parallel data for training, completely obviating the need for a coreference-annotated training set.

2 Task Setup

The overall setup of the classification task we address in this paper is shown in Figure 1. We are given an English discourse containing a pronoun along with its French translation and word alignments between the two languages, which in our case were computed automatically using a standard SMT pipeline with GIZA++ (Och and Ney, 2003). We focus on the four English third-person subject pronouns *he*, *she*, *it* and *they*. The output of the classifier is a multinomial distribution over six classes: the four French subject pronouns *il*, *elle*, *ils* and *elles*, corresponding to masculine and feminine singular and plural, respectively; the impersonal pronoun *ce/c'*, which occurs in some very frequent constructions such as *c'est* (*it is*); and a sixth class OTHER, which indicates that none of these pronouns was used. In general, a pronoun may be aligned to multiple words; in this case, a training example is counted as a positive example for a class if the target word occurs among the words aligned to the pronoun, irrespective of the presence of other

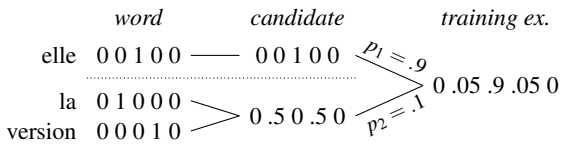


Figure 2: Antecedent feature aggregation

aligned tokens.

This task setup resembles the problem that an SMT system would have to solve to make informed choices when translating pronouns, an aspect of translation neglected by most existing SMT systems. An important difference between the SMT setup and our own classifiers is that we use context from human-made translations for prediction. This potentially makes the task both easier and more difficult; easier, because the context can be relied on to be correctly translated, and more difficult, because human translators frequently create less literal translations than an SMT system would. Integrating pronoun prediction into the translation process would require significant changes to the standard SMT decoding setup in order to take long-range dependencies in the target language into account, which is why we do not address this issue in our current work.

In all the experiments presented in this paper, we used features from two different sources:

– *Anaphora context features* describe the source language pronoun and its immediate context consisting of three words to its left and three words to its right. They are encoded as vectors whose dimensionality is equal to the source vocabulary size with a single non-zero component indicating the word referred to (one-hot vectors).
– *Antecedent features* describe an antecedent candidate. Antecedent candidates are represented by the target language words aligned to the syntactic head of the source language markable

	TED	News
ce	16.3 %	6.4 %
elle	7.1 %	10.1 %
elles	3.0 %	3.9 %
il	17.1 %	26.5 %
ils	15.6 %	15.1 %
OTHER	40.9 %	38.0 %

Table 1: Distribution of classes in the training data

noun phrase as identified by the Collins head finder (Collins, 1999).

The different handling of anaphora context features and antecedent features is due to the fact that we always consider a constant number of context words on the source side, whereas the number of word vectors to be considered depends on the number of antecedent candidates and on the number of target words aligned to each antecedent.

The encoding of the antecedent features is illustrated in Figure 2 for a training example with two antecedent candidates translated to *elle* and *la version*, respectively. The target words are represented as one-hot vectors with the dimensionality of the target language vocabulary. These vectors are then averaged to yield a single vector per antecedent candidate. Finally, the vectors of all candidates for a given training example are weighted by the probabilities assigned to them by the anaphora resolver (p_1 and p_2) and summed to yield a single vector per training example.

3 Data Sets and External Tools

We run experiments with two different test sets. The *TED* data set consists of around 2.6 million tokens of lecture subtitles released in the WIT^3 corpus (Cettolo et al., 2012). The WIT^3 training data yields 71,052 examples, which were randomly partitioned into a training set of 63,228 examples and a test set of 7,824 examples. The official WIT^3 development and test sets were not used in our experiments. The *news-commentary* data set is version 6 of the parallel news-commentary corpus released as a part of the WMT 2011 training data[1]. It contains around 2.8 million tokens of news text and yields 31,017 data points, which were randomly split into 27,900 training examples and 3,117 test instances. The distribution of the classes in the two training sets is shown in Table 1. One thing to note is the dominance of the OTHER class, which pools together such different phenomena as translations with other pronouns not in our list (e. g., *celui-ci*) and translations with full noun phrases instead of pronouns. Splitting this group into more meaningful subcategories is not straightforward and must be left to future work.

The feature setup of all our classifiers requires the detection of potential antecedents and the extraction of features pairing anaphoric pronouns with antecedent candidates. Some of our experiments also rely on an external anaphora resolution component. We use the open-source anaphora resolver BART to generate this information. BART (Broscheit et al., 2010) is an anaphora resolution toolkit consisting of a markable detection and feature extraction pipeline based on a variety of standard natural language processing (NLP) tools and a machine learning component to predict coreference links including both pronominal anaphora and noun-noun coreference. In our experiments, we always use BART's markable detection and feature extraction machinery. Markable detection is based on the identification of noun phrases in constituency parses generated with the Stanford parser (Klein and Manning, 2003). The set of features extracted by BART is an extension of the widely used mention-pair anaphora resolution feature set by Soon et al. (2001) (see below, Section 6).

In the experiments of the next two sections, we also use BART to predict anaphoric links for pronouns. The model used with BART is a maximum entropy ranker trained on the *ACE02-npaper* corpus (LDC2003T11). In order to obtain a probability distribution over antecedent candidates rather than one-best predictions or coreference sets, we modified the ranking component with which BART resolves pronouns to normalise and output the scores assigned by the ranker to all candidates instead of picking the highest-scoring candidate.

4 Baseline Classifiers

In order to create a simple, but reasonable baseline for our task, we trained a maximum entropy (ME)

[1] `http://www.statmt.org/wmt11/translation-task.html` (3 July 2013).

	TED (Accuracy: 0.685)			News commentary (Accuracy: 0.576)			
	P	R	F		P	R	F
ce	0.593	0.728	0.654	*ce*	0.508	0.294	0.373
elle	0.798	0.523	0.632	*elle*	0.530	0.312	0.393
elles	0.812	0.164	0.273	*elles*	0.538	0.062	0.111
il	0.764	0.550	0.639	*il*	0.600	0.666	0.631
ils	0.632	0.949	0.759	*ils*	0.593	0.769	0.670
OTHER	0.724	0.692	0.708	OTHER	0.564	0.609	0.586

Table 2: Maximum entropy classifier results

	TED (Accuracy: 0.700)			News commentary (Accuracy: 0.576)			
	P	R	F		P	R	F
ce	0.634	0.747	0.686	*ce*	0.477	0.344	0.400
elle	0.756	0.617	0.679	*elle*	0.498	0.401	0.444
elles	0.679	0.319	0.434	*elles*	0.565	0.116	0.193
il	0.719	0.591	0.649	*il*	0.655	0.626	0.640
ils	0.663	0.940	0.778	*ils*	0.570	0.834	0.677
OTHER	0.743	0.678	0.709	OTHER	0.567	0.573	0.570

Table 3: Neural network classifier with anaphoras resolved by BART

classifier with the MegaM software package[2] using the features described in the previous section and the anaphora links found by BART. Results are shown in Table 2. The baseline results show an overall higher accuracy for the TED data than for the news-commentary data. While the precision is above 50 % in all categories and considerably higher in some, recall varies widely.

The pronoun *elles* is particularly interesting. This is the feminine plural of the personal pronoun, and it usually corresponds to the English pronoun *they*, which is not marked for gender. In French, *elles* is a marked choice which is only used if the antecedent exclusively refers to females or feminine-gendered objects. The presence of a single item with masculine grammatical gender in the antecedent will trigger the use of the masculine plural pronoun *ils* instead. This distinction cannot be predicted from the English source pronoun or its context; making correct predictions requires knowledge about the antecedent of the pronoun. Moreover, *elles* is a low-frequency pronoun. There are only 1,909 occurrences of this pronoun in the TED training data, and 1,077 in the news-commentary training set. Because of these special properties of the feminine plural class, we argue that the performance of a classifier on *elles* is a good indicator of how well it can represent relevant knowledge about pronominal anaphora as opposed to overfitting to source contexts or acting on prior assumptions about class frequencies.

In accordance with the general linguistic preference for *ils*, the classifier tends to predict *ils* much more often than *elles* when encountering an English plural pronoun. This is reflected in the fact that *elles* has much lower recall than *ils*. Clearly, the classifier achieves a good part of its accuracy by making majority choices without exploiting deeper knowledge about the antecedents of pronouns.

An additional experiment with a subset of 27,900 training examples from the TED data confirms that the difference between TED and news commentaries is not just an effect of training data size, but that TED data is genuinely easier to predict than news commentaries. In the reduced data TED condition, the classifier achieves an accuracy of 0.673. Precision and recall of all classifiers are much closer to the

[2] `http://www.umiacs.umd.edu/~hal/megam/` (20 June 2013).

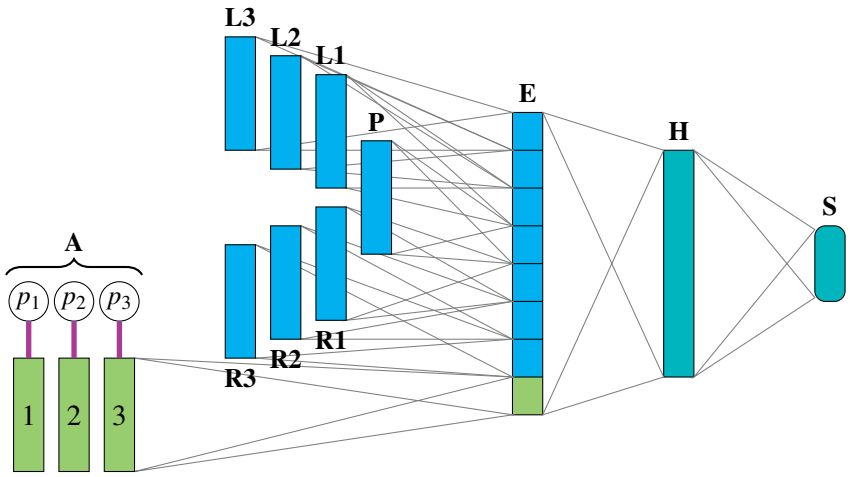


Figure 3: Neural network for pronoun prediction

large-data TED condition than to the news commentary experiments, except for *elles*, where we obtain an F-score of 0.072 (P 0.818, R 0.038), indicating that small training data size is a serious problem for this low-frequency class.

5 Neural Network Classifier

In the previous section, we saw that a simple multiclass maximum entropy classifier, while making correct predictions for much of the data set, has a significant bias towards making majority class decisions, relying more on prior assumptions about the frequency distribution of the classes than on antecedent features when handling examples of less frequent classes. In order to create a system that can be trained to rely more explicitly on antecedent information, we created a neural network classifier for our task. The introduction of a hidden layer should enable the classifier to learn abstract concepts such as gender and number that are useful across multiple output categories, so that the performance of sparsely represented classes can benefit from the training examples of the more frequent classes.

The overall structure of the network is shown in Figure 3. As inputs, the network takes the same features that were available to the baseline ME classifier, based on the source pronoun (**P**) with three words of context to its left (**L1** to **L3**) and three words to its right (**R1** to **R3**) as well as the words aligned to the syntactic head words of all possible antecedent candidates as found by BART (**A**). All words are encoded as one-hot vectors whose dimensionality is equal to the vocabulary size. If multiple words are aligned to the syntactic head of an antecedent candidate, their word vectors are averaged with uniform weights. The resulting vectors for each antecedent are then averaged with weights defined by the posterior distribution of the anaphora resolver in BART (p_1 to p_3).

The network has two hidden layers. The first layer (**E**) maps the input word vectors to a low-dimensional representation. In this layer, the embedding weights for all the source language vectors (the pronoun and its 6 context words) are tied, so if two words are the same, they are mapped to the same lower-dimensional embedding irrespective of their position relative to the pronoun. The embedding of the antecedent word vectors is independent, as these word vectors represent target language words. The entire embedding layer is then mapped to another hidden layer (**H**), which is in turn connected to a softmax output layer (**S**) with 6 outputs representing the classes *ce*, *elle*, *elles*, *il*, *ils* and OTHER. The non-linearity of both hidden layers is the logistic sigmoid function, $f(x) = 1/(1+e^{-x})$.

In all experiments reported in this paper, the dimensionality of the source and target language word embeddings is 20, resulting in a total embedding layer size of 160, and the size of the last hidden layer is equal to 50. These sizes are fairly small. In experiments with larger layer sizes, we were able to obtain similar, but no better results.

The neural network is trained with mini-batch stochastic gradient descent with backpropagated gradients using the RMSPROP algorithm with cross-entropy as the objective function.[3] In contrast to standard gradient descent, RMSPROP normalises the magnitude of the gradient components by dividing them by a root-mean-square moving average. We found this led to faster convergence. Other features of our training algorithm include the use of momentum to even out gradient oscillations, adaptive learning rates for each weight as well as adaptation of the global learning rate as a function of current training progress. The network is regularised with an ℓ_2 weight penalty. Good settings of the initial learning rate and the weight cost parameter (both around 0.001 in most experiments) were found by manual experimentation. Generally, we train our networks for 300 epochs, compute the validation error on a held-out set of some 10 % of the training data after each epoch and use the model that achieved the lowest validation error for testing.

Since the source context features are very informative and it is comparatively more difficult to learn from the antecedents, the network sometimes had a tendency to overfit to the source features and disregard antecedent information. We found that this problem can be solved effectively by presenting a part of the training without any source features, forcing the network to learn from the information contained in the antecedents. In all experiments in this paper, we zero out all source features (input layers **P**, **L1** to **L3** and **R1** to **R3**) with a probability of 50 % in each training example. At test time, no information is zeroed out.

Classification results with this network are shown in Table 3. We note that the accuracy has increased slightly for the TED test set and remains exactly the same for the news commentary corpus. However, a closer look on the results for individual classes reveals that the neural network makes better predictions for almost all classes. In terms of F-score, the only class that becomes slightly worse is the OTHER class for the news commentary corpus because of lower recall, indicating that the neural network classifier is less biased towards using the uninformative OTHER category. Recall for *elle* and *elles* increases considerably, but especially for *elles* it is still quite low. The increase in recall comes with some loss in precision, but the net effect on F-score is clearly positive.

[3] Our training procedure is greatly inspired by a series of online lectures held by Geoffrey Hinton in 2012 (`https://www.coursera.org/course/neuralnets`, 10 September 2013).

6 Latent Anaphora Resolution

Considering Figure 1 again, we note that the bilingual setting of our classification task adds some information not available to the monolingual anaphora resolver that can be helpful when determining the correct antecedent for a given pronoun. Knowing the gender of the translation of a pronoun limits the set of possible antecedents to those whose translation is morphologically compatible with the target language pronoun. We can exploit this fact to learn how to resolve anaphoric pronouns without requiring data with manually annotated anaphoric links.

To achieve this, we extend our neural network with a component to predict the probability of each antecedent candidate to be the correct antecedent (Figure 4). The extended network is identical to the previous version except for the upper left part dealing with anaphoric link features. The only difference between the two networks is the fact that anaphora resolution is now performed by a part of our neural network itself instead of being done by an external module and provided to the classifier as an input.

In this setup, we still use some parts of the BART toolkit to extract markables and compute features. However, we do not make use of the machine learning component in BART that makes the actual predictions. Since this is the only component trained on coreference-annotated data in a typical BART configuration, no coreference annotations are used anywhere in our system even though we continue to rely on the external anaphora resolver for preprocessing to avoid implementing our own markable and feature extractors and to make comparison easier.

For each candidate markable identified by BART's preprocessing pipeline, the anaphora resolution model receives as input a link feature vector (**T**) describing relevant aspects of the antecedent candidate-anaphora pair. This feature vector is generated by the feature extraction machinery in BART and includes a standard feature set for coreference resolution partially based on work by Soon et al. (2001). We use the following feature extractors in BART, each of

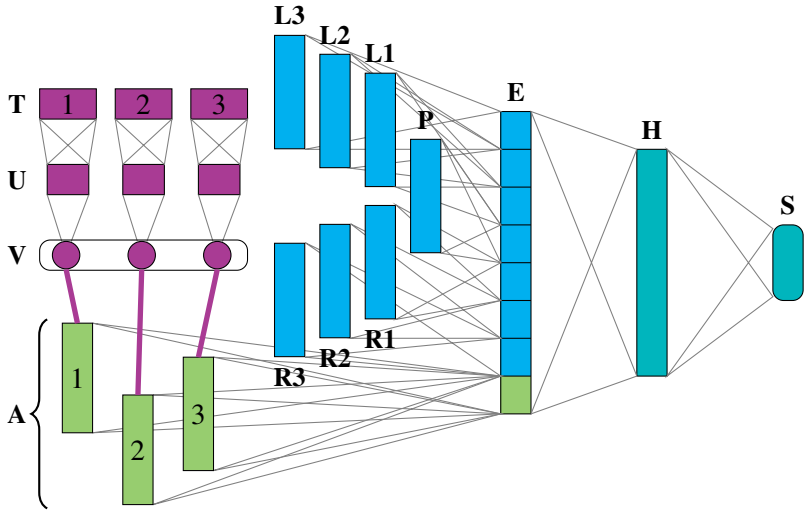


Figure 4: Neural network with latent anaphora resolution

which can generate multiple features:

- Anaphora mention type
- Gender match
- Number match
- String match
- Alias feature (Soon et al., 2001)
- Appositive position feature (Soon et al., 2001)
- Semantic class (Soon et al., 2001)
- Semantic class match
- Binary distance feature
- Antecedent is first mention in sentence

Our baseline set of features was borrowed wholesale from a working coreference system and includes some features that are not relevant to the task at hand, e. g., features indicating that the anaphora is a pronoun, is not a named entity, etc. After removing all features that assume constant values in the training set when resolving antecedents for the set of pronouns we consider, we are left with a basic set of 37 anaphoric link features that are fed as inputs to our network. These features are exactly the same as those available to the anaphora resolution classifier in the BART system used in the previous section.

Each training example for our network can have an arbitrary number of antecedent candidates, each of which is described by an antecedent word vector (**A**) and by an anaphoric link vector (**T**). The anaphoric link features are first mapped to a regular hidden layer with logistic sigmoid units (**U**). The activations of the hidden units are then mapped to a single value, which functions as an element in a softmax layer over all antecedent candidates (**V**). This softmax layer assigns a probability to each antecedent candidate, which we then use to compute a weighted average over the antecedent word vector, replacing the probabilities p_i in Figures 2 and 3.

At training time, the network's anaphora resolution component is trained in exactly the same way as the rest of the network. The error signal from the embedding layer is backpropagated both to the weight matrix defining the antecedent word embedding and to the anaphora resolution subnetwork. Note that the number of weights in the network is the same for all training examples even though the number of antecedent candidates varies because all weights related to antecedent word features and anaphoric link features are shared between all antecedent candidates.

One slightly uncommon feature of our neural network is that it contains an internal softmax layer to generate normalised probabilities over all possible antecedent candidates. Moreover, weights are shared between all antecedent candidates, so the inputs of our internal softmax layer share dependencies on the same weight variables. When computing derivatives with backpropagation, these shared dependencies must be taken into account. In particular, the outputs y_i of the antecedent resolution layer are the result of a softmax applied to functions of some shared variables q:

$$y_i = \frac{\exp f_i(q)}{\sum_k \exp f_k(q)} \quad (1)$$

The derivatives of any y_i with respect to q, which can be any of the weights in the anaphora resolution subnetwork, have dependencies on the derivatives of the other softmax inputs with respect to q:

$$\frac{\partial y_i}{\partial q} = y_i \left(\frac{\partial f_i(q)}{\partial q} - \sum_k y_k \frac{\partial f_k(q)}{\partial q} \right) \qquad (2)$$

This makes the implementation of backpropagation for this part of the network somewhat more complicated, but in the case of our networks, it has no major impact on training time.

Experimental results for this network are shown in Table 4. Compared with Table 3, we note that the overall accuracy is only very slightly lower for TED, and for the news commentaries it is actually better. When it comes to F-scores, the performance for *elles* improves by a small amount, while the effect on the other classes is a bit more mixed. Even where it gets worse, the differences are not dramatic considering that we eliminated a very knowledge-rich resource from the training process. This demonstrates that it is possible, in our classification task, to obtain good results without using any data manually annotated for anaphora and to rely entirely on unsupervised latent anaphora resolution.

7 Further Improvements

The results presented in the preceding section represent a clear improvement over the ME classifiers in Table 2, even though the overall accuracy increased only slightly. Not only does our neural network classifier achieve better results on the classification task at hand without requiring an anaphora resolution classifier trained on manually annotated data, but it performs clearly better for the feminine categories that reflect minority choices requiring knowledge about the antecedents. Nevertheless, the performance is still not entirely satisfactory.

By subjecting the output of our classifier on a development set to a manual error analysis, we found that a fairly large number of errors belong to two error types: On the one hand, the preprocessing pipeline used to identify antecedent candidates does not always include the correct antecedent in the set presented to the neural network. Whenever this occurs, it is obvious that the classifier cannot possibly find the correct antecedent. Out of 76 examples of the category *elles* that had been mistakenly predicted as *ils*, we found that 43 suffered from this problem. In other classes, the problem seems to be somewhat less common, but it still exists. On the other hand, in many cases (23 out of 76 for the category mentioned before) the anaphora resolution subnetwork does identify an antecedent manually recognised to belong to the right gender/number group, but still predicts an incorrect pronoun. This may indicate that the network has difficulties learning a correct gender/number representation for all words in the vocabulary.

7.1 Relaxing Markable Extraction

The pipeline we use to extract potential antecedent candidates is borrowed from the BART anaphora resolution toolkit. BART uses a syntactic parser to identify noun phrases as markables. When extracting antecedent candidates for coreference prediction, it starts by considering a window consisting of the sentence in which the anaphoric pronoun is located and the two immediately preceding sentences. Markables in this window are checked for morphological compatibility in terms of gender and number with the anaphoric pronoun, and only compatible markables are extracted as antecedent candidates. If no compatible markables are found in the initial window, the window is successively enlarged one sentence at a time until at least one suitable markable is found.

Our error analysis shows that this procedure misses some relevant markables both because the initial two-sentence extraction window is too small and because the morphological compatibility check incorrectly filters away some markables that should have been considered as candidates. By contrast, the extraction procedure does extract quite a number of first and second person noun phrases (*I*, *we*, *you* and their oblique forms) in the TED talks which are extremely unlikely to be the antecedent of a later occurrence of *he*, *she*, *it* or *they*. As a first step, we therefore adjust the extraction criteria to our task by increasing the initial extraction window to five sentences, excluding first and second person markables and removing the morphological compatibility requirement. The compatibility check is still used to control expansion of the extraction window, but it is no longer applied to filter the extracted markables. This increases the accuracy to 0.701 for TED and 0.602 for the news

	TED (Accuracy: 0.696)			News commentary (Accuracy: 0.597)		
	P	R	F	P	R	F
ce	0.618	0.722	0.666	0.419	0.368	0.392
elle	0.754	0.548	0.635	0.547	0.460	0.500
elles	0.737	0.340	0.465	0.539	0.135	0.215
il	0.718	0.629	0.670	0.623	0.719	0.667
ils	0.652	0.916	0.761	0.596	0.783	0.677
OTHER	0.741	0.682	0.711	0.614	0.544	0.577

Table 4: Neural network classifier with latent anaphora resolution

	TED (Accuracy: 0.713)			News commentary (Accuracy: 0.626)		
	P	R	F	P	R	F
ce	0.611	0.723	0.662	0.492	0.324	0.391
elle	0.749	0.596	0.664	0.526	0.439	0.478
elles	0.602	0.616	0.609	0.547	0.558	0.552
il	0.733	0.638	0.682	0.599	0.757	0.669
ils	0.710	0.884	0.788	0.671	0.878	0.761
OTHER	0.760	0.704	0.731	0.681	0.526	0.594

Table 5: Final classifier results

commentaries, while the performance for *elles* improves to F-scores of 0.531 (TED; P 0.690, R 0.432) and 0.304 (News commentaries; P 0.444, R 0.231), respectively. Note that these and all the following results are not directly comparable to the ME baseline results in Table 2, since they include modifications and improvements to the training data extraction procedure that might possibly lead to benefits in the ME setting as well.

7.2 Adding Lexicon Knowledge

In order to make it easier for the classifier to identify the gender and number properties of infrequent words, we extend the word vectors with features indicating possible morphological features for each word. In early experiments with ME classifiers, we found that our attempts to do proper gender and number tagging in French text did not improve classification performance noticeably, presumably because the annotation was too noisy. In more recent experiments, we just add features indicating all possible morphological interpretations of each word, rather than trying to disambiguate them. To do this, we look up the morphological annotations of the French words in the Lefff dictionary (Sagot et al., 2006) and introduce a set of new binary features to indicate whether a particular reading of a word occurs in that dictionary. These features are then added to the one-hot representation of the antecedent words. Doing so improves the classifier accuracy to 0.711 (TED) and 0.604 (News commentaries), while the F-scores for *elles* reach 0.589 (TED; P 0.649, R 0.539) and 0.500 (News commentaries; P 0.545, R 0.462), respectively.

7.3 More Anaphoric Link Features

Even though the modified antecedent candidate extraction with its larger context window and without the morphological filter results in better performance on both test sets, additional error analysis reveals that the classifiers has greater problems identifying the correct markable in this setting. One reason for this may be that the baseline anaphoric link feature set described above (Section 6) only includes two very rough binary distance features which indicate whether or not the anaphora and the antecedent candidate occur in the same or in immediately adjacent sentences. With the larger context window, this may be too unspecific. In our final experiment, we therefore enable some additional features which are available in BART, but disabled in the baseline system:

– Distance in number of markables
– Distance in number of sentences
– Sentence distance, log-transformed
– Distance in number of words
– Part of speech of head word

Most of these encode the distance between the anaphora and the antecedent candidate in more precise ways. Complete results for this final system are presented in Table 5.

Including these additional features leads to another slight increase in accuracy for both corpora, with similar or increased classifier F-scores for most classes except *elle* in the news commentary condition. In particular, we should like to point out the performance of our benchmark classifier for *elles*, which suffered from extremely low recall in the first classifiers and approaches the performance of the other classes, with nearly balanced precision and recall, in this final system. Since *elles* is a low-frequency class and cannot be reliably predicted using source context alone, we interpret this as evidence that our final neural network classifier has incorporated some relevant knowledge about pronominal anaphora that the baseline ME classifier and earlier versions of our network have no access to. This is particularly remarkable because no data manually annotated for coreference was used for training.

8 Related work

Even though it was recognised years ago that the information contained in parallel corpora may provide valuable information for the improvement of anaphora resolution systems, there have not been many attempts to cash in on this insight. Mitkov and Barbu (2003) exploit parallel data in English and French to improve pronominal anaphora resolution by combining anaphora resolvers for the individual languages with handwritten rules to resolve conflicts between the output of the language-specific resolvers. Veselovská et al. (2012) apply a similar strategy to English-Czech data to resolve different uses of the pronoun *it*. Other work has used word alignments to project coreference annotations from one language to another with a view to training anaphora resolvers in the target language (Postolache et al., 2006; de Souza and Orăsan, 2011). Rahman and Ng (2012) instead use machine translation to translate their test data into a language for which they have an anaphora resolver and then project the annotations back to the original language. Completely unsupervised monolingual anaphora resolution has been approached using, e. g., Markov logic (Poon and Domingos, 2008) and the Expectation-Maximisation algorithm (Cherry and Bergsma, 2005; Charniak and Elsner, 2009). To the best of our knowledge, the direct application of machine learning techniques to parallel data in a task related to anaphora resolution is novel in our work.

Neural networks and deep learning techniques have recently gained some popularity in natural language processing. They have been applied to tasks such as language modelling (Bengio et al., 2003; Schwenk, 2007), translation modelling in statistical machine translation (Le et al., 2012), but also part-of-speech tagging, chunking, named entity recognition and semantic role labelling (Collobert et al., 2011). In tasks related to anaphora resolution, standard feed-forward neural networks have been tested as a classifier in an anaphora resolution system (Stuckardt, 2007), but the network design presented in our work is novel.

9 Conclusion

In this paper, we have introduced cross-lingual pronoun prediction as an independent natural language processing task. Even though it is not an end-to-end task, pronoun prediction is interesting for several reasons. It is related to the problem of pronoun translation in SMT, a currently unsolved problem that has been addressed in a number of recent research publications (Le Nagard and Koehn, 2010; Hardmeier and Federico, 2010; Guillou, 2012) without reaching a major breakthrough. In this work, we have shown that pronoun prediction can be effectively modelled in a neural network architecture with relatively simple features. More importantly, we have demonstrated that the task can be exploited to train a classifier with a latent representation of anaphoric links. With parallel text as its only supervision this classifier achieves a level of performance that is similar to, if not better than, that of a classifier using a regular anaphora resolution system trained with manually annotated data.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Samuel Broscheit, Massimo Poesio, Simone Paolo Ponzetto, Kepa Joseba Rodriguez, Lorenza Romano, Olga Uryupina, Yannick Versley, and Roberto Zanoli. 2010. BART: A multilingual anaphora resolution system. In *Proceedings of the 5th International Workshop on Semantic Evaluations (SemEval-2010),* Uppsala, Sweden, 15–16 July 2010.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Eugene Charniak and Micha Elsner. 2009. EM works for pronoun anaphora resolution. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 148–156, Athens, Greece.

Colin Cherry and Shane Bergsma. 2005. An Expectation Maximization approach to pronoun resolution. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 88–95, Ann Arbor, Michigan.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2461–2505.

José de Souza and Constantin Orăsan. 2011. Can projected chains in parallel corpora help coreference resolution? In Iris Hendrickx, Sobha Lalitha Devi, António Branco, and Ruslan Mitkov, editors, *Anaphora Processing and Applications*, volume 7099 of *Lecture Notes in Computer Science*, pages 59–69. Springer, Berlin.

Liane Guillou. 2012. Improving pronoun translation for statistical machine translation. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–10, Avignon, France.

Christian Hardmeier and Marcello Federico. 2010. Modelling pronominal anaphora in statistical machine translation. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 283–289, Paris, France.

Christian Hardmeier. 2012. Discourse in statistical machine translation: A survey and a case study. *Discours*, 11.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montréal, Canada.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 252–261, Uppsala, Sweden.

Ruslan Mitkov and Catalina Barbu. 2003. Using bilingual corpora to improve pronoun resolution. *Languages in Contrast*, 4(2):201–211.

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

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 650–659, Honolulu, Hawaii.

Oana Postolache, Dan Cristea, and Constantin Orăsan. 2006. Transferring coreference chains through word alignment. In *Proceedings of the 5th Conference on International Language Resources and Evaluation (LREC-2006)*, pages 889–892, Genoa.

Altaf Rahman and Vincent Ng. 2012. Translation-based projection for multilingual coreference resolution. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, Montréal, Canada.

Benoît Sagot, Lionel Clément, Éric Villemonte de La Clergerie, and Pierre Boullier. 2006. The Lefff 2 syntactic lexicon for French: architecture, acquisition, use. In *Proceedings of the 5th Conference on International Language Resources and Evaluation (LREC-2006)*, pages 1348–1351, Genoa.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Roland Stuckardt. 2007. Applying backpropagation networks to anaphor resolution. In António Branco, editor, *Anaphora: Analysis, Algorithms and Applications. 6th*

Discourse Anaphora and Anaphor Resolution Colloquium, DAARC 2007, number 4410 in Lecture Notes in Artificial Intelligence, pages 107–124, Berlin.

Kateřina Veselovská, Ngụy Giang Linh, and Michal Novák. 2012. Using Czech-English parallel corpora in automatic identification of *it*. In *Proceedings of the 5th Workshop on Building and Using Comparable Corpora*, pages 112–120, Istanbul, Turkey.

Towards Situated Dialogue: Revisiting Referring Expression Generation

Rui Fang, Changsong Liu, Lanbo She, Joyce Y. Chai
Department of Computer Science and Engineering
Michigan State University, East Lansing, MI, 48824, USA
{fangrui,cliu,shelanbo,jchai}@cse.msu.edu

Abstract

In situated dialogue, humans and agents have mismatched capabilities of perceiving the shared environment. Their representations of the shared world are misaligned. Thus referring expression generation (REG) will need to take this discrepancy into consideration. To address this issue, we developed a hypergraph-based approach to account for group-based spatial relations and uncertainties in perceiving the environment. Our empirical results have shown that this approach outperforms a previous graph-based approach with an absolute gain of 9%. However, while these graph-based approaches perform effectively when the agent has perfect knowledge or perception of the environment (e.g., 84%), they perform rather poorly when the agent has imperfect perception of the environment (e.g., 45%). This big performance gap calls for new solutions to REG that can mediate a shared perceptual basis in situated dialogue.

1 Introduction

Situated human robot dialogue has received increasing attention in recent years. In situated dialogue, robots/artificial agents and their human partners are co-present in a shared physical world. Robots need to automatically perceive and make inference of the shared environment. Due to its limited perceptual and reasoning capabilities, the robot's representation of the shared world is often incomplete, error-prone, and significantly mismatched from that of its human partner's. Although physically co-present, a joint perceptual basis between the human and the robot cannot be established (Clark and Brennan, 1991). Thus, referential communication between the human and the robot becomes difficult.

How this mismatched perceptual basis affects referential communication in situated dialogue was investigated in our previous work (Liu et al., 2012). In that work, the main focus is on reference resolution: given referential descriptions from human partners, how to identify referents in the environment even though the robot only has imperfect perception of the environment. Since robots need to collaborate with human partners to establish a joint perceptual basis, referring expression generation (REG) becomes an equally important problem in situated dialogue. Robots have much lower perceptual capabilities of the environment than humans. How can a robot effectively generate referential descriptions about the environment so that its human partner can understand which objects are being referred to?

There has been a tremendous amount of work on referring expression generation in the last two decades (Dale, 1995; Krahmer and Deemter, 2012). However, most existing REG algorithms were developed and evaluated under the assumption that agents and humans have access to the same kind of domain information. For example, many experimental setups (Gatt et al., 2007; Viethen and Dale, 2008; Golland et al., 2010; Striegnitz et al., 2012) were developed based on a visual world for which the internal representation is assumed to be known and can be represented symbolically. However, this assumption no longer holds in situated dialogue with robots. There are two important distinctions in situated dialogue. *First, the perfect knowledge of the environment is not available to the agent ahead of time.* The agent needs to automatically make inferences to connect recognized lower-level visual features with

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 392–402,
Seattle, Washington, USA, 18-21 October 2013.

symbolic labels or descriptors. Both recognition and inference are error-prone and full of uncertainties. *Second, in situated dialogue the agent and the human have mismatched representations of the environment.* The agent needs to take this difference into consideration to identify the most reliable features for REG. Given these two distinctions, it is not clear whether state-of-the-art REG approaches are applicable under mismatched perceptual basis in situated dialogue.

To address this issue, this paper revisits the problem of REG in the context of mismatched perceptual basis. We extended a well known graph-based approach (Krahmer et al., 2003) that has shown to be effective in previous work (Gatt and Belz, 2008; Gatt et al., 2009). We incorporated uncertainties in perception into cost functions. We further extended regular graph representation into hypergraph representation to account for group-based spatial relations that are important for visual descriptions (Dhande, 2003; Tenbrink and Moratz, 2003; Funakoshi et al., 2006; Liu et al., 2012). Our empirical results demonstrate that both enhancements lead to about a 9% absolute performance gain compared to the original approach. However, while our approache performs effectively when the agent has perfect knowledge or perception of the environment (e.g., 84%), it performs poorly under the mismatched perceptual basis (e.g., 45%). This performance gap calls for new solutions for REG that are capable of mediating mismatched perceptual basis.

In the following sections, we first describe our hypergraph-based representations and illustrate how uncertainties from automated perception can be incorporated. We then describe an empirical study using Amazon Mechanical Turks for evaluating generated referring expressions. Finally we present evaluation results and discuss potential future directions.

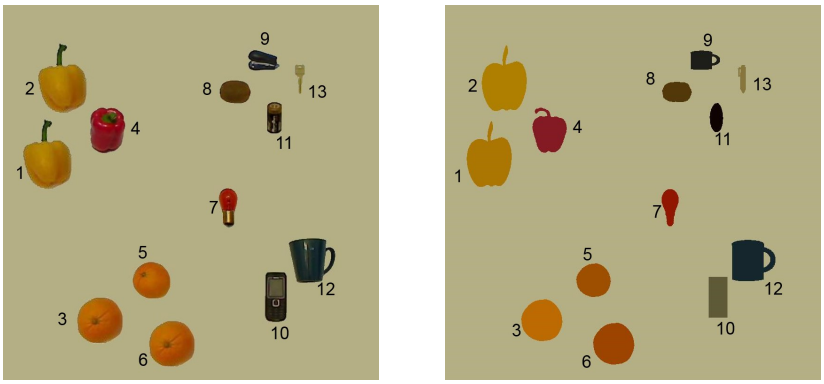
2 Related Work

Since the Full Brevity algorithm (Dale, 1989), many approaches have been developed and evaluated for REG (Dale, 1995; Krahmer and Deemter, 2012), such as the incremental algorithm (Dale, 1995), the locative algorithm (Kelleher and Kruijff, 2006), and graph-based approaches (Krahmer et al., 2003; Croitoru and Van Deemter, 2007). Most of these approaches assume the agent has access to a complete symbolic representation of the domain. While these approaches work well for many applications involving user interfaces, the question is whether they can be extended to the situation where the agent has incomplete or incorrect knowledge and needs to make inference about the domain or the world.

Recently, there has been increasing interest in REG for visual objects (Roy, 2002; Golland et al., 2010; Mitchell et al., 2013). Some work (Golland et al., 2010) uses visual scenes that are generated by computer graphics and thus the internal representation of the scene is known. Some other work focuses on the connection between lower-level visual features and symbolic descriptors for REG (Roy, 2002; Mitchell et al., 2013). However, most work assumes no vision recognition errors. It is well established that automated recognition of visual scenes is extremely challenging. This process is error-prone and full of uncertainties. It is not clear whether the existing approaches can be extended to the situation where the agent has imperfect perception of the shared environment.

An earlier work by Horacek (Horacek, 2005) has looked into the problem of mismatched knowledge between conversation partners for REG. The approach is a direct extension of the incremental algorithm (Dale, 1995). However, this work only provides a proof of concept example to illustrate the idea. No empirical evaluation was given.

All these previous works have motivated our present investigation. We are interested in REG under mismatched perceptual basis between conversation partners, where the agent has imperfect perception and knowledge of the shared environment. In particular, we took a well-studied graph-based approach (Krahmer et al., 2003) and extended it to incorporate group spatial relations and uncertainties associated with automated perception of the environment. The reason we chose a graph-based approach is that graph representations are widely used in the fields of computer vision (CV) and pattern recognition to represent spatially rich scenes. Nevertheless, the findings from this investigation provide insight to other approaches.



(a) an original scene

(b) the corresponding impoverished scene

Figure 1: An original scene and its impoverished scene processed by CV algorithm

3 Hypergraph-based REG

Towards mediating a shared perceptual basis in situated dialogue, our previous work (Liu et al., 2012) has conducted experiments to study referential communication between partners with mismatched perceptual capabilities. We simulated mismatched capabilities by making an original scene (Figure 1(a)) available to a director (simulating higher perceptual calibre) and a corresponding impoverished scene (Figure 1(b)) available to a matcher (simulating lowered perceptual calibre). The impoverished scene is created by re-rendering automated recognition results of the original scene by a CV algorithm. An example of the original scene and an impoverished scene is shown in Figure 1. Using this setup, the director and the matcher were instructed to collaborate with each other on some naming games. Through these games, they collected data on how partners with mismatched perceptual capabilities collaborate to ground their referential communication.

The setup in (Liu et al., 2012) is intended to simulate situated dialogue between a human (like the director) and a robot (like the matcher). The robot has a significantly lowered ability in perception and reasoning. The robot's internal representation of the shared world will be much like the impoverished scene which contains many recognition errors. The data from (Liu et al., 2012; Liu et al., 2013) shows that different strategies were used by conversation partners to produce referential descriptions. Besides directly describing attributes or binary relations with a relatum, they often use group-based descriptions (e.g., *a cluster of four objects on the right*). This is mainly due to the fact that some objects are simply not recognizable to the matcher. Binary spatial relationships sometimes are difficult to describe the target object, so the matcher must resort to group information to distinguish the target object from the rest of the objects. For example, suppose the matcher needs to describe the target object 5 in Figure 1(b), he/she may have to start by indicating the group of three objects at the bottom and then specify the relationship (i.e., top) of the target object within this group.

The importance of group descriptions has been shown not only here, but also in previous works on REG (Funakoshi et al., 2004; Funakoshi et al., 2006; Weijers, 2011). While the original graph-based approach can effectively represent attributes and binary relations between objects (Krahmer et al., 2003), it is insufficient to capture within-group or between-group relations. Therefore, to address the low perceptual capabilities of artificial agents, we introduce hypergraphs to represent the shared environment. Our approach has two unique characteristics compared to previous graph-based approaches: (1) A hypergraph representation is more general than a regular graph. Besides attributes and binary relations, it can also represent group-based relations. (2) Unlike previous work, here the generation of hypergraphs are completely driven by automated perception of the environment. This is done by incorporating uncertainties in perception and reasoning into cost functions associated with graphs. Next we

give a detailed account on hypergraph representation, cost functions incorporating uncertainties, and the search algorithm for REG.

3.1 Hypergraph Representation

A directed hypergraph G (Gallo et al., 1993) is a tuple of the form: $G = \langle X, A \rangle$, in which

$$X = \{x_m\}$$
$$A = \{a_i = (t_i, h_i) \mid t_i \subseteq X, h_i \subseteq X\}$$

Similar to regular graphs, a hypergraph consists of a set of nodes X and a set of arcs A. However, different from regular graphs, each arc in A is considered as a *hyperarc* in the sense that it can capture relations between any two subsets of nodes: a tail (t_i) and a head (h_i). Therefore, a hypergraph is a generalization of a regular graph. It becomes a regular graph if the cardinalities of both the tail and the head are restricted to one for all hyperarcs. While regular graphs are commonly used to represent binary relations between two nodes, hypergraphs provide a more general representation for n-ary relations among multiple nodes.

We use hypergraphs to represent the agent's perceived physical environment (also called *scene hypergraphs*). More specifically, each perceived object is represented by a node in the graph. Each perceived visual attribute of an object (e.g., color, size, type information) or a group of objects (e.g., number of objects in the group, location) is captured by a self-looping hyperarc. Hyperarcs are also used to capture the spatial relations between any two subsets of nodes, whether it is a relation between two objects, or between two groups of objects, or between one or more objects within a group of objects.

For example, Figure 2 shows a hypergraph created for part of the impoverished scene shown in Figure 1(b) (i.e., the upper right corner including objects 7, 8, 9, 11, and 13). One important characteristic is that, because the graph is created based on an automated vision recognition system, the values of an attribute or a relation in the hypergraph are numeric (except for the type attribute). For example, the value of the *color* attribute is the RGB distribution extracted from the corresponding visual object, the value of the *size* attribute is the width and height of the bounding box and the value of the *location* attribute is a function of spatial coordinates. These numerical features will be further converted to symbolic labels with certain confidence scores (described later in Section 3.3.2).

3.2 Hypergraph Pruning

The perceived visual scene can be represented as a complete hypergraph, in which any pair of two subsets of nodes are connected by a hyperarc. However, such a complete hypergraph is not only inefficient but also unnecessary. Instead of keeping all possible n-ary relations (i.e., hyperarcs), we only retain those relations that are likely used by humans to produce referring expressions, based on two heuristics.

The first heuristic is based on perceptual principles, also called the Gestalt Laws of perception (Sternberg, 2003), which describe how people group visually similar objects into entities or groups. Two well known principles of perceptual grouping are proximity and similarity (Wertheimer, 1938): objects that lie close together are often perceived as groups; objects of similar shape, size or color are more likely to form groups than objects differing along these dimensions. Based on these two principles, previous works have developed different algorithms for perceptual grouping (Thrisson, 1994; Gatt, 2006). In our investigation, we adopted Gatt's algorithm (Gatt, 2006), which has shown to be more accurate for spatial grouping. Given the results from spatial grouping, we only retain hyperarcs that represent spatial relations between two objects, between two perceived groups, between one object and a perceived group, or between one object and the group it belongs to.

The second heuristic is based on the observation that, given a certain orientation, people tend to use a relatum that is closer to the referent than more distant relata. In other words, it is less likely to refer to an object relative to a distant relatum when there is a closer relatum. For example, when referring to the stapler (object 9 in Figure 1(a)), it is more likely to use "the stapler above the battery" than "the stapler above the cellphone". Based on this observation, we prune the hypergraphs by only retaining hyperarcs between an object and their closest relata for each possible orientation.

Figure 2 shows the resulting hypergraph for representing a subset of objects (7, 8, 9, 11, and 13) in Figure 1(a).

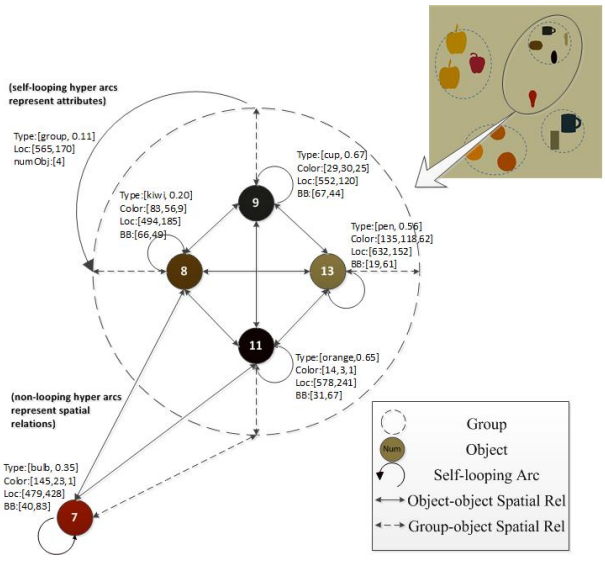


Figure 2: An example of hypergraph representing the perceived scene (a partial scene only including object 7, 8, 9, 11, 13 for Figure 1(a)).

3.3 Symbolic Descriptors for Attributes

As mentioned earlier, the values of attributes of objects and their relations are numerical in nature. In order for the agent to generate natural language descriptions, the first step is to assign symbolic labels or descriptors to those attributes and relations. Next we describe how we use a lexicon with grounded semantics in this process.

3.3.1 Lexicon with Grounded Semantics

Grounded semantics provides a bridge to connect symbolic labels or words with lower level visual features (Harnad, 1990). Previous work has developed various approaches for grounded semantics mainly for the reference resolution task, i.e., identifying visual objects in the environment given language descriptions (Dhande, 2003; Gorniak and Roy, 2004; Tenbrink and Moratz, 2003; Siebert and Schlangen, 2008; Liu et al., 2012). For the referring expression generation task here, we also need a lexicon with grounded semantics.

In our lexicon, the semantics of each category of words is defined by a set of semantic grounding functions that are parameterized on visual features. For example, for the *color* category it is defined as a multivariate Gaussian distribution based on the RGB distribution. Specific words such as *green, red, or blue* have different means and co-variances as the following:

$$color : red = f_r(\vec{v}_{color}) = N(\vec{v}_{color} \mid \mu_1, \sum_1)$$
$$color : green = f_g(\vec{v}_{color}) = N(\vec{v}_{color} \mid \mu_2, \sum_2)$$
$$color : blue = f_b(\vec{v}_{color}) = N(\vec{v}_{color} \mid \mu_3, \sum_3)$$

The above functions define how likely a set of recognized visual features (i.e., $\vec{v}_{color}$) describing the color dimensions (i.e., RGB distribution) is to match the color terms *red*, *green*, and *blue*.

For the spatial relation terms such as *above, below, left, right*, the semantic grounding functions take both vertical and horizontal coordinates of two objects, as follows [1]:

$$spatialRel : above(a, b) = f_{above}(\vec{v_{aloc}}, \vec{v_{bloc}})$$
$$= \begin{cases} 1 - \frac{|x_a - x_b|}{400} & \text{if } y_a < y_b; \\ 0 & \text{otherwise.} \end{cases}$$

Using the above convention, we have defined semantic grounding functions for *size* category words (e.g., *small* and *big*) and absolute position words (e.g., *top*, *below*, *left*, and *right*). In addition, we use object recognition models (Zhang and Lu, 2002) to define *class type* category words such as *apple* and *orange* used in our domain.

3.3.2 Attribute Descriptors and Cost Functions

Given the lexicon with grounded semantics as described above, the numerical attributes captured in the scene hypergraph can be converted to symbolic descriptors. For each attribute (e.g., *color*) or relation, the corresponding visual feature vector (i.e., $\vec{v}_{color}$) is plugged into the semantic grounding functions for the corresponding category of words. The word that best describes the attribute is chosen as the descriptor for that attribute. For example, given an RGB color distribution $\vec{v}_{color}$, we can find the color descriptor as follows:

$$color : w^* = \underset{red,green,blue}{\arg\max} \; f_w(\vec{v}_{color}),$$

For each attribute or relation, we can find a best descriptor in this manner. In addition, we also obtain a numerical value (returned from the semantic

[1] The size of the overall scene is 800x800.

grounding functions) that measures how well this descriptor describes the corresponding visual features. Intuitively, one would choose a descriptor that closely matches the visual features. Based on this intuition, we define the cost for each attribute A as the following:

$$cost(A) = 1 - f_{w^*}(\vec{v_A})$$

where w^* is the best descriptor for the attribute.

Given an attribute, the better the descriptor matches the extracted visual features, the lower the cost of the corresponding hyperarc.

3.4 Graph Matching for REG

Now the hypergraph representing the perceived environment has symbolic descriptors for its attributes and relations together with corresponding costs. Given this representation, REG can be formulated as a graph matching algorithm similar to that described in (Krahmer et al., 2003). We use the same Branch and Bound algorithm described in (Krahmer et al., 2003). In this approach, a hypothesis hypergraph (starting with one node representing the target object) is gradually expanded by adding in a least cost hyperarc from the scene hypergraph. At each expansion, the hypothesis graph is matched against the scene hypergraph to decide whether it matches any nodes other than the target node in the scene hypergraph. The expansion stops if the hypothesis graph does not cover any other nodes except for the target node. At this point, the hypothesis graph captures all the content (e.g., attributes and relations) required to uniquely describe the target object. We then apply a set of simple generation templates to generate the surface form of referring expressions based on the hypothesis graph.

4 Empirical Evaluations

4.1 Evaluation Setup

To evaluate the performance of this hypergraph-based approach to REG, we conducted a comparative study using crowd-sourcing. More specifically, we created 48 different scenes similar to that in Figure 1(a). Each scene has 13 objects on average and there are 621 objects in total. For each of these scenes, we applied a CV algorithm (Zhang and Lu, 2002) and generated scene hypergraphs as described in Section 3.1. We then use different generation strategies (varied in terms of graph representations and cost functions, to be explained in Section 4.2) to automatically generate referring expressions to refer to each object.

To evaluate the quality of these generated referring expressions, we applied Amazon Mechanical Turk to solicit feedback from the crowd [2]. Through an interface, we displayed an original scene and generated referring expressions (from different generation strategies) in a random order. We asked each turk to select the object in the scene that he/she believed was the one referred to by the shown referring expression (i.e., reference identification task). Each referring expression received three votes from the crowd. In total, 217 turks participated in our experiment.

4.2 Generation Strategies

We applied a set of different strategies to generate referring expressions for each object. The variations lie in two dimensions: (1) different graph representations: using a hypergraph to represent the perceived scene as described in Section 3.1 versus using a regular graph as introduced in (Krahmer et al., 2003); and (2) different cost functions for attributes and relations: cost functions that have been used in previous works (Theune et al., 2007; Krahmer et al., 2008) and cost functions that incorporate uncertainties of perception as described in Section 3.3.2.

Cost functions play an important role in graph-based approaches (Krahmer et al., 2003). Previous works have examined different types of cost functions (Theune et al., 2007; Krahmer et al., 2008; Theune et al., 2011). We adopted some commonly used cost functions from previous work together with the cost functions defined here. In particular, we experimented with the following different cost functions:

Simple Cost: The costs for all hyperarcs are set to 1. With this cost function, the graph-based algorithm resembles the Full Brevity algorithm of Dale (Dale,

[2] To control the quality of crowdsourcing, we recruited participants based on the following criteria: Participants' locations are limited to the United States. Approval rate for each participant's previous work is greater than or equal to 95%, and the number of each participant's previous approved work is greater than or equal to 1000.

1992) in that a shortest distinguishing description is preferred.

Absolute Preferred: The costs for hyperarcs representing absolute attributes (e.g., type, color, and position) are set to 1. The costs for relative attributes (e.g., size) and relations are set to 2. This cost function mimics human's preference for absolute attributes over relative ones (Dale, 1995).

Relative Preferred: The costs for hyperarcs representing absolute attributes are set to 2 and for relative attributes and relations are set to 1. This cost function has been applied previously to emphasize the importance of spatial relations in REG (Viethen and Dale, 2008).

Uncertainty Based: The costs for all hyperarcs are defined by incorporating uncertainties from perception as described in Section 3.3.2.

Uncertainty Relative Preferred: To emphasize the importance of spatial relations as demonstrated in situated interaction (Tenbrink and Moratz, 2003; Kelleher and Kruijff, 2006), the costs for hyperarcs representing relative attributes and relations are divided by 3. This cost function will allow the algorithm to prefer spatial relations through the reduced cost.

Note that we only tested a few (not all) commonly used cost functions proposed by previous work (Krahmer et al., 2003; Theune et al., 2007; Krahmer et al., 2008; Theune et al., 2011). For example, we did not include the stochastic cost function which is defined based on the frequencies of attribute selection from the training data (Krahmer et al., 2003). On the one hand, we did not have a large set of human descriptions of the impoverished scene to learn the stochastic cost. On the other hand, it is not clear whether human strategies of describing the impoverished scene should be used to represent optimal strategies for the robot. Nevertheless, the above different cost functions will allow us to evaluate whether incorporating perceptual uncertainties will make a difference in the REG performance.

4.3 Evaluation Results

As mentioned earlier, each generated referring expression received three independent votes regarding its referent from the crowd. The referent with the most votes is taken as the predicted referent and is used for evaluation. If all three votes are different, then by default, it is deemed that the referent is not correctly identified for that expression. We use the *accuracy* of the referential identification task (i.e., the percentage of generated referring expressions where the referents are correctly identified) as the metric to evaluate different generation strategies illustrated in Section 4.2.

Cost Function	Regular Graph	Hypergraph
Simple Costs	33.2%	33.3%
Absolute Preferred	30.1%	30.3%
Relative Preferred	31.1%	35.4%
Uncertainty Based	35.7%	37.5%
Uncertainty Rel. Prefer.	36.7%	45.2%

Table 1: Results with different cost functions

4.3.1 The Role of Cost Functions

Table 1 shows the results based on different cost functions and different graph representations. There are several observations.

First, when the agent does not have perfect knowledge of the environment and has to automatically infer the environment as in our setting here, cost functions based on uncertainties of perception lead to better results. This occurs for both regular graphs and hypergraphs. This result is not surprising and indicates that cost functions should be tied to the agent's ability to perceive and infer the environment. The uncertainty based cost functions allow the agent to prefer reliable attributes or relations.

Second, consistent with previous work (Viethen and Dale, 2008), we observed the importance of spatial relations. Especially when the perceived world is full of uncertainties, spatial relations tend to be more reliable. In particular, as shown in Table 1, using hypergraphs enables generating group-based relations and results in significantly better performance (45.2%) compared to regular graphs (36.7%) ($p = 0.002$).

Note that our current cost function only includes uncertainties of the agent's own perception in a simplistic form. When humans and agents have mismatched perceptual basis, the human's model of comprehension and tolerance of inaccurate description could play a role in REG. Incorporating human models in the cost function will require in-depth empirical studies and we will leave that to our future

work.

4.3.2 The Role of Imperfect Perception

To further understand the role of hypergraphs in mediating mismatched perceptions between humans and agents, we created a perfect scene regular graph and a perfect scene hypergraph (representing the agent's perfect knowledge of the environment) for each of the 48 scenes used in the experiments. In each of these scene graphs, the attribute and relation descriptors are manually provided. We further applied the *Absolute Preferred* cost function (which has shown competitive performance in previous work) to generate referring expressions for each object. Again, each referring expression received three votes from the crowd.

Table 2 shows the results comparing two conditions: (1) REs generated (by the *Absolute Preferred* cost function) based on the perfect graphs which represent the agent's perfect knowledge and perception of the environment; and (2) REs generated based on automatically created graphs (by the *Uncertainty Relative Preferred* cost function) which represent the agent's imperfect knowledge of the environment as a result of automated recognition and inference. The result shows that given perfect knowledge of the environment, hypergraphs only perform marginally better than the regular graphs ($p = 0.07$). Given imperfect knowledge of the environment, hypergraphs significantly outperforms the regular graphs by taking advantage of spatial grouping information ($p = 0.002$). It is worthwhile to mention that currently we use spatial proximity to identify groups. However, the hypergraph based approach is not restricted to spatial grouping. In theory, it can represent any type of group based on different similarity criteria.

Furthermore, our result shows that the graph-based approaches perform quite competitively under the condition of perfect knowledge and perception. Although evaluated on different data sets, this result is consistent with results from previous work (Gatt and Belz, 2008; Gatt et al., 2009). However, what is more interesting here is that while graph-based approaches perform well when the agent has perfect knowledge of the environment, as its human partner, these approaches literally fall apart with close to 40% performance degradation when applied to the situation where the agent's representation of the shared world is problematic and full of mistakes.

Environment	Regular Graph	Hypergraph
Pefect Perception	80.4%	84.2%
Imperfect Perception	36.7%	45.2%

Table 2: Results of comparing perfect perception and imperfect perception of the shared world.

These results indicate that REG for automatically perceived scenes can be extremely challenging. Many errors result from automated perception and reasoning that will affect the internal representation of the world and thus the generated REs. In our experiments here, we applied a very basic CV algorithm which resulted in rather poor performance in our data: overall, 60.3% of objects in the original scene are mis-recognized, and 10.5% of objects are mis-segmented. We think this poor CV performance represents a more challenging problem.

Some errors such as recognition errors can be bypassed using our current approach based on hypergraphs. For example, in Figure 1 target object 9 (a stapler) and 13 (a key) are mis-recognized as a cup and a pen. Using our hypergraph-based approach, for the target object 9, instead of generating "a small cup" (as in the case of using regular graphs), "a gray object on the top within a cluster of four objects" is generated. For the target object 13, instead of "a pen" as generated by regular graphs, "a small object on the right within a cluster of 4" is generated. Even with recognition errors, these group-based descriptions will allow the listener to identify target objects in their representation correctly. Nevertheless, many processing errors cannot be handled by our current approach. For example, an object can be mistakenly segmented into multiple parts or several objects can be mistakenly grouped into one object. In addition, our current semantic grounding functions are simple. Sometimes they do not provide correct descriptors for the extracted visual features. More sophisticated functions that better reflect human's visual perception (Regier, 1996; Mojsilovic, 2005; Mitchell et al., 2011) should be pursued in the future.

	Minimum Effort	Extra Effort
Pefect Perception	84.2%	88.1%
Imperfect Perception	45.2%	51.5%

Table 3: Results of comparing minimum effort and extra effort using hypergraphs

4.3.3 The Role of Extra Effort

While REG systems have a tendency to produce minimal descriptions, recent psycholinguistic studies have shown that speakers do not necessarily follow the Grice's maxim of quantity, and they tend to provide redundant properties in their descriptions (Jordan and Walker, 2000; Belke and Meyer, 2002; Arts et al., 2011). With this in mind, we conducted a very simple evaluation on the role of extra effort. Once a set of descriptors are selected based on the minimum cost, one additional descriptor (with the least cost among the remaining attributes or relations) is added to the referential description. We once again solicited the crowd feedback to this set of expressions generated by extra effort. Each expression again received three votes from the crowd.

Table 3 shows the results by comparing minimum effort with extra effort when using hypergraphs to generate REs. As indicated here, extra effort (by adding one additional descriptor) leads to more comprehensible REs with 3.9% improvement under perfect perception and 6.3% improvement under imperfect perception (both are significant, $p < 0.05$). The improvement is larger under imperfect perception. This seems to indicate that exploring extra effort in REG could help mediate mismatched perceptions in situated dialogue. However, more understanding on how to engage in such extra effort will be required in the future.

5 Conclusion

In situated dialogue, humans and agents have mismatched perceptions of the shared environment. To facilitate successful referential communication between a human and an agent, the agent needs to take such discrepancies into consideration and generate referential descriptions that can be understood by its human partner. With this in mind, we re-visited the problem of referring expression generation in the context of mismatched perceptions between humans and agents. In particular, we applied and extended the state of the art graph-based approach (Krahmer et al., 2003) in this new setting. Our empirical results have shown that, to address the agent's limited perceptual capability, REG algorithms will need to take into account the uncertainties in perception and reasoning. Group-based information appears more reliable and thus should be modeled by an approach that deals with automated perception of spatially rich scenes.

While graph-based approaches have shown effective for the situation where the agent has complete knowledge of the environment, as its human partner, these approaches are often inadequate when humans and agents have mismatched representations of the shared world. Our empirical results here call for new solutions to address the mismatched perceptual basis. Previous work indicated that referential communication is a collaborative process (Clark and Wilkes-Gibbs, 1986; Heeman and Hirst, 1995). Conversation partners make extra effort to collaborate with each other. For the situation with mismatched perceptual basis, a potential solution thus should go beyond the objective of generating a minimum description, and towards a collaborative model which incorporates immediate feedback from the conversation partner (Edmonds, 1994).

6 Acknowledgments

This work was supported by N00014-11-1-0410 from the Office of Naval Research and IIS-1208390 from the National Science Foundation.

References

Anja Arts, Alfons Maes, Leo Noordman, and Carel Jansen. 2011. Overspecification facilitates object identification. *Journal of Pragmatics*, 43(1):361–374.

E. Belke and A. S. Meyer. 2002. Tracking the time course of multidimensional stimulus discrimination: Analyses of viewing patterns and processing times during "same"-"different" decisions. *European Journal of Cognitive Psychology*, 14(2):237–266.

H.H. Clark and S.E. Brennan. 1991. Grounding in communication. *Perspectives on socially shared cognition*, 13:127–149.

H. H Clark and D Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22:1–39.

Madalina Croitoru and Kees Van Deemter. 2007. A conceptual graph approach to the generation of referring expressions. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2456–2461.

Robert Dale. 1989. Cooking up referring expressions. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, ACL '89, pages 68–75, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Dale. 1992. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes.* The MIT Press,Cambridge, Massachusetts.

Robert Dale. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19:233–263.

Sheel Sanjay Dhande. 2003. A computational model to connect gestalt perception and natural language. In *Masters thesis, Massachusetts Institure of Technology*.

Philip G. Edmonds. 1994. Collaboration on reference to objects that are not mutually known. In *Proceedings of the 15th conference on Computational linguistics - Volume 2*, COLING '94, pages 1118–1122, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kotaro Funakoshi, Satoru Watanabe, Naoko Kuriyama, and Takenobu Tokunaga. 2004. Generation of relative referring expressions based on perceptual grouping. In *COLING*.

Kotaro Funakoshi, Satoru Watanabe, and Takenobu Tokunaga. 2006. Group-based generation of referring expressions. In *INLG*, pages 73–80.

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

Albert Gatt and Anja Belz. 2008. Attribute selection for referring expression generation: new algorithms and evaluation methods. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, pages 50–58, Stroudsburg, PA, USA. Association for Computational Linguistics.

Albert Gatt, Ielka van der Sluis, and Kees van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 49–56, Stroudsburg, PA, USA. Association for Computational Linguistics.

Albert Gatt, Anja Belz, and Eric Kow. 2009. The tuna-reg challenge 2009: overview and evaluation results. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 174–182, Stroudsburg, PA, USA. Association for Computational Linguistics.

Albert Gatt. 2006. Structuring knowledge for reference generation: A clustering algorithm. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics*, pages 321–328.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 410–419, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter Gorniak and Deb Roy. 2004. Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research*, 21:429–470.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D*, 42:335–346.

Peter A. Heeman and Graeme Hirst. 1995. Collaborating on referring expressions. *Computational Linguistics*, 21:351–382.

Helmut Horacek. 2005. Generating referential descriptions under conditions of uncertainty. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG) pages 58-67, Aberdeen, UK*.

Pamela W Jordan and Marilyn Walker. 2000. Learning attribute selections for non-pronominal expressions. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pages 181-190.*

John D. Kelleher and Geert-Jan M. Kruijff. 2006. Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1041–1048, Stroudsburg, PA, USA. Association for Computational Linguistics.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *computational linguistics*, 38(1):173–218.

Emiel Krahmer Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, March.

Emiel Krahmer, Mariet Theune, Jette Viethen, and Iris Hendrickx. 2008. Graph: The costs of redundancy in referring expressions. In *In Proceedings of the 5th International Conference on Natural Language Generation, Salt Fork OH, USA*.

Changsong Liu, Rui Fang, and Joyce Y. Chai. 2012. Towards mediating shared perceptual basis in situated dialogue. In *Proceedings of the 13th Annual Meeting of*

the Special Interest Group on Discourse and Dialogue, SIGDIAL '12, pages 140–149, Stroudsburg, PA, USA. Association for Computational Linguistics.

Changsong Liu, Rui Fang, Lanbo She, and Joyce Y. Chai. 2013. Modeling collaborative referring for situated referential grounding. In *The 14th Annual SIGdial Meeting on Discourse and Dialogue*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2011. Two approaches for generating size modifiers. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2013. Generating expressions that refer to visible objects. In *Proceedings of NAACL-HLT 2013, pages 1174-1184*.

Aleksandra Mojsilovic. 2005. A computational model for color naming and describing color composition of images. *IEEE Transactions on Image Processing*, 14:690 – 699.

Terry Regier. 1996. *The human semantic potential*. The MIT Press,Cambridge, Massachusetts.

Deb Roy. 2002. Learning visually grounded words and syntax of natural spoken language. *Evolution of Communication*, 4.

Alexander Siebert and David Schlangen. 2008. A simple method for resolution of definite reference in a shared visual context. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, SIGdial '08, pages 84–87, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Sternberg. 2003. *Cognitive Psychology,Third Edition*. Thomson Wadsworth.

Kristina Striegnitz, Hendrik Buschmeier, and Stefan Kopp. 2012. Referring in installments: a corpus study of spoken object references in an interactive virtual environment. In *Proceedings of the Seventh International Natural Language Generation Conference*, INLG '12, pages 12–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thora Tenbrink and Reinhard Moratz. 2003. Group-based spatial reference in linguistic human-robot interaction. *Spatial Cognition and Computation*, 6:63–106.

Mariët Theune, Pascal Touset, Jette Viethen, and Emiel Krahmer. 2007. Cost-based attribute selection for gre (graph-sc/graph-fp). In *Proceedings of the MT Summit XI Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*.

Mariët Theune, Ruud Koolen, Emiel Krahmer, and Sander Wubben. 2011. Does size matter – how much data is required to train a reg algorithm? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 660–664, Portland, Oregon, USA, June. Association for Computational Linguistics.

Kristinn R. Thrisson. 1994. Simulated perceptual grouping: An application to human-computer interaction. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 876–881.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, pages 59–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. Weijers. 2011. Referring expressions with groups as landmarks. volume 15. University of Twente.

Max Wertheimer. 1938. *Laws of organization in perceptual forms. A Source Book of Gestalt Psychology*. Routledge and Kegan Paul, London.

Dengsheng Zhang and Guojun Lu. 2002. An integrated approach to shape based image retrieval. In *Proc. of 5th Asian Conference on Computer Vision (ACCV*, pages 652–657.

Open-Domain Fine-Grained Class Extraction from Web Search Queries

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

Abstract

This paper introduces a method for extracting fine-grained class labels (*"countries with double taxation agreements with india"*) from Web search queries. The class labels are more numerous and more diverse than those produced by current extraction methods. Also extracted are representative sets of instances (*singapore*, *united kingdom*) for the class labels.

1 Introduction

Motivation: As more semantic constraints are added, concepts like *companies* become more specific, e.g., *companies* that are in the *software* business, and have been *started in a garage*. The sets of instances associated with the classes become smaller; the class labels used to concisely describe the meaning of more specific concepts tend to become longer. In fact, fine-grained class labels such as *"software companies started in a garage"* are often complex noun phrases, since they must somehow summarize multiple semantic constraints. Although Web users are interested in both coarse (e.g., *"companies"*) and fine-grained (e.g., *"software companies started in a garage"*) class labels, virtually all class labels acquired from text by previous extraction methods (Etzioni et al., 2005; Van Durme and Paşca, 2008; Kozareva and Hovy, 2010; Snow et al., 2006) exhibit little syntactic diversity. Indeed, instances and class labels that are relatively complex nouns are known to be difficult to detect and pick out precisely from surrounding text (Downey et al., 2007). This and other challenges associated with large-scale extraction from Web text (Etzioni et al., 2011) cause the extracted class labels to usually follow a rigid modifiers-plus-nouns format. The format covers nouns (*"companies"*) possibly preceded by one or many modifiers (*"software companies"*, *"computer security software companies"*). Examples of actual extractions include *"european cities"* (Etzioni et al., 2005), *"strong acids"* (Pantel and Pennacchiotti, 2006), *"prestigious private schools"* (Van Durme and Paşca, 2008), *"aquatic birds"* (Kozareva and Hovy, 2010).

As an alternative to extracting class labels from text, some methods simply import them from human-curated resources, for example from the set of categories encoded in Wikipedia (Remy, 2002). As a result, class labels potentially exhibit higher syntactic diversity. The modifiers-plus-nouns format (*"computer security software companies"*) is usually still the norm. But other formats are possible: *"software companies based in london"*, *"software companies of the united kingdom"*. Vocabulary coverage gaps remain a problem, with many relevant class labels (*"software companies of texas"* *"software companies started in a garage"*, *"software companies that give sap training"*) still missing. There is a need for methods that more aggressively identify fine-grained class labels, beyond those extracted by previous methods or encoded in existing, manually-created resources. Such class labels increase coverage, for example in scenarios that enrich Web search results with instances available for the class labels specified in the queries.

Contributions: The contributions of this paper are twofold. First, it proposes a weakly-supervised

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 403–414,
Seattle, Washington, USA, 18-21 October 2013.

method to assemble a large vocabulary of class labels from queries. The class labels include fine-grained class labels (*"countries with double taxation agreements with india"*, *"no front license plate states"*) that are difficult to extract from text by previous methods for open-domain information extraction. Second, the method acquires representative instances (*singapore*, *united kingdom*; *arizona*, *new mexico*) that belong to fine-grained class labels (*"countries with double taxation agreements with india"*, *"no front license plate states"*). Both class labels and their instances are extracted from Web search queries.

2 Extraction from Queries

2.1 Extraction of Class Labels

Overview: Given a set of arbitrary Web search queries as input, our method produces a vocabulary of fine-grained class labels. For this purpose, it: a) selects an initial vocabulary of class labels, as a subset of input queries that are likely to correspond to search requests for classes; b) expands the vocabulary, by generating a large, noisy set of other possible class labels, through replacements of ngrams within initial class labels with their similar phrases; c) restricts the generated class labels to those that match the syntactic structure of class labels within the initial vocabulary; and d) further restricts the generated class labels to those that appear within the larger set of arbitrary Web search queries.

Initial Vocabulary of Class Labels: Out of a set of arbitrary search queries available as input, the queries in the format *"list of .."* are selected as the initial vocabulary of class labels. The prefix *"list of"* is discarded from each query. Thus, the query *"list of software companies that use linux"* gives the class label *"software companies that use linux"*.

Generation via Phrase Similarities: As a prerequisite to generating class labels, distributionally similar phrases (Lin and Pantel, 2002; Lin and Wu, 2009; Pantel et al., 2009) and their scores are collected in advance. A phrase is represented as a vector of its contextual features. A feature is a word, collected from windows of three words centered around the occurrences of the phrase in sentences across Web documents (Lin and Wu, 2009). In the contextual vector of a phrase, the weight of a feature is the pointwise-mutual information (Lin and Wu, 2009) between the phrase P and the feature F. The distributional similarity score between two phrases is the cosine similarity between the contextual vectors of the two phrases. The lists of most distributionally similar phrases of a phrase P are thus compiled offline, by ranking the similar phrases of P in decreasing order of their similarity score relative to P.

Each class label from the initial vocabulary is expanded into a set of generated, candidate class labels. To this effect, every ngram P within a given class label is replaced with each of the distributionally similar phrases, if any, available for the ngram. As shown later in the experimental section, the expansion can increase the vocabulary by a factor of 100.

Approximate Syntactic Filtering: The set of generated class labels is noisy. The set is filtered, by retaining only class labels whose syntactic structure matches the syntactic structure of some class label(s) from the initial vocabulary. The syntactic structure is loosely approximated at surface rather than syntactic level. A generated class label is retained, if its sequence of part of speech tags matches the sequence of part of speech tags of one of the class labels from the initial vocabulary. As an additional constraint, the sequence must contain one tag corresponding to a common noun in plural form, i.e., NNS. Otherwise, the class label is discarded.

Query Filtering: Generated class labels that pass previous filters are further restricted. They are intersected with the set of arbitrary Web search queries available as input. Generated class labels that are not full queries are discarded.

2.2 Extraction of Instances

Overview: Our method mines instances of fine-grained class labels from queries. In a nutshell, it identifies queries containing two types of information simultaneously. First, the queries contain an instance (*marvin gaye*) of the more general class labels (*"musicians"*) from which the fine-grained class labels (*"musicians who have been shot"*) can be obtained. Second, the queries contain the constraints added by the fine-grained class labels (*"... shot"*) on top of the more general class labels.

Instances of General Class Labels: Following (Ponzetto and Strube, 2007), the Wikipedia category network is refined into a hierarchy that discards

non-IsA (thematic) edges, and retains only IsA (subsumption) edges from the network (Ponzetto and Strube, 2007). Instances, i.e., titles of Wikipedia articles, are propagated upwards to all their ancestor categories. The class label *"musicians"* would be mapped into *madonna, marvin gaye, jon bon jovi* etc. The mappings from each ancestor category, to all its descendant instances in the Wikipedia hierarchy, represent our mappings from more general class labels to instances.

Decomposition of Fine-Grained Class Labels: A fine-grained class label (e.g., *"musicians who have been shot"*) is effectively decomposed into pairs of two pieces of information. The first piece is a more general class label (*"musicians"*), if any occurs in it. The second piece is a bag of words, collected from the remainder of the fine-grained class label after discarding stop words. Note that the standard set of stop words is augmented with auxiliary verbs (e.g., *does*, *has*, *is*, *would*), determiners, conjunctions, prepositions, and question wh-words (Radev et al., 2005) (e.g., *where*, *how*). In the first piece of each pair, the general class label is then replaced with each of its instances. This produces multiple pairs of a candidate instance and a bag of words, for each fine-grained class label. As an illustration, the class labels *"musicians who have been shot"* and *"automobiles with remote start"* are decomposed into pairs like <*madonna*, {*shot*}>, <*marvin gaye*, {*shot*}>; and <*buick lacrosse*, {*remote*, *start*}>, <*nissan versa*, {*remote*, *start*}>, respectively.

Matching of Candidate Instances: A decomposed class label is retained, if there are matching queries that contain the candidate instance, the bag of words, and optionally stop words. Otherwise, the decomposed class label is discarded. The word matching is performed after word stemming (Porter, 1980). The aggregated frequency of the matching queries is assigned as the score of the candidate instance for the fine-grained class label:

$$Score(I, C) = \sum_{Q} (Freq(Q) | Match(Q, < I, C >)) \quad (1)$$

For example, the score of the candidate instance *marvin gaye* for the class label *"musicians who have been shot"*, is the sum of the frequencies of the matching queries *"marvin gaye is shot"*, *"when was marvin gaye shot"*, *"why marvin gaye was shot"* etc. Similarly, the score of *buick lacrosse* for *"automobiles with remote start"* is given by the aggregated frequencies of the queries *"buick lacrosse remote start"*, *"how to remote start buick lacrosse"*, *"remote start for buick lacrosse"*. Candidate instances of a class label are ranked in decreasing order of their scores.

3 Experimental Setting

Web Textual Data: The experiments rely on a sample of 1 billion queries in English submitted by users of a Web search engine. Each query is accompanied by its frequency of occurrence. Also available is a sample of around 200 million Web documents in English.

Phrase Similarities: Web documents are used in the experiments only to construct a phrase similarity repository following (Lin and Wu, 2009; Pantel et al., 2009). The repository contains ranked lists of the top 1000 phrases, computed to be the most distributionally similar to each of around 16 million phrases.

Text Pre-Processing: The TnT tagger (Brants, 2000) assigns part of speech tags to words in class labels.

Instances: To collect mappings from Wikipedia categories (as more general class labels) to titles of descendant Wikipedia articles (as instances), a snapshot of Wikipedia articles was intersected with the Wikipedia category hierarchy from (Ponzetto and Strube, 2007). The mappings connect a total of 1,535,083 instances to a total of 108,756 class labels.

4 Evaluation of Class Labels

4.1 Evaluation Procedure

Experimental Runs: Human-compiled information available within Wikipedia serves as the source of data for two baseline runs. The set of all categories, listed in Wikipedia for any of its articles, corresponds to the set of class labels "acquired" in run R_{wc}. Categories used for internal Wikipedia bookkeeping (Ponzetto and Strube, 2007) are discarded. Their names contain one of the words *article(s)*, *category(ies)*, *indices*, *pages*, *redirects*, *stubs*, or *templates*. Similarly, the titles of Wikipedia articles with the prefix *"List of ..."* (e.g., *"List of automobile manufacturers of Germany"*) form the set of class labels

"acquired" in run R_{wl}. The prefix *"List of"* is discarded.

For completeness, a third baseline run, R_{dc}, corresponds to class labels extracted from Web documents. The class labels are noun phrases *C* that fill extraction patterns equivalent to *"C such as I"*. The patterns are matched to document sentences. The boundaries of the class labels *C* are approximated from part of speech tags of sentence words (Van Durme and Paşca, 2008). The patterns were proposed in (Hearst, 1992). They were employed widely in subsequent methods (Etzioni et al., 2005; Kozareva et al., 2008; Wu et al., 2012), which extract class labels precisely from the set of class labels *C* produced by the extraction patterns. Even methods using queries as a textual data source still extract class labels from documents using the same extraction patterns (Paşca, 2010). Therefore, from the point of view of evaluating class labels, run R_{dc} is a valid representative of previous extraction methods, including (Etzioni et al., 2005; Kozareva et al., 2008; Van Durme and Paşca, 2008; Paşca, 2010; Wu et al., 2012).

Besides the baseline runs, three experimental runs are considered. In run R_{ql}, the queries starting with the prefix *"list of"* form the set of class labels. The prefix *"list of"* is discarded from each query. In run R_{qg}, the class labels are generated via phrase similarities, starting from R_{ql} as an initial set of class labels. Run R_{qa} represents an ablation experiment. It is created from R_{qg}, by limiting the expansion of a given class label via distributional similarities to only one, rather than multiple, phrases within the class label. Note that, by design, none of the class labels that appear in R_{ql} also appear in runs R_{qa} or R_{qg}. Therefore, the intersection between R_{ql}, on one hand, and R_{qa} and R_{qg}, on the other hand, is the empty set.

All data, including the class labels extracted in all experimental runs, is converted to lower case.

4.2 Relative Coverage of Class Labels

Coverage Over Entire Sets: Table 1 illustrates the overall coverage of the various experimental runs. The table takes all class labels into account, relative to the Wikipedia-based runs as reference sets: R_{wc} (Wikipedia categories), in the upper part of the table; and R_{wl} (Wikipedia List-Of categories), in the lower part of the table. Note that the number of class labels extracted by the individual run shown in the second column (B) is shown in the fourth column (|B|). In particular, there are around 1.6 million unique *"list of .."* queries, from which class labels are collected in run R_{ql}.

Counts					Cvg
A	B	|A|	|B|	|A∩B|	$\frac{\|A \cap B\|}{\|A\|}$
vs. Wikipedia categories:					
R_{wc}	R_{dc}	295,587	2,884,390	15,011	0.051
	R_{ql}	295,587	1,649,261	21,979	0.074
	R_{qa}	295,587	33,073,741	33,502	0.113
	R_{qg}	295,587	134,235,151	43,935	0.148
	$R_{ql} \cup R_{qg}$	295,587	135,884,412	65,914	0.222
vs. Wikipedia categories that are queries:					
$R_{wc} \cap Q$	R_{dc}	126,318	2,884,390	14,840	0.117
	R_{ql}	126,318	1,649,261	21,979	0.173
	R_{qa}	126,318	33,073,741	33,502	0.265
	R_{qg}	126,318	134,235,151	43,935	0.347
	$R_{ql} \cup R_{qg}$	126,318	135,884,412	65,914	0.521
vs. Wikipedia List-Of categories:					
R_{wl}	R_{dc}	134,840	2,884,390	8,099	0.060
	R_{ql}	134,840	1,649,261	26,446	0.196
	R_{qa}	134,840	33,073,741	16,204	0.120
	R_{qg}	134,840	134,235,151	20,021	0.148
	$R_{ql} \cup R_{qg}$	134,840	135,884,412	46,467	0.344
vs. Wikipedia List-Of categories that are queries:					
$R_{wl} \cap Q$	R_{dc}	47,442	2,884,390	7,985	0.168
	R_{ql}	47,442	1,649,261	24,821	0.523
	R_{qa}	47,442	33,073,741	16,204	0.341
	R_{qg}	47,442	134,235,151	20,021	0.422
	$R_{ql} \cup R_{qg}$	47,442	135,884,412	44,842	0.945

Table 1: Coverage of class labels extracted by various experimental runs, relative to class labels available in Wikipedia before and after intersecting them with a large set of arbitrary queries (A = reference set, relative to which coverage is computed; B = measured set, for which coverage is computed relative to the reference set; |A| = size of set A; Q = set of input queries)

During the computation of coverage, the reference set, and the set for which coverage is being computed, are intersected. Intersection relies on strict string matching. All words, including punctuation, must match exactly in order for a class label to be part of the intersection. The reference sets are intersected with the set of all Web search queries Q used in the experiments. Coverage is computed both before and after intersection. Less than half (126,318 of 295,587) of the class labels, for

the reference set R_{wc}; and about a third (47,442 of 134,840) for R_{wl}; appear in the set Q of all queries.

Three conclusions can be drawn from the results. First, query-based runs vastly outperform Wikipedia-based runs in terms of absolute coverage. Run R_{ql} contains around 5 and 12 times more class labels, than R_{wc} and R_{wl} respectively. On top of that, generating class labels via phrase similarities further increases the class label count by about 20 times for R_{qa}, and 80 times for R_{qg}. Second, query-based runs R_{qa} and R_{qg} surpass the document-based run R_{dc}. Third, higher class label counts translate into higher relative coverage. In the upper part of the table, run R_{wl} contains 3.9% (relative to R_{wc}) and 7.1% (relative to $R_{wc} \cap Q$) of the reference set. But the relative coverage doubles for R_{ql} at 7.4% (relative to R_{wc}) and 17.3% (relative to $R_{wc} \cap Q$). Coverage again doubles for R_{qg} at 14.8% (relative to R_{wc}) and 34.7% (relative to $R_{wc} \cap Q$). The union of query-based initial and generated class labels is $R_{ql} \cup R_{qg}$. The union contains about a quarter (i.e., 22.2%) or half (52.1%) of the reference set R_{wc}, depending on whether the reference set is intersected with the set of all queries or not. In the lower part of the table, more than 90% of the queries in the reference set R_{wl} that are also queries are found among the class labels collectively extracted in the query-based runs. Note that, since R_{ql} is disjoint from R_{qa} and R_{qg}, none of the class labels already in R_{ql} can be "re-discovered" (generated) again in R_{qa} or R_{qg}. Therefore, by experimental design, relative coverage scores of R_{ql} may be relatively difficult to surpass by R_{qa} or R_{qg} taken individually.

Diversity: Class labels restricted to those that have the format *".. that/which/who .."* are relatively more specific, e.g., *"grocery stores that double coupons in omaha"*, *"airlines which fly from santa barbara"*, *"writers who were doctors"*. The most frequent head phrases of such restricted class labels offer an idea about how diverse the class labels are. The counts of class labels for the most frequent head phrases are in the order of 10's in the case of R_{wl} vs. 10,000's for R_{qg}. In comparison, none of the class labels of run R_{dc} have this format. The lack of such class labels in run R_{dc}, and their smaller proportion in run R_{wl} vs. R_{qg}, suggest that class labels extracted by the proposed method exhibit higher lexical and syntactic diversity than previous methods do.

Tag (Value): Examples of Class Labels
correct (1.0): angioplasty specialists in kolkata, good things pancho villa did, eating disorders inpatient units in the uk nhs specialist services
questionable (0.5): picture framers adelaide cbd, side effects bicalutamide, different eating disorders, private hospitals treat kidney stones uk
incorrect (0.0): al hirschfield theatre hours, value of berkshire hathaway shares, remove spaces in cobol, dogs with loss of appetite, 1999 majorca open

Table 2: Correctness tags manually assigned to class labels containing one of the (underlined) target phrases, extracted by various runs

4.3 Precision of Class Labels

Evaluation Metric: Class labels being evaluated are manually assigned a correctness tag. A class label is deemed *correct*, if it is grammatically well-formed and describes a relevant concept that embodies some (unspecified) set of instances that share similar properties; *questionable*, if it is relevant but not well-formed; or *incorrect*. A *questionable* class label is not well-formed because it lacks necessary linking particles (e.g., the prepositions *of* or *for* in *"side effects bicalutamide"*), or contains undesirable modifiers (*"different eating disorders"*). Examples of *correct* and *incorrect* class labels are *"angioplasty specialists in kolkata"* and *"al hirschfield theatre hours"* respectively.

To compute the precision score, the correctness tags are converted to numeric values, as shown in Table 2: *correct* to 1; *questionable* to 0.5; and *incorrect* to 0. Precision over a list of class labels is measured as the sum of the correctness values of the class labels in the list, divided by the size of the list.

Precision Relative to Target Phrases: The precision of the class labels in each run is determined similarly to how relative coverage was computed earlier. More precisely, the precision is computed over the class labels whose names contain each phrase from the set of 75 target phrases from (Alfonseca et al., 2010). For each phrase, and for each run, a random sample of at most 50 of the class labels that match the phrase is selected for evaluation. The samples taken for each run, corresponding to the same phrase, are combined into a merged list. This produces one merged list for each phrase, for a total of 75 merged lists. The precision score over a target

phrase is the precision score over its sample of class labels.

The last two columns of Table 3 capture the precision scores for the class labels. The scores are computed in two ways: averaged over the (variable) subsets of target phrases for which some matching class label(s) exist, in the last but one column, e.g., over 19 of the 75 target phrases for R_{wc}; and averaged over the entire set of 75 target phrases, in the last column. The former does not penalize a run for not being able to extract any class labels containing a particular target phrase, whereas the latter does penalize. Naturally, precision scores over the entire set of target phrases decrease when coverage is lower, for runs R_{wc}, R_{wl} and, to a lesser extent, R_{dc} and R_{ql}. But even after ignoring target phrases with no matching class labels, precision scores in the last but one column in Table 3 reveal important properties of the experimental runs. First, between the two Wikipedia-based runs, R_{wl} has perfect class labels, whereas as many as 1 in 4 class labels of run R_{wc} are marked as incorrect during the evaluation. Second, the class labels collected from *"list of .."* queries in run R_{ql} correspond to relevant, well-formed concepts in 80% of the cases. Third, the generation of class labels via phrase similarities (R_{qg}) greatly increases coverage as shown earlier. The increase comes at the expense of lowering precision from 80% to 72%. However, the phrases from initial queries that are expanded via distributional similarities can be limited from multiple to only one, by switching from R_{qg} to R_{qa}. This gives higher precision for R_{qa} than for R_{qg}.

As a complement to Table 3, the graphs in Figure 1 offer a more detailed view into the precision of class labels. The figure covers a Wikipedia-based run (R_{wc}) and two query-based runs (R_{ql}, R_{qg}). The graphs show the precision scores, over each of the 75 target phrases. Among target phrases for which some matching class labels exist in the respective run, the target phrases with the lowest precision scores are *robotics* (score of 0.15) and *karlsruhe* (0.33), for R_{wc}; *carotid arteries* and *kidney stones*, both with a score of 0.00 because their matching class labels are all incorrect, for R_{dc}; *african population* and *chester arthur*, both with a score of 0.00 because their matching class labels are all incorrect, for R_{ql}; and *arlene martel* (0.00) and *right to vote* (0.25), for R_{qg}.

Run	Target Phrases			Precision of Class Labels Over Target Phrases	
	All	Matched	Cvg	Over Matched	Over All
R_{wc}	75	19	0.253	0.756	0.191
R_{wl}	75	15	0.200	1.000	0.200
R_{dc}	75	35	0.467	0.834	0.389
R_{ql}	75	48	0.640	0.800	0.512
R_{qa}	75	70	0.933	0.868	0.810
R_{qg}	75	73	0.973	0.724	0.705

Table 3: Precision of class labels that match (i.e., whose names contain) each target phrase, computed as an average over (variable) subsets of target phrases for which some matching class label(s) exist, and as an average over the entire set of 75 target phrases

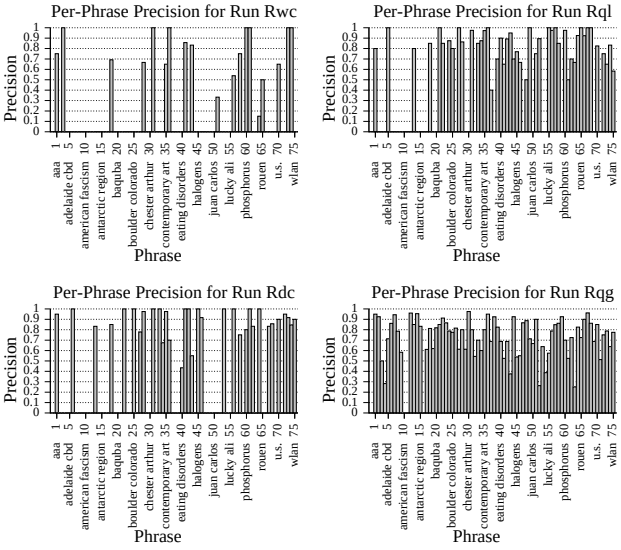


Figure 1: Precision scores for runs R_{wc}, R_{ql}, R_{dc} and R_{qg}, over class labels that match (i.e., contain) each of the 75 target phrases

Precision over Samples of Class Labels: The precision is separately computed over a random sample of 400 class labels per experimental run. The samples are selected from the set of all class labels extracted by the respective run. The precision scores are: 0.759 for R_{wc}; 1.000 for R_{wl}; 0.806 for R_{dc}; 0.811 for R_{ql}; 0.856 for R_{qa}; and 0.711 for R_{qg}. The scores are in line with scores computed earlier over the target phrases, in the fourth column of Table 3.

Discussion: As noted in (Ponzetto and Strube, 2007), Wikipedia organizes its articles and categories into a category network that mixes IsA (subsumption) edges with non-IsA (thematic) edges. Whenever an edge in Wikipedia is not IsA, the par-

Longest Class Labels
R_{wl}: [japanese army and navy members in military or politic services in proper japan korea manchuria occupied china and nearest areas in previous times and pacific war epoch(1930-40s), mental disorders as defined by the diagnostic and statistical manual of mental disorders and the international statistical classification of diseases and related health problems,..]
R_{qg}: [differences between transformational leadership and transactional leadership, things to do in llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch, philosophical differences between thomas jefferson and alexander hamilton, musculoskeletal manifestations of human immunodeficiency virus infection,..]

Table 4: Longest class labels extracted by runs R_{wl} and R_{qg}

ent category may not be a relevant concept that describes some set of instances that share similar properties. Such categories are not good class labels, and therefore are marked as incorrect. Examples include the class labels *"austrian contemporary art"*, *"1999 majorca open"* and *"u.s. route 30"*, listed in Wikipedia as categories of the instances *vienna biennale*, *1999 majorca open* and *squirrel hill tunnel* respectively. This affects the precision scores for R_{wc} in Table 3. It also affects the coverage values relative to R_{wc} in Table 1. Ideally, high-precision experimental runs would not extract any incorrect class labels that happen to appear in R_{wc}, for example *"austrian contemporary art"*. But the coverage relative to R_{wc} would artificially penalize such runs, for not extracting the incorrect class labels from R_{wc}.

As a proxy for estimating class label complexity, Table 4 shows the longest class labels derived from Wikipedia (R_{wl}) vs. generated from queries (R_{qg}).

Class labels derived from Web search queries may be semantically overlapping. Examples are *"writers who killed themselves"* vs. *"writers who committed suicide"*. The overlap is desirable, since different Web users may request the same information via different queries. The same phenomenon has been observed in other information extraction tasks. It also affects manually-created resources like Wikipedia. The continuous manual refinements to Wikipedia content still cannot prevent the occurrence of duplicate class labels among Wikipedia List-Of categories. The duplicates are present in run R_{wl}. Examples are *"formula one drivers that never qualified for a race"* vs. *"formula one drivers who never qualified for a race"*; or *"goaltenders who have scored a goal in a nhl game"* vs. *"goaltenders who have scored a goal in an nhl game"*. Some of the lexical differences among class labels are due to undesirable misspellings. Again, similar problems occasionally affect existing Wikipedia categories: *"nobel laureates who endorse barack obama"* vs. *"nobel laureates who endorse barrack obama"*.

Target Class Labels
007 movie actors, .308 weapons, actors with obsessive compulsive disorder, antibiotics for multiple sclerosis, astronauts in space station, automobiles with remote start, beatles songs of love, beetles that bite, companies with sustainable competitive advantage, countries with double taxation agreements with india, criminals who have been executed, daft punk live albums, dallas medical companies, direct democracy states, electronic companies in electronic city bangalore, expensive brands of shoes, eye diseases in cats, f1 car companies, fwd sports cars, garden landscaping magazines, heliskiing resorts, hell in a cell wrestlers, holidays celebrated in sydney, ibf weight classes, ibiza 2011 djs, immunology scientists, jewelry manufacturing companies, kanye west songs on youtube, kingston upon thames supermarkets, latin military ranks, ludhiana newspapers, maastricht treaty countries, musicians who have been shot, no front license plate states, non-profit organizations in nashville tennessee, organic chocolate companies, plants which are used in homeopathy, programming languages for server side programming, qatar chemical companies, qld private schools, real estate companies in virginia beach virginia, respiratory infection antibiotics, serial killers with antisocial personality disorder, singers with curly hair, telecommunications companies in the philippines, trains from la to san diego, visual basic database management systems, warmblood colors, washington university basketball players, world heritage sites in northern ireland

Table 5: Set of 50 class labels, used in the evaluation of extracted instances

5 Evaluation of Instances

5.1 Evaluation Procedure

Target Set of Class Labels: The target set for evaluation is shown in Table 5. Initially, a random sample of 100 class labels is selected from all class labels in

Tag (Value): Examples of Instances
correct (1.0): countries with double taxation agreements with india: thailand; hell in a cell wrestlers: brock lesnar; ibiza 2011 djs: dimitri from paris; heliskiing resorts: valle nevado
questionable (0.5): 007 movie actors: david niven; kanye west songs on youtube: the good life; holidays celebrated in sydney: waitangi day
incorrect (0.0): electronic companies in electronic city bangalore: bank of baroda; garden landscaping magazines: marquis; immunology scientists: rosalind franklin

Table 6: Correctness tags manually assigned to instances extracted from queries for various class labels

run R_{qg}. Class labels deemed incorrect, as well as class labels for which no instances are extracted, are manually removed from the sample. Out of the remaining class labels, a smaller random sample of 50 of the remaining class labels is retained, for the purpose of evaluating the quality of instances extracted for various class labels.

Evaluation Metric: The evaluation computes the precision of the ranked list of instances extracted for each target class label. To remove any undesirable bias towards higher-ranked instances, the ranked list is sorted alphabetically, then each instance is assigned one of the correctness tags from Table 6. Instances are deemed questionable, if they would be correct for a rather obscure interpretation of the class label. For example, *david niven* is an actor in one of the spoofs rather than main releases of the *007 movie*. Instances that would be correct if a few words were dropped or added are also deemed questionable: *the good life* is not one of the *"kanye west songs on youtube"* but *good life* is.

To compute the precision score over a ranked list of instances, the correctness tags are converted to numeric values. Precision at some rank N in the list is measured as the sum of the correctness values of the instances extracted up to rank N, divided by the number of instances extracted up to rank N.

5.2 Precision of Instances

Precision: Precision scores in Table 7 vary across target class labels. For some class labels, the extracted instances are noisy enough that scores are below 0.50 at ranks 10 and higher. This is the case for *"electronic companies in electronic city bangalore"* and *"daft punk live albums"*, and especially for *"garden landscaping magazines"* which has the worst precision. On the other hand, instances extracted for *"companies with sustainable competitive advantage"* or *"criminals who have been executed"* have high precision across all ranks. As an average over all target class labels, precision is 0.76 at rank 10, and 0.71 at rank 50. Although there is room for improvement, we find these accuracy levels to be encouragingly good, especially at rank 50. As a reminder, instances are extracted from noisy queries, and for class labels as fine-grained as those acquired and used in our experiments. Some of the extracted ranked lists of instances are shown in Table 8.

Target Class Label	Precision of Instances			
	@1	@5	@10	@50
007 movie actors	1.00	1.00	0.85	0.85
actors with obsessive compulsive disorder	0.00	0.60	0.70	0.70
antibiotics for multiple sclerosis	0.50	0.60	0.55	0.58
astronauts in space station	1.00	0.70	0.85	0.83
automobiles with remote start	1.00	1.00	0.75	0.75
beatles songs of love	0.00	0.50	0.65	0.52
beetles that bite	1.00	0.80	0.50	0.56
companies with sustainable competitive advantage	1.00	1.00	0.80	0.88
countries with double taxation agreements with india	1.00	1.00	1.00	0.90
criminals who have been executed	1.00	1.00	0.90	0.82
daft punk live albums	0.50	0.40	0.35	0.35
dallas medical companies	0.00	0.70	0.65	0.54
direct democracy states	1.00	1.00	0.90	0.86
electronic companies in electronic city bangalore	1.00	0.40	0.40	0.42
expensive brands of shoes	1.00	1.00	0.90	0.92
eye diseases in cats	0.50	0.50	0.35	0.35
f1 car companies	1.00	1.00	0.80	0.30
fwd sports cars	1.00	1.00	1.00	1.00
garden landscaping magazines	0.00	0.10	0.15	0.06
heliskiing resorts	1.00	1.00	1.00	1.00
hell in a cell wrestlers	1.00	1.00	1.00	0.92
holidays celebrated in sydney	1.00	0.70	0.75	0.75
...	...	...	...	...
Average over 50 class labels	0.80	0.80	0.76	0.71

Table 7: Precision at various ranks in the ranked lists of instances extracted from queries, for various target class labels and as an average over the entire set of 50 target class labels

Target Class Label	Extracted Instances
countries with double taxation agreements with india	[singapore, malaysia, mauritius, kenya, australia, united kingdom, cyprus, turkey, thailand, germany,..]
direct democracy states	[california, oregon, nevada, wisconsin, louisiana, arizona, vermont, alaska, illinois, michigan,..]
fwd sports cars	[scion tc, ford probe, honda prelude, nissan 200sx, lotus elan, mitsubishi fto, dodge srt-4, mitsubishi gto, volvo c30, toyota celica,..]
garden landscaping magazines	[front, contemporary, gallery, edge, view, chelsea, wallpaper, expo, wizard, sunset,..]
holidays celebrated in sydney	[halloween, australia day, anzac day, independence day, waitangi day, melbourne cup, hogmanay, rotuma day, solstice, yule,..]

Table 8: Ranked lists of instances extracted for a sample of class labels

In additional experiments, the same evaluation procedure is applied to output from two previous extraction methods. The first method starts by internally generating a small set of seed instances for a class label given as input (Wang and Cohen, 2009). A set expansion module then expands the seed set into a longer, ranked list of instances. The instances are extracted from unstructured and semi-structured text within Web documents. The documents are accessed via the search interface of a general-purpose Web search engine (cf. (Wang and Cohen, 2009) for more details). The second method extracts instances of class labels using the extraction patterns proposed in (Hearst, 1992). As such, it is similar to (Kozareva et al., 2008; Van Durme and Paşca, 2008; Wu et al., 2012). The method corresponds to the run R_{dc} described earlier, where the relative ranking of instances and class labels uses the co-occurrence of instances and class labels within queries (Paşca, 2010). For the purpose of the evaluation, when no instances are available for a target class label, the class label is generalized into iteratively shorter phrases containing fewer modifiers, until some instances are available for the shorter phrase. For example, target class labels like *actors with obsessive compulsive disorder*, *beatles songs of love*, *garden landscaping magazines* do not have any instances extracted by the second method. Therefore, the instances evaluated for the second method for these target class labels are collected from the instances of the more general *actors*, *beatles songs*, *landscaping magazines*. Without the generalization, the target class label would receive no credit during the evaluation, and the two previous methods would have lower precision scores. Over the 50 target class labels, the precision of the two methods is 0.11 and 0.27 at rank 5; 0.06 and 0.25 at rank 10; 0.05 and 0.22 at rank 20; and 0.05 and 0.20 at rank 50. The results confirm that, as explained earlier, previous methods for open-domain information extraction have limited ability to extract instances of fine-grained class labels.

Discussion: Earlier errors in the acquisition of the class label affect the usefulness of any instances that may be subsequently extracted for them. The experiments require candidate instances to appear in Wikipedia. This may improve precision, at the expense of not extracting instances that are not yet in Wikipedia (Lin et al., 2012).

6 Related Work

Previous methods for extracting classes of instances from text acquire sets of instances that are each either unlabeled (Pennacchiotti and Pantel, 2009; Jain and Pennacchiotti, 2010; Shi et al., 2010), or associated with a class label (Banko et al., 2007; Wang and Cohen, 2009). The sets of instances and/or class labels may be organized as flat sets or hierarchically, relative to inferred hierarchies (Kozareva and Hovy, 2010) or existing hierarchies such as WordNet (Snow et al., 2006; Davidov and Rappoport, 2009) or the category network within Wikipedia (Wu and Weld, 2008; Ponzetto and Navigli, 2009). Semi-structured text from Web documents is a complementary resource to unstructured text, for the purpose of extracting relations in general (Cafarella et al., 2008), and classes and instances in particular (Talukdar et al., 2008; Dalvi et al., 2012).

With previous methods, the vocabulary of class labels potentially produced for any instance is confined to a closed set provided manually as input (Wang and Cohen, 2009; Carlson et al., 2010). The closed set is often derived from resources like Wikipedia (Talukdar and Pereira, 2010; Lin et al.,

2012; Hoffart et al., 2013) or Freebase (Pantel et al., 2012). Alternatively, the vocabulary is not a closed set, but instead is acquired along with the instances (Pantel and Pennacchiotti, 2006; Snow et al., 2006; Banko et al., 2007; Van Durme and Paşca, 2008; Kozareva and Hovy, 2010). In the latter case, the extracted class labels take the form of head nouns preceded by modifiers. Examples are *"cities"*, *"european cities"* (Etzioni et al., 2005); *"artists"*, *"strong acids"* (Pantel and Pennacchiotti, 2006); *"outdoor activities"*, *"prestigious private schools"* (Van Durme and Paşca, 2008); *"methaterians"*, *"aquatic birds"* (Kozareva and Hovy, 2010). In contrast, the class labels extracted in our method exhibit greater syntactic diversity and are finer-grained. In addition, they are not constrained to a particular set of categories available in resources like Wikipedia.

Fine-grained class labels roughly correspond to queries submitted in typed search (Demartini et al., 2009) or entity search (Balog et al., 2010) or list-seeking questions (*"name the circuit judges in the cayman islands that are british"*). But our focus is on generating, rather than answering such queries or, more generally, attempting to deeply understand their semantics (Li, 2010). Phrase similarities can be derived with any methods, using documents (Lin and Wu, 2009) or search queries (Jain and Pennacchiotti, 2010).

Whether Web search queries are a useful textual data source for open-domain information extraction has been investigated in several tasks. Examples are collecting unlabeled sets of similar instances (Jain and Pennacchiotti, 2010), ranking of class labels already extracted from text (Paşca, 2010), extracting attributes of instances (Alfonseca et al., 2010) and identifying the occurrences in queries of instances of several types, where the types are defined in a manually-created resource (Pantel et al., 2012). Comparatively, we show that queries are useful in identifying possible class labels, not only re-ranking them; and even in populating the class labels with relevant, albeit small, sets of corresponding instances.

As automatically-extracted class labels become finer-grained, they more clearly illustrate a phenomenon that received little attention. Namely, class labels of an instance, on one hand, and relations linking the instance with other instances and classes, on the other hand, are not mutually exclusive pieces of knowledge. Their extraction does not necessarily require different, dedicated techniques. Quite the opposite, class labels serve in text as nothing more than convenient lexical representations, or lexical shorthands, of relations linking instances with other instances. The class labels *"no front license plate states"* and *"states with no front license plate requirement"* are applicable to *arizona*. If so, it is because *arizona* is a *state*, and states require the installation of *license plates* on vehicles, and the requirement does not apply to the *front* of vehicles in the case of *arizona*. The connection between class labels and relations has been judiciously exploited in (Nastase and Strube, 2008). In that study, relations encoded implicitly within Wikipedia categories are transformed into explicit relations. As an example, the explicit relation that *deconstructing harry* is *directed* by *woody allen* is obtained from the fact that *deconstructing harry* is listed under *"movies directed by woody allen"* in Wikipedia. Ours is the first approach to examine the potential for extracting relations from search queries, where relations are compactly and loosely folded into the respective class labels. A variety of methods address the more general task of acquisition of open-domain relations from documents, e.g., (Zhu et al., 2009; Carlson et al., 2010; Fader et al., 2011; Lao et al., 2011).

7 Conclusion

The approach introduced in this paper exploits knowledge loosely encoded within Web search queries. It acquires a vocabulary of class labels that are finer grained than in previous literature. The class labels have precision comparable to that of class labels derived from human-created knowledge repositories. Furthermore, representative instances are extracted from queries for the fine-grained class labels, at encouraging levels of accuracy. Current work explores the use of noisy syntactic features to increase the accuracy of extracted class labels; the extraction of instances from evidence in multiple, rather than single queries; the expansion of extracted instances into larger sets; and the conversion of fine-grained class labels into relations among classes.

References

E. Alfonseca, M. Paşca, and E. Robledo-Arnuncio. 2010. Acquisition of instance attributes via labeled and related instances. In *Proceedings of the 33rd International Conference on Research and Development in Information Retrieval (SIGIR-10)*, pages 58–65, Geneva, Switzerland.

K. Balog, M. Bron, and M. de Rijke. 2010. Category-based query modeling for entity search. In *Proceedings of the 32nd European Conference on Information Retrieval (ECIR-10)*, pages 319–331, Milton Keynes, United Kingdom.

M. Banko, Michael J Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676, Hyderabad, India.

T. Brants. 2000. TnT - a statistical part of speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00)*, pages 224–231, Seattle, Washington.

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. WebTables: Exploring the power of tables on the Web. In *Proceedings of the 34th Conference on Very Large Data Bases (VLDB-08)*, pages 538–549, Auckland, New Zealand.

A. Carlson, J. Betteridge, R. Wang, E. Hruschka, and T. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the 3rd ACM Conference on Web Search and Data Mining (WSDM-10)*, pages 101–110, New York.

B. Dalvi, W. Cohen, and J. Callan. 2012. Websets: Extracting sets of entities from the Web using unsupervised information extraction. In *Proceedings of the 5th ACM Conference on Web Search and Data Mining (WSDM-12)*, pages 243–252, Seattle, Washington.

D. Davidov and A. Rappoport. 2009. Enhancement of lexical concepts using cross-lingual Web mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 852–861, Singapore.

G. Demartini, T. Iofciu, and A. de Vries. 2009. Overview of the INEX 2009 Entity Ranking track. In *INitiative for the Evaluation of XML Retrieval Workshop*, pages 254–264, Brisbane, Australia.

D. Downey, M. Broadhead, and O. Etzioni. 2007. Locating complex named entities in Web text. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2733–2739, Hyderabad, India.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the Web: an experimental study. *Artificial Intelligence*, 165(1):91–134.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 3–10, Barcelona, Spain.

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*, pages 1535–1545, Edinburgh, Scotland.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France.

J. Hoffart, F. Suchanek, K. Berberich, and G. Weikum. 2013. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.

A. Jain and M. Pennacchiotti. 2010. Open entity extraction from Web search query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 510–518, Beijing, China.

Z. Kozareva and E. Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, pages 1110–1118, Cambridge, Massachusetts.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the Web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 1048–1056, Columbus, Ohio.

N. Lao, T. Mitchell, and W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*, pages 529–539, Edinburgh, Scotland.

X. Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1337–1345, Uppsala, Sweden.

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7, Taipei, Taiwan.

D. Lin and X. Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 1030–1038, Singapore.

T. Lin, Mausam, and O. Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-12)*, pages 893–903, Jeju Island, Korea.

V. Nastase and M. Strube. 2008. Decoding Wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1219–1224, Chicago, Illinois.

M. Paşca. 2010. The role of queries in ranking labeled instances extracted from text. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 955–962, Beijing, China.

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 113–120, Sydney, Australia.

P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 938–947, Singapore.

P. Pantel, T. Lin, and M. Gamon. 2012. Mining entity types from query logs via user intent modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-12)*, pages 563–571, Jeju Island, Korea.

M. Pennacchiotti and P. Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 238–247, Singapore.

S. Ponzetto and R. Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 2083–2088, Pasadena, California.

S. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, pages 1440–1447, Vancouver, British Columbia.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

D. Radev, W. Fan, H. Qi, H. Wu, and A. Grewal. 2005. Probabilistic question answering on the Web. *Journal of the American Society for Information Science and Technology*, 56(3):571–583.

M. Remy. 2002. Wikipedia: The free encyclopedia. *Online Information Review*, 26(6):434.

S. Shi, H. Zhang, X. Yuan, and J. Wen. 2010. Corpus-based semantic class mining: Distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 993–1001, Beijing, China.

R. Snow, D. Jurafsky, and A. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 801–808, Sydney, Australia.

P. Talukdar and F. Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1473–1481, Uppsala, Sweden.

P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 582–590, Honolulu, Hawaii.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, pages 1243–1248, Chicago, Illinois.

R. Wang and W. Cohen. 2009. Automatic set instance extraction using the Web. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*, pages 441–449, Singapore.

F. Wu and D. Weld. 2008. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the 17th World Wide Web Conference (WWW-08)*, pages 635–644, Beijing, China.

W. Wu, , H. Li, H. Wang, and K. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the 2012 International Conference on Management of Data (SIGMOD-12)*, pages 481–492, Scottsdale, Arizona.

J. Zhu, Z. Nie, X. Liu, B. Zhang, and J. Wen. 2009. StatSnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th World Wide Web Conference (WWW-09)*, pages 101–110, Madrid, Spain.

Unsupervised Relation Extraction with General Domain Knowledge

Oier Lopez de Lacalle[1,2] and **Mirella Lapata**[1]
[1]Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB
[2]IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
oier.lopezdelacalle@ehu.es, mlap@inf.ed.ac.uk

Abstract

In this paper we present an unsupervised approach to relational information extraction. Our model partitions tuples representing an observed syntactic relationship between two named entities (e.g., "X was born in Y" and "X is from Y") into clusters corresponding to underlying semantic relation types (e.g., *BornIn*, *Located*). Our approach incorporates general domain knowledge which we encode as First Order Logic rules and automatically combine with a topic model developed specifically for the relation extraction task. Evaluation results on the ACE 2007 English Relation Detection and Categorization (RDC) task show that our model outperforms competitive unsupervised approaches by a wide margin and is able to produce clusters shaped by both the data and the rules.

1 Introduction

Information extraction (IE) is becoming increasingly useful as a form of shallow semantic analysis. Learning relational facts from text is one of the core tasks of IE and has applications in a variety of fields including summarization, question answering, and information retrieval. Previous work (Surdeanu and Ciaramita, 2007; Culotta and Sorensen, 2004; Zhou et al., 2007) has traditionally relied on extensive human involvement (e.g., hand-annotated training instances, manual pattern extraction rules, hand-picked seeds). Standard supervised techniques can yield high performance when large amounts of hand-labeled data are available for a fixed inventory of relation types (e.g., *Employment*, *Located*), however, extraction systems do not easily generalize beyond their training domains and often must be re-engineered for each application. Unsupervised approaches offer a promising alternative which could lead to significant resource savings and more portable extraction systems.

It therefore comes as no surprise that latent topic analysis methods have been used for a variety of IE tasks. Yao et al. (2011), for example, propose a series of topic models which perform relation discovery by clustering tuples representing an observed syntactic relationship between two named entities (e.g., "X was born in Y" and "X is from Y"). The clusters correspond to semantic relations whose number or type is not known in advance. Their models depart from standard Latent Dirichlet Allocation (Blei et al., 2003) in that a document consists of relation tuples rather than individual words; moreover, tuples have features each of which is generated independently from a hidden relation (e.g., the words corresponding to the first and second entities, the type and order of the named entities). Since these features are *local*, they cannot capture more global constraints pertaining to the relation extraction task. Such constraints may take the form of restrictions on which tuples should be clustered together or not. For instance, different types of named entities may be indicative of different relations (ORG-LOC entities often express a *Location* relation whereas PER-PER entities express *Business* or *Family* relations) and thus tuples bearing these entities should not be grouped together. Another example are tuples with identical or similar features which intuitively should be clustered together.

In this paper, we propose an unsupervised approach to relation extraction which does not re-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 415–425,
Seattle, Washington, USA, 18-21 October 2013.

quire any relation-specific training data *and* allows to incorporate *global* constraints general expressing domain knowledge. We encode domain knowledge as First Order Logic (FOL) rules and automatically integrate them with a topic model to produce clusters shaped by the data and the constraints at hand. Specifically, we extend the Fold-all (First-Order Logic latent Dirichlet Allocation) framework (Andrzejewski et al., 2011) to the relation extraction task, explain how to incorporate meaningful constraints, and develop a scalable inference technique. In the presence of multiple candidate relation decompositions for a given corpus, domain knowledge can steer the model towards relations which are best aligned with user and task modeling goals. We also argue that a general mechanism for encoding additional modeling assumptions and side information can lessen the need for "custom" relation extraction model variants. Experimental results on the ACE-2007 Relation Detection and Categorization (RDC) dataset show that our model outperforms competitive unsupervised approaches by a wide margin and is able to uncover meaningful relations with only two general rule types.

Our contributions in this work are three-fold: a new model that modifies the Fold-all framework and extends it to the relation extraction task; a new formalization of the logic rules applicable to topic models defined over a rich set of features; and a proposal for mining the logic rules *automatically* from a corpus contrary to Andrzejewski et al. (2011) who employ manually crafted seeds.

2 Related Work

A variety of learning paradigms have been applied to relation extraction. As mentioned earlier, supervised methods have been shown to perform well in this task. The reliance on manual annotation, which is expensive to produce and thus limited in quantity, has provided the impetus for semi-supervised and purely unsupervised approaches. Semi-supervised methods use a small number of seed instances or patterns (per relation) to launch an iterative training process (Riloff and Jones, 1999; Agichtein and Gravano, 2000; Bunescu and Mooney, 2007; Pantel and Pennacchiotti, 2006). The seeds are used to extract a new set of patterns from a large corpus, which are then used to extract more instances, and so on. Unsupervised relation extraction methods are not limited to a predefined set of target relations, but discover all types of relations found in the text. The relations represent clusters over strings of words (Banko et al., 2007; Hasegawa et al., 2004), syntactic patterns between entities (Yao et al., 2011; Shinyama and Sekine, 2006), or logical expressions (Poon and Domingos, 2009). Another learning paradigm is distant supervision which does not require labeled data but instead access to a relational database such as Freebase (Mintz et al., 2009). The idea is to take entities that appear in some relation in the database, find the sentences that express the relation in an unlabeled corpus, and use them to train a relation classifier.

Our own work adds an additional approach into the mix. We use a topic model to infer an arbitrary number of relations between named entities. Although we do not have access to relation-specific information (either as a relational database or manually annotated data), we impose task-specific constraints which inject domain knowledge into the learning algorithm. We thus alleviate known problems with the interpretability of the clusters obtained from topic models and are able to guide our model towards reasonable relations. Andrzejewski et al. (2011) show how to integrate First-Order Logic with vanilla LDA. We extend their formulation to relation tuples rather than individual words. Our model generates a corpus of entity tuples which are in turn represented by features and uses automatically acquired FOL rules. The idea of integrating topic modeling with FOL builds on research in probabilistic logic modeling such as Markov Logic Networks (Richardson and Domingos, 2006). Schoenmackers et al. (2010) learn Horn clauses from web-scale text with aim of finding answers to a user's query. Our work is complementary to theirs. We could make use of their rules to discover more accurate relations.

The general goal of assisting the learner in recovering the "correct" clustering by supplying additional domain knowledge is not new. Gondek and Hofmann (2004) supply a known clustering they do not want the learner to return, whereas Wagstaff et al. (2001) use pairwise labels for items indicating whether they belong in the same cluster. These methods combine domain knowledge with statistical learning in order to improve performance with re-

spect to the true target clustering. Although, the target labels are not available in our case, we are able to show that the inclusion of domain knowledge yields clustering improvements.

3 Learning Setting

Our relation extraction task broadly adheres to the ACE specification guidelines. Our aim is to detect and characterize the semantic relations between two named entities. The input to our model is a corpus of documents, where each document is a bag of relation tuples which can be obtained from the output of any dependency parser. Each tuple represents a syntactic relationship between two named entity (NE) mentions, and consists of three components: the dependency path between the two mentions, the source NE, and the target NE. A dependency path is the concatenation of dependency edges and nodes along a path in the dependency tree. For example, the sentence "***George Bush*** *traveled to* ***France*** *on Thursday for a summit.*" would yield the tuple [SOURCE:*George_Bush*(PER), PATH:*→nsubj→traveled→prep→to→pobj→*, DES:*France*(LOC)]. The tuple here expresses the relation *Located*, however our model does not observe any relation labels during training. The model assigns tuples to clusters, corresponding to an underlying relation type. Each tuple instance can be then labeled with an identifier corresponding to the cluster (aka relation) it has been assigned to.

4 Modeling Framework

Our model builds on the work of Yao et al. (2011) who develop a series of generative probabilistic models for relation extraction. Specifically, we extend their relational LDA model by interfacing it with FOL-rules. In the following, we first describe their approach in more detail and then present our extensions and modifications.

4.1 Relational LDA

Relational LDA is an extension to LDA with a similar generative story. LDA models each document as a mixture of topics, which are in turn characterized as distributions over words. In relational LDA, each document is a mixture of relations over tuples representing syntactic relations between two named entities. The relation tuples are in turn generated a by set of features drawn independently from the underlying relation distribution.

More technically, a multinomial distribution over relations θ_{d_i} is drawn from a Dirichlet prior ($\theta \sim \text{Dir}(\alpha)$) at the document level. Relation tuples are generated from a multinomial distribution θ_{d_i} ($z_i|\theta_{d_i} \sim \text{Mult}(\theta_{d_i})$) and are represented with k features. Each feature is drawn (independently) from a multinomial distribution selected by the relation assigned to tuple i ($f_{ik}|z_i, \phi_{z_i} \sim \text{Mult}(\phi_{z_i})$). Relations are drawn from a Dirichlet prior ($\phi \sim \text{Dir}(\beta)$). In other words, each tuple in a document is assigned a hidden relation ($z = z_1...z_N$); each relation is represented by a multinomial distribution over features ϕ_r (Dirichlet prior β). ϕ_r is a vector with F dimensions each corresponding to a feature. Finally, documents ($j = 1...D$) are associated with a multinomial distribution θ_j over relations (Dirichlet prior α). θ_j is a vector with R dimensions, one for each relation.

Figure 1 represents relational LDA model as a an undirected graphical model or factor graph (Bishop, 2006), ignoring for the moment the factor which connects the $\boldsymbol{d}$, $\boldsymbol{z}$, $f_{1...k}$ and $\boldsymbol{o}$ variables. Directed graphical models can be converted into undirected ones by adding edges between co-parents (Koller and Friedman, 2009). Each clique in the graph defines a potential function which replaces the conditional probabilities in the directed graph. Each maximal clique is associated with a special factor node (the black squares) and clique members are connected to that factor. The probability of any specific configuration is calculated by multiplying the potential functions and normalizing them. We adopt the factor graph representation as is it convenient for introducing logic rules into the model. The joint probability of the model given the priors and the documents ($P(\boldsymbol{p}, \boldsymbol{z}, \phi, \theta|\alpha, \beta, \boldsymbol{d})$) is equivalent to:

$$\prod_{r}^{R} p(\phi_r|\beta) \prod_{j}^{D} p(\theta_j|\alpha) \prod_{i}^{N} \theta_{d_i}(z_i) \prod_{k \in p_i} \phi_{z_i}(f_k) \quad (1)$$

where $\theta_{d_i}(z_i)$ is the z_i-th element in the vector θ_{d_i} and $\phi_{z_i}(f_k)$ is f_k-th feature in the ϕ_{z_i} vector. Variable p_i is the i-th tuple containing k features. The parameters of the latent variables (e.g., ϕ, θ) are typically estimated using an approximate inference algorithm such as Gibbs Sampling (Griffiths and Steyvers, 2004).

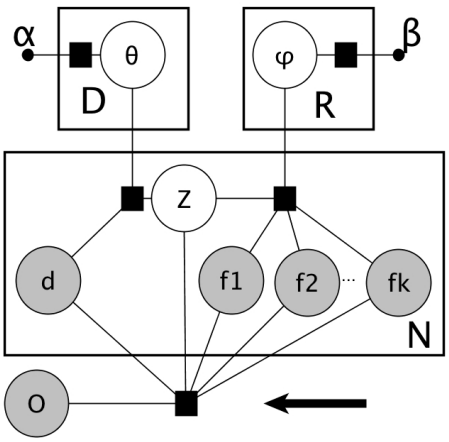


Figure 1: Relational LDA as a factor graph. Filled circles represent observed variables, empty circles are associated with latent variables or model hyperparameters, and plates indicate repeating structures. The black squares are the factor nodes and are associated with the potential functions corresponding to conditional independence among the variables. The model observes D documents ($\boldsymbol{d}$) consisting of N tuples ($\boldsymbol{p}$), each represented by a set of features f_1,$f_2 \ldots f_k$. z represents the relation type assignment to a tuple, θ is the relation type proportion for a given document, and ϕ the relation type distribution over the features. The logic factor (indicated with the arrow) connects the KB with the relational LDA model. Variable $\boldsymbol{o}$ is an observed variable which contains the side information expressed in FOL.

As shown in Figure 1, the observed variables are represented by filled circles. In our case, our model sees the corpus ($\boldsymbol{p}$, $\boldsymbol{d}$), where $\boldsymbol{d}$ is the variable representing the document and the tuples ($\boldsymbol{p}$) are represented by a set of features f_1,$f_2 \ldots f_k$ in the graph. Empty circles are associated with latent variables to be estimated: z represents the relation type assignment to the tuple, θ is the relation type proportion for the given document, and ϕ is the relation type distribution over the features.

The features representing the tuples tap onto semantic information expressed by different surface forms and are an important part of the model. We use a subset of the features proposed in Yao et al. (2011) which we briefly describe below:

SOURCE This feature corresponds to the first entity mention of the tuple. In the sentence ***George Bush*** *traveled to* ***France*** *on Thursday for a summit.*, the value of this feature would be *George Bush*.

Value	Predicate	Description
$z_i = r$	$\mathtt{Z}(i, r)$	Latent relation type
$f_k = v$	$\mathtt{F}(k, v)$	feature of relation tuple
$p_i = i$	$\mathtt{P}(i, f_k)$	tuple i contains feature f_k
$d_i = j$	$\mathtt{D}(i, j)$	observed document

Table 1: Logical variables for Relational LDA. The variable i ranges over tuples in the corpus ($i = [1 \ldots N]$), and k over features in the corpus ($k = [1 \ldots F]$).

DEST The feature corresponds to the second entity mention and its value would be *France* in the previous example.

NEPAIR The feature indicates the type and order of two entity mentions in the tuple. This would be PER-ORG in our example.

PATH This feature refers to the dependency path between two entity mentions. In our sentence, the value of the feature would be PATH:→*nsubj*→*traveled*→*prep*→*to*→*pobj*→.

TRIGGER Finally, trigger features are content words occurring in the dependency path. The path PATH:→*nsubj*→*traveled*→*prep*→*to*→*pobj*→ contains only one trigger word, namely *traveled*. The intuition behind this feature is that paths sharing the same set of trigger words should be grouped in the same cluster.

4.2 First Order Logic and Relational LDA

We next couple relational LDA with global constraints, which we express using FOL rules. We begin by representing relational LDA as a Markov Logic Network (Richardson and Domingos, 2006). We define a logical predicate for each model variable. For example, assigned relation variable ($\mathtt{Z}(i, r)$) is true if $z_i = r$ and false otherwise. Table 1 shows the mapping of model variables onto logical predicates. Logical rules are encoded in the form of a weighted FOL knowledge base (KB) which is then converted into Conjunctive Normal form:

$$KB = \{(\lambda_1, \psi_1), ..., (\lambda_L, \psi_L)\} \quad (2)$$

The KB consists of L pairs, where each ψ_l represents a FOL rule and $\lambda_l \geq 0$ its weight. Rules are soft preferences rather than hard constraints; the weights represent the importance of ψ_l and are

set manually by the domain expert. The KB is tied to the probabilistic model via its *groundings* in the corpus. For each FOL rule ψ_l, let $G(\psi_l)$ be the set of groundings, each mapping the free variables in ψ_l to a specific value. For example, in the rule $\forall i, j, p : \mathtt{F}(i, Obama) \wedge \mathtt{F}(j, WhiteHouse) \wedge \mathtt{P}(p, i) \wedge \mathtt{P}(p, j) \Rightarrow \mathtt{Z}(p, r)$[1], G consists of all the rules where the free variables i, j and p are instantiated. At grounding time, we parse the corpus searching for the tuples that satisfy the logic rules and store the indices of the tuples that ground the rule. The stored indices are used to set ψ_l to a specific value. For the (*Obama*, *White House*) example above, G consists of F propositional rules for each observed feature, where $i \in [1 \dots F]$. For each grounding ($g \in G(\psi_l)$) we define an indicator function:

$$\mathbb{1}_g(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{o}) = \begin{cases} 1, & \text{if } g \text{ is true under} \\ & \boldsymbol{z} \text{ and } \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{o} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $\boldsymbol{z}$ are relation assignments to tuples, $\boldsymbol{p}$ is the set of features in tuples, $\boldsymbol{d}$ are documents, and $\boldsymbol{o}$ the side information encoded in FOL. Contrary to Andrzejewski et al. (2011), we need to ground the rules while taking into account if the feature specified in the rule is expressed by any tuple or the specific given tuple, since we are assigning relations to tuples, and not directly to words.

Next, we define a Markov Random Field (MRF) which combines relational LDA with the FOL knowledge base. The MRF is defined over latent relation tuple assignments $\boldsymbol{z}$, relation feature multinomials ϕ, and relation document multinomials θ (the feature set, document, and external information $\boldsymbol{o}$ are observed). Under this model the conditional probability $P(\boldsymbol{z}, \phi, \theta | \alpha, \beta, \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{o}, KB)$ is proportional to:

$$\exp\left(\sum_{l}^{L} \sum_{g \in G(\psi_l)} \lambda_l \mathbb{1}_g(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{o})\right) \times \prod_{r}^{R} p(\phi_r|\beta) \prod_{j}^{D} p(\theta_j|\alpha) \prod_{i}^{N} \theta_{d_i}(z_i) \prod_{k \in p_i} \phi_{z_i}(f_k) \quad (4)$$

The first term in Equation (4) corresponds to the logic factor in Figure 1 that groups variables $\boldsymbol{d}$, $\boldsymbol{z}$, $f_1, f_2, \dots f_k$ and $\boldsymbol{o}$. The remaining terms in Equation (4) refer to relational LDA. The goal of the model is to estimate the most likely θ and ϕ for the given observed state. As $\boldsymbol{z}$ can not be marginalized out, we proceed with MAP estimation of ($\boldsymbol{z}$, ϕ, θ), maximizing the log of the probability as in Andrzejewski et al. (2011):

$$\arg\max_{\boldsymbol{z}, \phi, \theta} \sum_{l}^{L} \sum_{g \in G(\psi_l)} \lambda_l \mathbb{1}_g(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{o}) + \sum_{r}^{R} \log p(\phi_r|\beta) + \sum_{i}^{N} \log \theta_{d_i}(z_i) \prod_{k \in p_i} \phi_{z_i}(f_k) \quad (5)$$

Once the parameters of the model are estimated (see Section 4.3 for details), we use the ϕ probability distribution to assign a relation to a new test tuple. We select the relation that maximizes the probability $\arg\max_r \prod_i^k P(f_i|\phi_r)$ where $f_1 \dots f_k$ are features representing the tuple and r the relation index.

4.3 Inference

Exact inference is intractable for both relational LDA and MLN models. In order to infer the most likely multinomial parameters ϕ and θ, we applied the Alternating Optimization with Mirror Descent algorithm introduced in Andrzejewski et al. (2011). The algorithm alternates between optimizing the multinomial parameters (ϕ, θ), whilst holding the relation assignments ($\boldsymbol{z}$) fixed, and vice-versa. At each iteration, the algorithm first finds the optimal (ϕ, θ) for a fixed $\boldsymbol{z}$ as the MAP estimate of the Dirichlet posterior:

$$\phi_r(f) \propto n_{rf} + \beta - 1 \quad (6)$$

$$\theta_j(r) \propto n_{jr} + \alpha - 1 \quad (7)$$

where n_{rf} is the number of times feature f is assigned to relation r in relation assignments $\boldsymbol{z}$, and n_{jr} is the number of times relation r is assigned to document j. Next, $\boldsymbol{z}$ is optimized while keeping ϕ and θ fixed. This step is divided into two parts. The algorithm first deals with all z_i which appear only in trivial groundings, i.e., groundings whose indicator functions $\mathbb{1}_g$ are not affected by the latent relation assignment $\boldsymbol{z}$. As $\boldsymbol{z}_i$ only appears in the last term of

[1]This rule translates as "every tuple containing *Obama* and *White House* as features should be in relation cluster r".

Equation (5), the algorithm needs only optimize the following term:

$$z_i = \arg\max_{r=1...R} \theta_{d_i}(r) \prod_{k \in p_i} \phi_{z_i}(f_k) \quad (8)$$

The second part deals with the remaining z_i that appear in non-trivial groundings in the first term of Equation (5). We follow Andrzejewski et al. (2011) in relaxing (5) into a continuous optimization problem and refer the reader to their paper for a more in depth treatment. Suffice it to say that once the binary variables $z_{ir} \in \{0, 1\}$ are relaxed to continuous values $z_{ir} \in [0, 1]$, it is possible to introduce the relational LDA term in the equation and compute the gradient using the Entropic Mirror Descent Algorithm (Beck and Teboulle, 2003):

$$\begin{aligned} &\arg\max_{\boldsymbol{z} \in [0,1]^{|KB|}} \sum_{l}^{L} \sum_{g \in G(\psi_l)} \lambda_l \mathbb{1}_g(\boldsymbol{z}) + \\ &\sum_{i,r} z_{ir} \log \theta_{d_i}(r) \prod_{k \in p_i} \phi_{z_i}(f_k) \quad (9) \\ &\text{s.t } z_{ir} \geq 0 \,, \sum_{i,r} z_{ir} = 1 \end{aligned}$$

In every iteration the approximation algorithm randomly samples a term from the objective function (Equation (9)). The sampled term can be a particular ground rule g or the relational LDA term $(\sum_r z_{ir} \log \theta_{d_i}(r) \prod_{k \in p_i} \phi_{z_i}(f_k))$ for some uniformly sampled index i. The sampling of the terms is weighted according to the rule weight (λ_l) and the grounded value ($G(\psi_l)$) in the case of logic rules, and the size of corpus in tuples ($|\boldsymbol{z}_{KB}|$) for relational LDA. Once we choose term f and take the gradient, we can apply the Entropic Mirror Descent update:

$$z_{ir} \leftarrow \frac{z_{ir} \exp(\eta \nabla_{z_{ir}} f)}{\sum_{r'} z_{ir'} \exp(\eta \nabla_{z_{ir'}} f)} \quad (10)$$

Finally, z_i is recovered by rounding to $\arg\max_r z_{ir}$. The main advantage of this approach is that it requires only a means to sample groundings g for each rule ψ_l, and can avoid fully grounding the FOL rules.

4.4 Logic Rules

Our model assigns relations to tuples rather than topics to words. Since our tuples are described in terms of features our logic rules must reflect this too. For our experiments we defined two very general types of rules described below.

Must-link Tuple The motivation behind this rule is that tuples which share features probably express the same underlying relation. The rule must specify which feature has to be shared for the tuples to be clustered together. For example, the rule below states that tuples containing the dependency path PATH:*→appos→president→prep→of→pobj→* should go in the same cluster:

$$\begin{aligned} \forall i, j, k : \mathtt{F}(i, \text{PATH:}\textit{is\_the\_president\_of}) \wedge \mathtt{P}(j, f_i) \\ \wedge \mathtt{P}(k, f_i) \Rightarrow \neg \mathtt{Z}(j, t) \vee \mathtt{Z}(k, r) \end{aligned}$$

Cannot-link Tuple We also define rules prohibiting tuples to be clustered together because they do not share any features. For example, tuples with ORG-LOC entities, probably express a *Location* relation and should not be clustered together with PER-PER tuples, which in all likelihood express a different relationship (e.g., *Family*). The rule below expresses this constraint:

$$\begin{aligned} \forall i, j, k, l : \mathtt{F}(i, \text{NEPAIR:}\textit{PER-PER}) \\ \wedge \mathtt{F}(j, \text{NEPAIR:}\textit{ORG-LOC}) \\ \wedge \mathtt{P}(k, f_i) \wedge \mathtt{P}(l, f_j) \Rightarrow \neg \mathtt{Z}(k, r) \vee \neg \mathtt{Z}(l, r) \end{aligned}$$

The specification of the first order logic rules is an integral part of the model. The rules express knowledge about the task at hand, the domain involved, and the way the relation extraction problem is modeled (i.e., tuples expressed as features). So far, we have abstractly formulated the rules without explaining how they are specifically instantiated in our model. We could write them down by hand after inspecting some data or through consultation with a domain expert. Instead, we obtain logic rules *automatically* from a corpus following the procedure described in Section 5.

5 Experimental Setup

Data We trained our model on the New York Times (years 2000–2007) corpus created by Yao et al. (2011). The corpus contains approximately 2M entity tuples. The latter were extracted from 428K documents. After post-processing (tokenization, sentence-splitting, and part-of-speech tagging),

Must-link Tuple
$\texttt{F}(i, \text{NEPAIR:}\textit{PER-PER}, \text{TRIGGER:}\textit{wife}) \wedge \texttt{P}(j, f_i) \wedge \texttt{P}(k, f_i) \Rightarrow \neg\texttt{Z}(j, t) \vee \texttt{Z}(k, r)$
$\texttt{F}(i, \text{NEPAIR:}\textit{PER-LOC}, \text{TRIGGER:}\textit{die}) \wedge \texttt{P}(j, f_i) \wedge \texttt{P}(k, f_i) \Rightarrow \neg\texttt{Z}(j, t) \vee \texttt{Z}(k, r)$
$\texttt{F}(i, \text{PATH:}\leftarrow\textit{nsubj}\leftarrow\textit{die}\rightarrow\textit{prep}\rightarrow\textit{in}\rightarrow\textit{pobj}\rightarrow) \wedge \texttt{P}(j, f_i) \wedge \texttt{P}(k, f_i) \Rightarrow \neg\texttt{Z}(j, t) \vee \texttt{Z}(k, r)$
$\texttt{F}(i, \text{SOURCE:}\textit{Kobe}, \text{DEST:}\textit{Lakers}) \wedge \texttt{P}(j, f_i) \wedge \texttt{P}(k, f_i) \Rightarrow \neg\texttt{Z}(j, t) \vee \texttt{Z}(k, r)$
Cannot-link Tuple
$\texttt{F}(i, \text{NEPAIR:}\textit{ORG-LOC}) \wedge \texttt{F}(j, \text{NEPAIR:}\textit{PER-PER}) \wedge \texttt{P}(k, f_i) \wedge \texttt{P}(l, f_j) \Rightarrow \neg\texttt{Z}(k, r) \vee \neg\texttt{Z}(l, r)$
$\texttt{F}(i, \text{NEPAIR:}\textit{LOC-LOC}) \wedge \texttt{F}(j, \text{TRIGGER:}\textit{president}) \wedge \texttt{P}(k, f_i) \wedge \texttt{P}(l, f_j) \Rightarrow \neg\texttt{Z}(k, r) \vee \neg\texttt{Z}(l, r)$
$\texttt{F}(i, \text{NEPAIR:}\textit{PER-LOC}) \wedge \texttt{F}(j, \text{TRIGER:}\textit{member}) \wedge \texttt{P}(k, f_i) \wedge \texttt{P}(l, f_j) \Rightarrow \neg\texttt{Z}(k, r) \vee \neg\texttt{Z}(l, r)$
$\texttt{F}(i, \text{NEPAIR:}\textit{PER-PER}) \wedge \texttt{F}(j, \text{TRIGER:}\textit{sell}) \wedge \texttt{P}(k, f_i) \wedge \texttt{P}(l, f_j) \Rightarrow \neg\texttt{Z}(k, r) \vee \neg\texttt{Z}(l, r)$

Table 2: Examples of automatically extracted Must-link and Cannot-link tuple rules.

named entities were automatically recognized and labeled with PER, ORG, LOC, and MISC (Finkel et al., 2005). Dependency paths for each pair of named entity mentions were extracted from the output of the MaltParser (Nivre et al., 2004). In our experiments, we discarded tuples with paths longer than 10 edges (Lin and Pantel, 2001). We evaluated our model on the test partition of the ACE 2007 (English) RDC dataset which is labeled with gold standard entity mentions and their relations. There are six general relation types and 18 subtypes. We used 25% of the ACE training partition as a development set for parameter tuning.

Logic Rule Extraction We automatically extracted logic rules from the New York Times (NYT) corpus as follows. The intuition behind Must-link rules is that tuples with common features should cluster together. Although we do not know which features would yield the best rules, we naively assume that good features are frequently co-occurring features. Using the log-likelihood ratio (Dunning, 1993), we first discarded low confidence feature co-occurrences ($p < 0.05$). Two features co-occur if they are both found within the same sentence. We then sorted the remaining co-occurrences by their frequency and retained the N-best ones. We only considered unigram and bigram features since higher-order ones tend to be sparse. An example of a bigram feature would be (PATH:←nsubj←grow→prep→in→pobj→, DEST:Chicago).

The main intuition behind Cannot-link rules is that tuples without any common features should not cluster together. So, if two features never co-occur, they probably express different relations. For every unigram and bigram feature in the respective N-best list, we find the features it does not co-occur with in the NYT corpus. For example, NEPAIR:PER–LOC does not co-occur with DEST:*Yankees* and the bigram DEST:*United Nations*, NEPAIR:PER–ORG does not co-occur with SOURCE:Mr. Bush, NEPAIR:PER–LOC. Cannot-link rules are then based on such non-co-occurring feature pairs.

We optimized N empirically on the development set. We experimented with values ranging from 20 to 500. We obtained 20 Must-link rules for coarse-grained relations and 400 rules for their subtypes. We extracted 1,814 Cannot-link rules for general relations ($N = 50$) and 34,522 rules for subtypes ($N = 400$). The number of features involved in the Must-link rules was 25 for coarse-grained relations and 422 for fine-grained relations. For Cannot-link rules, 62 features were involved in coarse-grained relations and 422 in fine-grained relations.

Examples of the rules we extracted are shown in Table 2. The first rule in the upper half of the table states that tuples must cluster together if their source and target entities are PER and contain the trigger word *wife* in their dependency path. The second rule is similar, the source entity here is PER, the target LOC and the trigger word is *die*. According to the third rule, tuples featuring the path PATH:←*nsubj*←*die*→*prep*→*in*→*pobj*→ should be in the same cluster. The fourth rule forces tuples whose source entity is *Kobe* and target entity is *Lakers* to cluster together. The second half of the table illustrates Cannot-link tuple rules. The first rule prevents tuples with ORG-LOC entities to cluster to-

gether with PER-PER tuples. The second rule states that we cannot link LOC-LOC tuples with those whose trigger word is *president*, and so on.

Parameter Tuning Our framework has several parameters that must be adjusted for an optimal clustering solution. These include the hyperparameters α and β as well as the number of clusters. In addition, we have to assign a weight to each FOL rule grounding. An exhaustive search on the hyperparameters and rule weights is not possible. We therefore followed a step-wise approximation procedure. First, we find the best α and β values, whilst varying the number of clusters. Once we have the best hyperparameters for each clustering, we set the weights for the FOL rules. We varied the number of relations from 5 to 50. We experimented with α values in the range of $[0.05 - 0.5]$ and β values in the range of $[0.05 - 0.5]$. These values were optimized separately for coarse- and fine-grained relations. Table 3 shows the optimal number of clusters for different model variants and relation types.

The FOL weights can also make a difference in the final output; the bigger the weight the more times the rule will be sampled in the Mirror Descent algorithm. We experimented with two weighting schemes: (a) we gave a weight of 1 or 0.5 to each rule grounding and (b) we scaled the weights so as to make their contribution comparable to relational LDA. We obtained best results on the development set with the former scheme.

Baselines We compared our FOL relational LDA model against standard LDA (Blei et al., 2003) and relational LDA without the FOL component. In the case of standard LDA, we estimated topics (relations) over words, and used the context of the entity mentions pairs as a bag of words feature to select the most likely cluster at test time. Parameters for LDA and relational LDA were optimized following the same parameter tuning procedure described above.

We also compared our model against the unsupervised method introduced in Hasegawa et al. (2004). Their key idea is to cluster pairs of co-occurring named entities according to the similarity of their surrounding contexts. Following their approach, we measured context similarity using the vector space model and the cosine metric and grouped NE pairs into clusters using a complete linkage hierarchical clustering algorithm. We adopted the same parameter values as detailed in their paper (e.g., cosine similarity threshold, length of context vectors). At test time, instances were assigned to the relation cluster most similar to them (according to the cosine measure).

Evaluation We evaluated the clusters obtained by our model and the comparison systems using the Fscore measure introduced in the SemEval 2007 task (Agirre and Soroa, 2007); it is the harmonic mean of precision and recall defined as the number of correct members of a cluster divided by the number of items in the cluster and the number of items in the gold-standard class, respectively.

6 Results

Our results are summarized in Table 3 which reports Fscore for (Hasegawa et al., 2004), LDA, relational LDA (RelLDA), and our model with the FOL component. To assess the impact of the rules on the clustering, we conducted several rule ablation studies. We thus present results with a model that includes both Must-link and Cannot-link tuple rules (CLT+MLT), and models that include either Must-link (MLT) or Cannot-link (CLT) rules but not both. We show the performance of these models with the entire feature set (see (ALL) in the table) and with a subset consisting solely of NE pair related features (see (NEPAIR) in the table). We report results against coarse- and fine-grained relations (6 and 18 relation types in ACE, respectively). The table shows the optimal number of relation clusters (in parentheses) per model and relation type.

We also wanted to examine the quality of the logic rules. Recall that we learn these heuristically from the NYT corpus. We thus trained an additional variant of our model with rules extracted from the ACE training set (75%) which contains relation annotations. The extraction procedure was similar to the unsupervised case, save that the relation types were known and thus informative features could be mined more reliably. For Must-link rules, we extracted unigram and bigram feature frequencies for each relation type and applied TF-IDF weighting in order to discover the most discriminative ones. We created logic rules for the 10 best feature combinations in each relation type. Regarding Cannot-link rules, we enumerated the features (unigrams and bigrams)

Model	Subtype		Type	
HASEGAWA	26.1	(12)	34.7	(12)
LDA	23.4	(10)	29.0	(5)
RelLDA	30.4	(40)	38.6	(5)
U-MLT (ALL)	36.6	(10)	48.0	(5)
U-CLT (ALL)	30.5	(5)	39.3	(5)
U-CLT+MLT (ALL)	29.8	(5)	42.0	(5)
U-MLT (NEPAIR)	36.5	(10)	47.2	(5)
U-CLT (NEPAIR)	28.8	(50)	40.5	(5)
U-CLT+MLT (NEPAIR)	30.9	(10)	41.5	(5)
S-MLT (ALL)	37.0	(10)	47.0	(5)
S-CLT (ALL)	31.4	(50)	40.9	(5)
S-CLT+MLT (ALL)	32.3	(10)	42.5	(5)
S-MLT (NEPAIR)	37.0	(10)	47.6	(10)
S-CLT (NEPAIR)	31.4	(10)	40.1	(5)
S-CLT+MLT (NEPAIR)	37.1	(10)	46.0	(5)

Table 3: Model performance on the ACE 2007 test set using Fscore. Results are shown for six main relation types and their subtypes (18 in total). (ALL) models contain rules extracted from the entire feature set. For (NEPAIR) models, rules were extracted from NEPAIR-related features only. Prefix U- denotes models that use unsupervised rules; prefix S- highlights models using supervised rules. The optimal number of relations per model is shown in parentheses.

that did not co-occur in any relation type and applied TF-IDF weighting. Again, we created rules for the 10 most discriminative features. We defined rules over the entire feature set (466 Must-link and 26,074 Cannot-link rules) and a subset containing only NE pairs. In Table 3, prefixes S- and U- indicate model variants with supervised and unsupervised rules, respectively.

Our results show that standard LDA is not suitable for relation extraction. The obtained clusters are not informative enough to induce semantic relations, whereas RelLDA yields substantially better Fscores. This is not entirely surprising, given that RelLDA is a relation extraction specific model. Hasegawa et al.'s (2004) model lies somewhere in the middle between LDA and RelLDA. The combination of RelLDA with automatically extracted FOL rules improves over RelLDA across the board (see the U- models in Table 3). MLT rules deliver the largest improvement for both coarse and fine-grained relation types. In general, CLT models perform worse as well as models using both types of rules (MLT+CLT). The inferior performance of the rule combination may be due to the fact that MLT and CLT rules contain conflicting information and to a certain extent cancel each other out. The use of many rules might also negatively impact inference, i.e., discriminative rules are sampled less and cannot influence the model towards a better solution. Restricting the number of features and rules to named entity pairs only incurs a negligible drop in performance. This is good news for scaling purposes, since a small number of rules can greatly speed-up inference. Interestingly, model variants which use supervised FOL rules (see the prefix S- in Table 3) perform on par with unsupervised models. Again, MLT rules perform best in the supervised case, whereas CLT rules marginally improve over RelLDA.

We assessed whether differences in performance are statistically significant ($p < 0.05$) using bootstrap resampling (Noreen, 1989). All models across all relation types are significantly better than LDA and Hasegawa et al. (2004). FOL-based models perform significantly better than RelLDA, with the exception of all CLT models and U-CLT+MLT (ALL). MLT models are significantly better than any other rule-based model, except those that only use NEPAIR features. We also measured whether different models agree on their topic assignments using Cohen's Kappa.[2] RelLDA agrees least with MLT models and most with CLT models (i.e., $\kappa = 0.50$ for U-MLT (ALL) and $\kappa = 0.65$ for U-CLT (ALL)). This suggests that the CLT rules do not affect the output of RelLDA as much as MLT ones. Examples of relation clusters discovered by the U-MLT (ALL) model are shown in Table 4.

A last note on parameter selection. Our experiments explored the parameter space extensively in order to examine any interactions between the induced relations and the logic rules. For most model variants inferring subtype relations, the preferred number of clusters is 10. For coarse-grained relations, the optimal number of clusters is five. Overall, we found that the quality of the output is highly correlated with the quality of the logic rules and that a few *good* rules are more important than the optimal number of clusters. We consider these findings robust enough to apply across domains and datasets.

[2]For all comparison models the number of relation clusters was set to 10.

	SOURCE	PATH	DEST
Employment	Republican	president of	Senate
	Senate	director of	Yankees
	House	professor at	Republican
	Bush	chairman of	Congress
	Democrat	spokesman for	House
	Mr. Bush	executive of	Mets
	Democrats	director at	U. of California
	Republican	analyst at	United Nations

	SOURCE	PATH	DEST
Sports	Yankees	defeat	World Series
	Mets	win	Olympic
	United States	beat	World Cup
	Giants	play	Yankees
	Jets	win	Super Bowl
	Nets	lose	Olympics
	Knicks	sign	Mets
	Rangers	victory over	Giants

Table 4: Clusters discovered by the U-MLT (ALL) model indicating employment- and sports-type relations. For the sake of readability, we do not display the syntactic dependencies between words in a path.

7 Conclusions

In this paper we presented a new model for unsupervised relation extraction which operates over tuples representing a syntactic relationship between two named entities. Our model clusters such tuples into underlying semantic relations (e.g., *Located*, *Family*) by incorporating general domain knowledge which we encode as First Order Logic rules. Specifically, we combine a topic model developed for the relation extraction task with domain relevant rules, and present an algorithm for estimating the parameters of this model. Evaluation results on the ACE 2007 (English) RDC task show that our model outperforms competitive unsupervised approaches by a wide margin and is able to produce clusters shaped by both the data and the rules.

In the future, we would like to explore additional types of rules such as seed rules, which would assign tuples complying with the "seed" information to distinct relations. Aside from devising new rule types, an obvious next step would be to explore different ways of extracting the rule set based on different criteria (e.g., the most general versus most specific rules). Also note that in the current framework rule weights are set manually by the domain expert. An appealing direction would be to learn these automatically e.g., via a procedure that optimizes some clustering objective. Finally, it should be interesting to use some form of distant supervision (Mintz et al., 2009) either as a means of obtaining useful rules or to discard potentially noisy or uninformative rules.

Acknowledgments

We gratefully acknowledge financial support from the Department of Education, Universities and Research of the Basque Government (BFI-2011-442). We also thank Limin Yao and Sebastian Riedel for sharing their corpus with us and the members of the Probabilistic Models reading group at the University of Edinburgh for helpful feedback.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 85–94, San Antonio, Texas.

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 7–12, Prague, Czech Republic.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Ben Recht. 2011. A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1171–1177, Barcelona, Spain.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India.

Amir Beck and Marc Teboulle. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting*

of the Association of Computational Linguistics, pages 576–583, Prague, Czech Republic.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics, Main Volume*, pages 423–429, Barcelona, Spain.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, Ann Arbor, Michigan.

David Gondek and Thomas Hofmann. 2004. Non-redundant data clustering. In *IEEE International Conference on Data Mining*, pages 75–82. IEEE Computer Society.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *PNAS*, 101(1):5228–5235.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 415–422, Barcelona, Spain.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Dekang Lin and Patrick Pantel. 2001. DIRT – discovery of inference rules from text. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francisco, California.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, pages 49–56, Boston, Massachusetts.

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

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Sydney, Australia.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Suntec, Singapore.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1–2):107–136.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 474–479, Stockholm, Sweden.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order Horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098, Cambridge, MA, October. Association for Computational Linguistics.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA.

Mihai Surdeanu and Massimiliano Ciaramita. 2007. Robust information extration with perceptrons. In *Proceedings of the NIST 2007 Automatic Content Extraction Workshop*.

Kiri Wagstaff, Claire Cardie, C Rogers, and S Schrödl. 2001. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, Scotland, UK.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 728–736, Prague, Czech Republic.

Efficient Collective Entity Linking with Stacking

Zhengyan He[†] **Shujie Liu**[‡] **Yang Song**[†] **Mu Li**[‡] **Ming Zhou**[‡] **Houfeng Wang**[†*]
[†] Key Laboratory of Computational Linguistics (Peking University) Ministry of Education,China
[‡] Microsoft Research Asia
hezhengyan.hit@gmail.com {shujliu,muli,mingzhou}@microsoft.com
songyangmagic@gmail.com wanghf@pku.edu.cn

Abstract

Entity disambiguation works by linking ambiguous mentions in text to their corresponding real-world entities in knowledge base. Recent collective disambiguation methods enforce coherence among contextual decisions at the cost of non-trivial inference processes. We propose a fast collective disambiguation approach based on stacking. First, we train a local predictor g_0 with learning to rank as base learner, to generate initial ranking list of candidates. Second, top k candidates of related instances are searched for constructing expressive global coherence features. A global predictor g_1 is trained in the augmented feature space and stacking is employed to tackle the train/test mismatch problem. The proposed method is fast and easy to implement. Experiments show its effectiveness over various algorithms on several public datasets. By learning a rich semantic relatedness measure between entity categories and context document, performance is further improved.

1 Introduction

When extracting knowledge from natural language text into a machine readable format, ambiguous names must be resolved in order to tell which real-world entity the name refers to. The task of linking names to knowledge base is known as entity linking or disambiguation (Ji et al., 2011). The resulting text is populated with semantic rich links to knowledge base like Wikipedia, and ready for various downstream NLP applications.

*Corresponding author

Previous researches have proposed several kinds of effective approaches for this problem. Learning to rank (L2R) approaches use hand-crafted features $f(d, e)$ to describe the similarity or dissimilarity between contextual document d and entity definition e. L2R approaches are very flexible and expressive. Features like name matching, context similarity (Li et al., 2009; Zheng et al., 2010; Lehmann et al., 2010) and category context correlation (Bunescu and Pasca, 2006) can be incorporated with ease. Nevertheless, decisions are made independently and inconsistent results are found from time to time.

Collective approaches utilize dependencies between different decisions and resolve all ambiguous mentions within the same context simultaneously (Han et al., 2011; Hoffart et al., 2011; Kulkarni et al., 2009; Ratinov et al., 2011). Collective approaches can improve performance when local evidence is not confident enough. They often utilize semantic relations across different mentions, and is why they are called $global$ approaches, while L2R methods fall into $local$ approaches (Ratinov et al., 2011). However, collective inference processes are often expensive and involve an exponential search space.

We propose a collective entity linking method based on stacking. Stacked generalization (Wolpert, 1992) is a powerful meta learning algorithm that uses two levels of learners. The predictions of the first learner are taken as augmented features for the second learner. The nice property of stacking is that it does not restrict the form of the base learner. In this paper, our base learner, an L2R ranker, is first employed to generate a ranking list of candidates.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 426–435,
Seattle, Washington, USA, 18-21 October 2013.

At the next level, we search for semantic coherent entities from the top k candidates of neighboring mentions. The second learner is trained on the augmented feature space to enforce semantic coherence. Stacking is employed to handle train/test mismatch problem. Compared with existing collective methods, the inference process of our method is much faster because of the simple form of its base learner.

Wikipedians annotate each entity with categories which provide another source of valuable semantic information. (Bunescu and Pasca, 2006) propose to generalize beyond context-entity correlation $s(d, e)$ with word-category correlation $s(w, c)$. However, this method works at word level, and does not scale well to large number of categories. We explore a representation learning technique to learn the category-context association in latent semantic space, which scales much better to large knowledge base.

Our contributions are as follows: (1) We propose a fast and accurate stacking-based collective entity linking method, which combines the benefits of both coherence modeling of collective approaches and expressivity of L2R methods. We show an effective usage of ranking list as global features, which is a key improvement for the global predictor. (2) To overcome problems of scalability and shallow word-level comparison, we learn the category-context correlation with recent advances of representation learning, and show that this extra semantic information indeed helps improve entity linking performance.

2 Related Work

Most popular entity linking systems use the L2R framework (Bunescu and Pasca, 2006; Li et al., 2009; Zheng et al., 2010; Lehmann et al., 2010). Its discriminative nature gives the model enough flexibility and expressivity. It can include any features that describe the similarity or dissimilarity of context d and candidate entity e. They often perform well even on small training set, with carefully-designed features. This category falls into the $local$ approach as the decision processes for each mention are made independently (Ratinov et al., 2011).

(Cucerzan, 2007) first suggests to optimize an objective function that is similar to the collective approach. However, the author adopts an approximation method because of the large search space (which is $O(n^m)$ for a document with m mentions, each with n candidates). Various other methods like integer linear programming (Kulkarni et al., 2009), personalized PageRank (Han et al., 2011) and greedy graph cutting (Hoffart et al., 2011) have been explored in literature. Our method without stacking resembles the method of (Ratinov et al., 2011) in that they use the predictions of a local ranker to generate features for global ranker. The differences are that we use stacking to train the local ranker to handle the train/test mismatch problem and top k candidates to generate features for the global ranker.

Stacked generalization (Wolpert, 1992) is a meta learning algorithm that uses multiple learners outputs to augment the feature space of subsequent learners. It utilizes a cross-validation strategy to address the train set / testset label mismatch problem. Various applications of stacking in NLP have been proposed, such as collective document classification (Kou and Cohen, 2007), stacked dependency parsing (Martins et al., 2008) and joint Chinese word segmentation and part-of-speech tagging (Sun, 2011). (Kou and Cohen, 2007) propose stacked graphical learning which captures dependencies between data with relational template. Our method is inspired by their approach. The difference is our base learner is an L2R model. We search related entity candidates in a large semantic relatedness graph, based on the assumption that true candidates are often semantically correlated while false ones scattered around.

Wikipedians annotate entries in Wikipedia with category network. This valuable information generalizes entity-context correlation to category-context correlation. (Bunescu and Pasca, 2006) utilize category-word as features in their ranking model. (Kataria et al., 2011) employ a hierarchical topic model where each inner node in the hierarchy is a category. Both approaches must rely on pruned categories because the large number of noisy categories. We try to address this problem with recent advances of representation learning (Bai et al., 2009), which learns the relatedness of category and context in latent continuous space. This method scales well to potentially large knowledge base.

3 Method

In this section, we first introduce our base learner and local features used; next, the stacking training strategy is given, followed by an explanation of our global coherence model with augmented feature space; finally we explain how to learn category-context correlation with representation learning technique.

3.1 Base learner and local predictor g_0

Entity linking is formalized as follows: given an ambiguous name mention m with its contextual document d, a list of candidate entities $e_1, e_2, \ldots, e_{n(m)} \in C(m)$ is generated for m, our predictor g will generate a ranking score $g(e_i)$ for each candidate e_i. The ranking score will be used to construct augmented features for the next level learner, or used by our end system to select the answer:

$$\hat{e} = \arg \max_{e \in C(m)} g(e) \quad (1)$$

In an L2R framework, the model is often defined as a linear combination of features. Here, our features $\vec{f}(d, e)$ are derived from document d and candidate e. The model is defined as $g(e) = \vec{w}\vec{f}(d, e)$. In our problem, we are given a list of training data $\mathcal{D} = \{(d_i, e_i)\}$. We want to optimize the parameter $\vec{w}$, such that the correct entity has a higher score over negative ones. This is done via a preference learning technique SVM^{rank}, first introduced by (Joachims, 2002). The following margin based loss is minimized w.r.t $\vec{w}$:

$$L = \frac{1}{2}\|\vec{w}\|^2 + C \sum \xi_{d,e'} \quad (2)$$

$$\text{s.t.} \quad \vec{w}(\vec{f}(d, e) - \vec{f}(d, e')) \geq 1 - \xi_{d,e'} \quad (3)$$

$$\xi_{d,e'} \geq 0 \quad (4)$$

where C is a trade-off between training error and margin size; ξ is slacking variable and loops over all query documents d and negative candidates $e' \in C(m) - \{e\}$.

This model is expressive enough to include any form of features describing the similarity and dissimilarity of d and e. We only include some typical features seen in literature. The inclusion of these features is not meant to be exhaustive. Our purpose is to build a moderate model in which some of the useful features like string matching and entity popularity cannot be easily expressed by collective approaches like (Hoffart et al., 2011; Han et al., 2011). The features for level 0 predictor g_0 are described in Table 1. The reader can consult (Li et al., 2009; Zheng et al., 2010; Lehmann et al., 2010) for further reference.

Surface matching:
1. mention string m exactly matches candidate e, i.e. $m = e$
2. neither m is a substring of e nor e is a substring of m
3. $m \neq e$ and m is a substring of e
4. $m \neq e$ and e is a substring of m
5. $m \neq e$ and m is a redirect pointing to e in Wikipedia
6. $m \neq e$ and e starts with m
7. $m \neq e$ and e ends with m
Context matching:
1. cosine similarity of TF-IDF score between context and entire Wikipedia page of candidate
2. cosine similarity of TF-IDF score between context and introduction of Wikipedia page
3. jaccard distance between context and entire Wikipedia page of candidate
4. jaccard distance between context and introduction of Wikipedia page
Popularity or prominence feature:
percentage of Wikipedia hyperlinks pointing to e given mention m, i.e. P(e|m)
Category-context coherence model:
cat_0 and cat_1 (details in Section 3.4)

Table 1: Features for local predictor g_0.

3.2 Stacking training for global predictor g_1

Stacked generalization (Wolpert, 1992) is a meta learning algorithm that stacks two "levels" of predictors. Level 0 includes one or more predictors $h_1^{(0)}, h_2^{(0)}, \ldots, h_K^{(0)} : \mathbb{R}^d \to \mathbb{R}$, each one is trained on the original d-dimensional feature space. The level 1 predictor $h^{(1)} : \mathbb{R}^{d+K} \to \mathbb{R}$ is trained in the augmented $(d+K)$-dimensional feature space, in which predictions at level 0 are taken as extra features in $h^{(1)}$.

(Kou and Cohen, 2007) proposed stacked graphi-

cal learning for learning and inference on relational data. In stacked graphical learning, dependencies among data are captured by *relational template*, with which one searches for *related instances* of the current instance. The augmented feature space does not necessarily to be $d + K$. Instead, one can construct any declarative feature with the original data and predictions of related instances. For instance, in collective document classification (Kou and Cohen, 2007) employ relational template to extract documents that link to this document, then apply a COUNT aggregator over each category on neighboring documents as level 1 features.

In our entity linking task, we use a single predictor g_0 trained with local features at level 0. Compared with (Kou and Cohen, 2007), both g_0 and g_1 are L2R models rather than classifier. At level 1, for each document-candidate entity pair, we use the *relational template* $\mathcal{N}(x)$ to find related entities for entity x, and construct global features with some function $\mathcal{G}(\{g_0(n)|n \in \mathcal{N}(x)\})$ (details in Sec. 3.3). The global predictor g_1 receives as input the original features plus $\mathcal{G}$.

One problem is that if we use g_0 trained on the entire training set to predict related instances in training set, the accuracy can be somehow different (typically lower) for future unseen data. g_1 with this prediction as input doesn't generalize well to test data. This is known as train/test mismatch problem. To mimic test time behavior, training is performed in a cross-validation-like way. Let $\mathcal{D}$ be the entire training set:

1. Split $\mathcal{D}$ into L partitions $\{\mathcal{D}_1, \ldots, \mathcal{D}_L\}$
2. For each split $\mathcal{D}_i$:
   - 2.1 Train an instance of g_0 on $\mathcal{D} - \mathcal{D}_i$
   - 2.2 Predict all related instances in $\mathcal{D}_i$ with this predictor g_0
   - 2.3 Augment feature space for $x \in \mathcal{D}_i$, with $\mathcal{G}$ applied on predictions of $\mathcal{N}(x)$
3. Train level 0 predictor g_0 on entire $\mathcal{D}$, for expanding feature space for test data
4. Train level 1 predictor g_1 on entire $\mathcal{D}$, in the augmented feature space.

In the next subsection, we will describe how to construct global features from the predictions of g_0 on neighbors $\mathcal{N}(x)$ with $\mathcal{G}$.

3.3 Enforcing coherence with global features $\mathcal{G}$

If one wants to identify the correct entity for an ambiguous name, he would possibly look for related entities in its surrounding context. However, surrounding entities can also exhibit some degree of ambiguity. In ideal cases, most true candidates are inter-connected with semantic links while negative candidates are scattered around (Fig. 1). Thus, we ask the following question: Is there any highly relevant entity to this candidate in context? Or, is there any mention with highly relevant entity to this candidate in the top k ranking list of this mention? And how many those mentions are? The reason to look up top k candidates is to improve recall. g_0 may not perfectly rank related entity at the first place, e.g. "Mitt Romney" in Figure 1.

Assume the ambiguous mention set is M. For each mention $m_i \in M$, we rank each entity $e_{i,j} \in C(m_i)$ by its score $g_0(e_{i,j})$. Denote its rank as $Rank(e_{i,j})$. For each entity e in the candidate set $E = \{e_{i,j}|\forall e_{i,j} \in C(m_i), \forall m_i \in M\}$, we search related instances for e as follows:

1. search in E for entities with semantic relatedness above a threshold ($\{0.1,0.3,0.5,0.7,0.9\}$);
2. select those entities in step (1) with $Rank(e)$ less than or equal to k ($k \in \{1, 3, 5\}$);
3. map entities in step (2) to unique set of mentions U, excluding current m, i.e. $e \in C(m)$.

This process is relatively fast. It only involves a sparse matrix slicing operation on the large precomputed semantic relatedness matrix in step (1), and logical operation in step (2,3). The following features are fired concerning the unique set U:

- if U is empty;
- if U is not empty;
- if the percentage $|U|/|M|$ is above a threshold (e.g. 0.3).

The above process generates a total of 45 ($5 \times 3 \times 3$) global features.

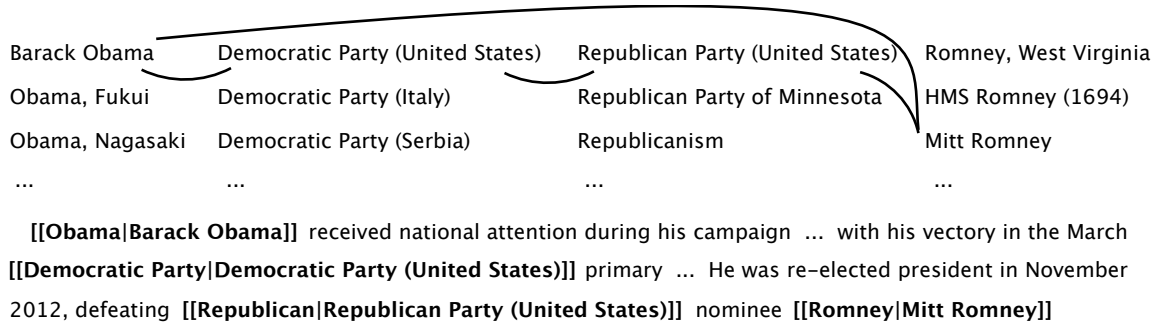


Figure 1: Semantic links for collective entity linking. Annotation $[[mention|entity]]$ follows Wikipedia conventions.

Finally, the semantic relatedness measure of two entities e_i,e_j is defined as the common in-links of e_i and e_j in Wikipedia (Milne and Witten, 2008; Han et al., 2011):

$$SR(e_i, e_j) = 1 - \frac{\log(\max(|A|, |B|)) - \log(|A \cap B|)}{\log(|W|) - \log(\min(|A|, |B|))} \quad (5)$$

where A and B are the set of in-links for entity e_i and e_j respectively, and W is the set of all Wikipedia pages.

Our method is a trade-off between exact collective inference and approximating related instance with top ranked entities produced by g_0. Most collective approaches take all ambiguous mentions into consideration and disambiguate them simultaneously, resulting in difficulty when inference in large search space (Kulkarni et al., 2009; Hoffart et al., 2011). Others resolve to some kinds of approximation. (Cucerzan, 2007) construct features as the average of all candidates for one mention, introducing considerable noise. (Ratinov et al., 2011) also employ a two level architecture but only take top 1 prediction for features. This most resembles our approach, except we use stacking to tackle the train/test mismatch problem, and construct different set of features from top k candidates predicted by g_0. We will show in our experiments that this indeed helps boost performance.

3.4 Learning category-context coherence model cat

Entities in Wikipedia are annotated with rich semantic structures. Category network provides us with another valuable information for entity linking. Take the mention "Romney" as an example, one candidate "Mitt Romney" with category "Republican party presidential nominee" co-occurs frequently with context like "election" and "campaign", while another candidate "Milton Romney" with category "Utah Utes football players" is frequently observed with context like "quarterback" and "backfield". The category network forms a directed acyclic graph (DAG). Some entities can share category through the network, e.g. "Barack Obama" with category "Democratic Party presidential nominees" shares the category "United States presidential candidates by party" with "Mitt Romney" when travelling two levels up the network.

(Bunescu and Pasca, 2006) propose to learn the category-context correlation at word level through category-word pair features. This method creates sparsity problem and does not scale well because the number of features grows linearly with both the number of categories and the vocabulary size. Moreover, the category network is somewhat noisy, e.g. travelling up four levels of the hierarchy can result in over ten thousand categories, with many irrelevant ones.

Rather than learning the correlation at word level, we explore a representation learning method that learns category-context correlation in the latent semantic space. Supervised Semantic Indexing (SSI) (Bai et al., 2009) is trained on query-document pairs to predict their degree of matching. The comparison is performed in the latent semantic space, so that synonymy and polysemy are implicitly handled by its inner mechanism. The score function between query q and document d is defined as:

$$f(q, d) = q^T W d \quad (6)$$

where W is learned with supervision like click-through data.

Given training data $\{(q_i, d_i)\}$, training is done by randomly sampling a negative target d^-. The model optimizes W such that $f(q, d^+) > f(q, d^-)$. Thus, the training objective is to minimize the following margin-based loss function:

$$\sum_{q,d^+,d^-} \max(0, 1 - f(q, d^+) + f(q, d^-)) \quad (7)$$

which is also known as contrastive estimation (Smith and Eisner, 2005).

W can become very large and inefficient when we have a big vocabulary size. This is addressed by replacing W with its low rank approximation:

$$W = U^T V + I \quad (8)$$

here, the identity term I is a trade-off between the latent space model and a vector space model. The gradient step is performed with Stochastic Gradient Descent (SGD):

$$U \leftarrow U + \lambda V(d^+ - d^-)q^T, \quad \texttt{if } 1 - f(q, d^+) + f(q, d^-) > 0 \quad (9)$$

$$V \leftarrow V + \lambda U q(d^+ - d^-)^T, \quad \texttt{if } 1 - f(q, d^+) + f(q, d^-) > 0. \quad (10)$$

where λ is the learning rate.

The query and document are not necessary real query and document. In our case, we treat our problem as: given the occurring context of an entity, retrieving categories corresponding to this entity. Thus, we use context as query q and the categories of this candidate entity as d. We also treat the definition page of an entity as its context, and first train the model with definition pages, because definition pages exhibit more focused topic. This considerably accelerates the training process. To reduce noise, We input the categories directly connected with one entity as a word vector. The input can be a TF-IDF vector or binary vector. We denote model trained with normalized TF-IDF and with binary input as cat_0 and cat_1 respectively.

4 Experiments

4.1 Datasets

Previous researches have used diverse datasets for evaluation, which makes it hard for comparison with others' approaches. TAC-KBP has several years of data for evaluating entity linking system, but is not well suited for evaluating collective approaches. Recently, (Hoffart et al., 2011) annotated a clean and much larger dataset AIDA [1] for collective approaches evaluation based on CoNLL 2003 NER dataset. (Ratinov et al., 2011) also refined previous work and contribute four publicly available datasets [2]. Thanks to their great works, we have enough data to evaluate against. According to the setting of (Hoffart et al., 2011), we split the AIDA dataset for train/development/test with 946/216/231 documents. We train a separate model on the Wikipedia training set for evaluating ACE/QUAINT/WIKI dataset (Ratinov et al., 2011). Table 2 gives a brief overview of the datasets used.

For knowledge base, we use the Wikipedia XML dump [3] to extract over 3.3 million entities. We use annotation from Wikipedia to build a name dictionary from mention string m to entity e for candidate generation, including redirects, disambiguation pages and hyperlinks, follows the approach of (Cucerzan, 2007). For candidate generation, we keep the top 30 candidates by popularity (Tbl. 1). Note that our name dictionary is different from (Ratinov et al., 2011) and has a much higher recall. Since (Ratinov et al., 2011) evaluate on "solvable" mentions and we have no way to recover those mentions, we re-implement their global features and the final scores are not directly comparable to theirs.

4.2 Methods under comparison

We compare our algorithm with several state-of-the-art collective entity disambiguation systems. The AIDA system proposed by (Hoffart et al., 2011) use a greedy graph cutting algorithm that iteratively remove entities with low confidence scores. (Han et al., 2011) employ personalized PageRank to propagate evidence between different decisions. Both algorithms use simple local features without discriminative training. (Kulkarni et al., 2009) propose to use integer linear programming (ILP) for inference. Except our re-implementation of Han's

[1] available at http://www.mpi-inf.mpg.de/yago-naga/aida/

[2] http://cogcomp.cs.illinois.edu/Data, we don't find the MSNBC dataset in the zip file.

[3] available at http://dumps.wikimedia.org/enwiki/, we use the 20110405 xml dump.

Dataset	ndocs	non-NIL	identified	solvable
AIDA dev	216	4791	4791	4707
AIDA test	231	4485	4485	4411
ACE	36	257	238	209(185)
AQUAINT	50	727	697	668(588)
Wikipedia	40	928	918	854(843)

Table 2: Number of mentions in each dataset. "identified" means the mention exists in our name dictionary and "solvable" means the true entity are among the top 30 candidates by popularity. Number in parenthesis shows the results of (Ratinov et al., 2011).

method, both AIDA and ILP solution are quite slow at running time. The online demo of AIDA takes over 10 sec to process one document with moderate size, while the ILP solution takes around 2-3 sec/doc. In contrast, our method takes only 0.3 sec/doc, and is easy to implement.

(Ratinov et al., 2011) also utilize a two layer learner architecture. The difference is that their method use top 1 candidate generated by local learner for global feature generation , while we search the top k candidates. Moreover, stacking is used to tackle the train/test mismatch problem in our model. We re-implement the global features of (Ratinov et al., 2011) and use our local predictor g_0 for level 0 predictor. Note that we only implement their global features concerning common inlinks and inter-connection (totally 9 features) for fair comparison because all other models don't use common outgoing links for global coherence.

4.3 Settings

We implement SVM^{rank} with an adaptation of linear SVM in scikit-learn (which is a wrapper of Liblinear). The category-context coherence model is implemented with Numpy configured with OpenBlas library, and we train this model on the entire Wikipedia hyperlink annotation. It takes about 1.5d for one pass over the entire dataset. The learning rate λ is set to 1e-4 and training cost before update is below 0.02.

Parameter tuning: there aren't many parameters to tune for both g_0 and g_1. The context document window size is fixed as 100 for compatibility with (Ratinov et al., 2011; Hoffart et al., 2011). The number of candidates is fixed to top 30 ranked by entity's popularity. Increase this value will generally boost recall at the cost of lower precision.

We introduce the following default parameter for global features in g_1. The number of fold for stacking is set to $\{1,5,10\}$ (see Table 4, default is 10; 1 means no stacking, i.e. training g_0 with all training data and generating level 1 features for training data directly with this g_0). The number k for searching neighboring entities with relational template is set to $\{1,3,5,7\}$ (e.g. in step 2 of Section 3.3 $k = 5$; default is 5).

For category-context modeling, the vocabulary sizes of context and category are set to top 10k and 6k unigrams by frequency. The latent dimension of low rank approximation is set to 200.

Performance measures: For all non-NIL queries, we evaluate performance with micro precision averaged over queries and macro precision averaged over documents. Mean Reciprocal Rank (MRR) is an information retrieval measure and is defined as $\frac{1}{|Q|}\sum_i^{|Q|}\frac{1}{rank_i}$, where $rank_i$ is the rank of correct answer in response to query i. For ACE/AQUAINT/WIKI we also give the accuracy of "solvable" mentions, but this is not directly comparable to (Ratinov et al., 2011). Our name dictionary is different from theirs and ours has a higher recall rate (Tbl. 2). Hence, the "solvable" set is different.

k	recall	k	recall
1	78.56	6	96.31
2	89.59	7	97.04
3	93.01	8	97.37
4	94.97	9	97.62
5	95.78	10	97.81

Table 3: Top k recall for local predictor g_0.

4.4 Discussions

Table 4 shows the evaluation results on AIDA dataset and Table 5 shows results on datasets ACE/AQUAINT/WIKI.

Effect of cat: The first group in Table 4 shows some baseline features for comparison. We can see even if the categories only carry incomplete and noisy information about an entity, it performs much

Methods	Devset			Testset		
	micro p@1	macro p@1	MRR	micro p@1	macro p@1	MRR
cosine	33.25	28.61	46.03	33.33	28.63	46.54
jaccard	44.71	36.56	57.76	45.66	36.89	57.08
cat_0	54.75	47.14	67.70	61.52	54.72	72.55
cat_1	60.15	54.64	72.98	65.46	61.04	76.84
popularity	69.21	67.59	79.26	69.07	72.63	79.45
g_0	76.04	73.63	84.21	76.16	78.17	84.58
g_0+global(Ratinov)	81.30	78.03	88.14	81.45	81.89	88.70
g_1+1fold	**82.01**	78.52	88.90	**83.59**	**83.58**	**90.05**
g_1+5fold	81.99	78.42	88.87	83.52	83.37	89.99
g_1+10fold	**82.01**	**78.53**	**88.91**	**83.59**	83.55	90.03
g_1+top1	81.65	**78.76**	88.51	81.81	82.55	89.06
g_1+top3	**82.20**	78.64	**88.98**	83.52	83.34	89.94
g_1+top5	82.01	78.57	88.90	83.63	**83.76**	90.05
g_1+top7	82.05	78.40	88.90	**83.75**	83.58	**90.08**
g_0+cat	79.36	76.14	86.66	79.64	80.47	87.32
g_1+cat	82.24	78.49	89.02	84.88	84.49	90.65
g_1+cat+all context	**82.99**	**78.56**	**89.51**	**86.49**	**85.11**	**91.55**
(Hoffart et al., 2011)	-	-	-	**82.29**	82.02	-
(Shirakawa et al., 2011)	-	-	-	81.40	**83.57**	-
(Kulkarni et al., 2009)	-	-	-	72.87	76.74	-
(Han et al., 2011)	-	-	-	78.97	75.77	-

Table 4: Performance on AIDA dataset. Maximal value in each group are highlighted with bold font. top k means up to k candidates are used for searching related instances with relational template.

better than word level features. Group 5 in Table 4 shows cat information generally boosts performance for both predictor g_0 and g_1.

Effect of stacking: Group 3 in Table 4 shows the results with different fold in stacking training. 1 fold means training g_0 with all training data and directly augment training data with this g_0. Surprisingly, we do not observe any substantial difference with various fold size. We deduce it is possible the way we fire global features with top k candidates that alleviates the problem of train/test mismatch when extending feature space for g_1. Despite the ranking of true entity can be lower in testset than in training set, the semantic coherence information can still be captured with searching over top k candidates.

Effect of top k global features: Group 4 in Table 4 shows the effect of k on g_1 performance. Clearly, increasing k generally improves precision and one possible reason is the improvement in recall when searching for related instances. Table 3 shows the top k recall of local predictor g_0. Further increasing k does not show any improvement.

Our method benefits from such a searching strategy, and consistently outperforms the global features of (Ratinov et al., 2011). While their method is a trade-off between expensive exact search over all mentions and greedy assigning all mentions with local predictor, we show this idea can be further extended, somewhat like increasing the beam search size without additional computational overhead. The only exception is the ACE dataset, since this dataset is so small, the difference translates to only one mention. One may notice the improvement on ACE/AQUAINT datasets is a little inconsistent. These datasets are much smaller and the results only differ within 4 mentions. Because these models are

Method	micro p@1	macro p@1	MRR	correct / solvable
ACE				
g_0	77.43	81.30	79.03	95.22
Ratinov	77.43	80.70	78.81	95.22
g_1+5fold	77.04	79.85	78.96	94.74
g_0+cat	**77.82**	**81.48**	**79.31**	**95.69**
g_1+cat	77.43	80.16	79.25	95.22
AQUAINT				
g_0	84.46	84.69	87.49	91.92
Ratinov	85.14	85.29	87.90	92.66
g_1+5fold	**85.83**	**85.55**	**88.27**	**93.41**
g_0+cat	85.01	85.00	87.89	92.51
g_1+cat	85.28	85.14	88.23	92.81
Wikipedia test				
g_0	83.19	84.30	86.63	90.40
Ratinov	84.48	85.96	87.62	91.80
g_1+5fold	84.81	86.29	88.13	92.15
g_0+cat	84.38	86.13	87.51	91.69
g_1+cat	**85.45**	**87.16**	**88.31**	**92.86**

Table 5: Evaluation on ACE/AQUAINT/WIKI datasets.

trained on Wikipedia, the annotation style can be quite different.

Finally, as we analyze the development set of AIDA, we discover that some location entities rely on more distant information across the context, as we increase the context to the entire contextual document, we can gain extra performance boost.

4.5 Error analysis

As we analyze the development set of AIDA, we find some general problems with location names. Location name generally is not part of the main topic of one document. Thus, comparing context with its definition is not realistic. Most of the time, we can find some related location names in context; but other times, it is not easily distinguished. For instance, in "France beats Turkey in men's football..." France refers to "France national football team" but our system links it to the country page "France" because it is more popular. This can be addressed by modeling finer context (Sen, 2012) or local syntactic pattern (Hoffart et al., 2011). In other cases, our system misclassifies "New York City" for "New York" and "Netherlands" for "Holland" and "People's Republic of China" for "China", because in all these cases, the latter ones are the most popular in Wikipedia. It is even hard for us humans to tell the difference based only on context or global coherence.

5 Conclusions

We propose a stacking based collective entity linking method, which stacks a global predictor on top of a local predictor to collect coherence information from neighboring decisions. It is fast and easy to implement. Our method trades off between inefficient exact search and greedily assigning mention with local predictor. It can be seen as searching related entities with relational template in stacked graphical learning, with beam size k. Furthermore, we adopt recent progress in representation learning to learn category-context coherence model. It scales better than existing approaches on large knowledge base and performs comparison in the latent semantic space. Combining these two techniques, our model consistently outperforms all existing more sophisticated collective approaches in our experiments.

Acknowledgments

This research was partly supported by Major National Social Science Fund of China(No. 12&ZD227),National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101) and National Natural Science Foundation of China (No.91024009).

References

B. Bai, J. Weston, D. Grangier, R. Collobert, O. Chapelle, and K. Weinberger. 2009. Supervised semantic indexing. In *The 18th ACM Conference on Information and Knowledge Management (CIKM)*.

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, volume 6, pages 9–16.

S. Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, volume 6, pages 708–716.

X. Han, L. Sun, and J. Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Pro-*

ceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pages 765–774. ACM.

J. Hoffart, M.A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2011. Overview of the tac 2011 knowledge base population track. In *Proceedings of the Fourth Text Analysis Conference*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

S.S. Kataria, K.S. Kumar, R. Rastogi, P. Sen, and S.H. Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *Proceedings of KDD*.

Zhenzhen Kou and William W Cohen. 2007. Stacked graphical models for efficient inference in markov random fields. In *SDM*.

S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM.

J. Lehmann, S. Monahan, L. Nezda, A. Jung, and Y. Shi. 2010. Lcc approaches to knowledge base population at tac 2010. In *Proc. TAC 2010 Workshop*.

F. Li, Z. Zheng, F. Bu, Y. Tang, X. Zhu, and M. Huang. 2009. Thu quanta at tac 2009 kbp and rte track. In *Proceedings of Test Analysis Conference 2009 (TAC 09)*.

André FT Martins, Dipanjan Das, Noah A Smith, and Eric P Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 157–166. Association for Computational Linguistics.

D. Milne and I.H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

P. Sen. 2012. Collective context-aware topic models for entity disambiguation. In *Proceedings of the 21st international conference on World Wide Web*, pages 729–738. ACM.

M. Shirakawa, H. Wang, Y. Song, Z. Wang, K. Nakayama, T. Hara, and S. Nishio. 2011. Entity disambiguation based on a probabilistic taxonomy. Technical report, Technical Report MSR-TR-2011-125, Microsoft Research.

N.A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Weiwei Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *ACL*, pages 1385–1394.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 483–491, Los Angeles, California, June. Association for Computational Linguistics.

Joint Bootstrapping of Corpus Annotations and Entity Types

Hrushikesh Mohapatra Siddhanth Jain Soumen Chakrabarti*
IIT Bombay

Abstract

Web search can be enhanced in powerful ways if token spans in Web text are annotated with disambiguated entities from large catalogs like Freebase. Entity annotators need to be trained on sample mention snippets. Wikipedia entities and annotated pages offer high-quality labeled data for training and evaluation. Unfortunately, Wikipedia features only one-ninth the number of entities as Freebase, and these are a highly biased sample of well-connected, frequently mentioned "head" entities. To bring hope to "tail" entities, we broaden our goal to a second task: assigning types to entities in Freebase but not Wikipedia. The two tasks are synergistic: knowing the types of unfamiliar entities helps disambiguate mentions, and words in mention contexts help assign types to entities. We present TMI, a bipartite graphical model for joint type-mention inference. TMI attempts no schema integration or entity resolution, but exploits the above-mentioned synergy. In experiments involving 780,000 people in Wikipedia, 2.3 million people in Freebase, 700 million Web pages, and over 20 professional editors, TMI shows considerable annotation accuracy improvement (e.g., 70%) compared to baselines (e.g., 46%), especially for "tail" and emerging entities. We also compare with Google's recent annotations of the same corpus with Freebase entities, and report considerable improvements within the people domain.

1 Introduction

Thanks to automatic information extraction and semantic Web efforts, keyword search over unstructured Web text is rapidly evolving toward entity- and type-oriented queries (Guo et al., 2009; Pantel et al., 2012) over semi-structured databases such as Wikipedia, Freebase, and other forms of Linked Data.

A key enabling component for such enhanced search capability is a type and entity *catalog*. This includes a directed acyclic graph of types under the *subTypeOf* relation between types, and entities attached to one or more types via *instanceOf* edges. YAGO (Suchanek et al., 2007) provides such a catalog by unifying Wikipedia and WordNet, followed by some cleanup.

Another enabling component is an *annotated corpus* in which token spans (e.g., the word "Albert") are identified as a mention of an entity (e.g., the Physicist Einstein). Equipped with suitable indices, a catalog and an annotated corpus let us find "scientists who played some musical instrument", and answer many other powerful classes of queries (Li et al., 2010; Sawant and Chakrabarti, 2013).

Consequently, accurate corpus annotation has been intensely investigated (Mihalcea and Csomai, 2007; Cucerzan, 2007; Milne and Witten, 2008; Kulkarni et al., 2009; Han et al., 2011; Ratinov et al., 2011; Hoffart et al., 2011). With two exceptions (Zheng et al., 2012; Gabrilovich et al., 2013) that we discuss later, public-domain corpus annotation work has almost exclusively used Wikipedia and derivatives, partly because Wikipedia provides not only a standardized space of entities, but also reliably labeled mention text within its own documents, which can be used to train machine learning algorithms for entity disambiguation.

However, the high quality of Wikipedia comes at the cost of low entity coverage (4.2 million) and bias toward often-mentioned, richly-connected "head" entities. Hereafter, Wikipedia entities are called W. Freebase has fewer editorial controls, but has at least nine times as many entities. This is particularly perceptible for *people* entities: one needs to be relatively famous to be featured on Wikipedia, but Freebase is less selective. Hereafter, Freebase entities are called F.

As in any heavy-tailed distribution, even relatively obscure entities from $F \setminus W$ are *collectively* mentioned a great many times on the Web, and including them in Web annotation is critical, if entity-oriented search is to impact the vast number of tail

*soumen@cse.iitb.ac.in

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 436–446,
Seattle, Washington, USA, 18-21 October 2013.

queries submitted to Web search engines.

Primary goal — corpus annotation: We have thus established a pressing need to bootstrap from a small entity catalog W (such as Wikipedia entities), and a small *reference* corpus C_W (e.g., Wikipedia text) reliably annotated with entities from W, to a much larger catalog F (e.g., Freebase), and an open-domain large *payload* corpus C (e.g., the Web).

We can and will use entities in $F \cap W$[1] in the bootstrapping process, but the real challenge is to annotate C with mentions m of entities in $F \setminus W$. Unlike for $F \cap W$, we have no training mentions for $F \setminus W$. Therefore, the main disambiguation signal is from the immediate entity neighborhood $N(e)$ of the candidate entity e in the Freebase graph. I.e., if m also reliably mentions some entity in $N(e)$, then e becomes a stronger candidate. Unfortunately, for many "tail" entities $e \in F \setminus W$, $N(e)$ is sparse. Is there hope for annotating the Web with tail entities? Here, we achieve enhanced accuracy for the primary annotation goal by extending it with a related secondary goal.

Secondary goal — entity typing: If we had available a suitable type catalog $\mathcal{T}$ with associated entities in W, which in turn have known textual mentions, we can build models of contexts referring to types like chemists, sports people and politicans. When faced with people called John Williams in $F \setminus W$, we may first try to associate them with types. This can then help disambiguate mentions to specific instances of John Williams in $F \setminus W$. In principle, useful information may also flow in the reverse direction: words in mention contexts may help assign types to entities in $F \setminus W$. For reasons to be made clear, we choose YAGO (Suchanek et al., 2007) as the type catalog $\mathcal{T}$ accompanying entities in W.

Our contributions: We present TMI, a bootstrapping system for solving the two tasks jointly. Apart from matches between the context of m and entity names in $N(e)$, TMI combines and balances evidence from two other sources to decide if e is mentioned at token span m, and has type t:

- a language model for the context in which entities of type t are usually mentioned
- correlations between t and certain path features generated from $N(e)$.

TMI uses a novel probabilistic graphical model formulation to integrate these signals. We give a detailed account of our design of node and edge potentials, and a natural reject option (recall/precision tradeoff).

We report on extensive experiments using YAGO types, Wikipedia entities and text, Freebase entities, and text from ClueWeb12[2], a 700-million-page Web corpus. We focus on all people entities in Wikipedia and Freebase, and provide three kinds of evaluation. First, we evaluate TMI on over 1100 entities in $F \cap W$ and 5500 snippets from Wikipedia text, where it visibly improves upon baselines and a recently proposed alternative method (Zheng et al., 2012). Second, we resort to extensive manual evaluation of annotation on ClueWeb12 Web text with Freebase entities, by professional editors at a commercial search company. TMI again clearly outperforms strong baselines, doing particularly well for nascent or tail entities. TMI improves per-snippet accuracy, for some classes of entities, from 46% to 70%, and pooled F1 score from 66% to 73%. Third, we compare TMI annotations with Google's FACC1 (Gabrilovich et al., 2013) annotations restricted to people; TMI is significantly better. Our annotations and related data can be downloaded from `http://www.cse.iitb.ac.in/~soumen/doc/CSAW/`. To our knowledge, this is among the first reports on extensive human evaluation of machine annotation for $F \setminus W$ on a large Web corpus.

2 Related work

The vast majority of entity annotation work (Mihalcea and Csomai, 2007; Cucerzan, 2007; Milne and Witten, 2008; Kulkarni et al., 2009; Han et al., 2011; Ratinov et al., 2011; Hoffart et al., 2011) use Wikipedia or derivative knowledge bases. (Ritter et al., 2011) and (Zheng et al., 2012) are notable exceptions. (Ritter et al., 2011) use entity names for distant supervision in POS tagging, chunking and broad named entity typing in short tweets, which are different from our goals.

Recently, others have investigated inferring types

[1] With F=Freebase and W=Wikipedia, $F \cap W \approx W$ but not quite; $W \setminus F$ is small but non-empty.

[2] `http://lemurproject.org/clueweb12/`

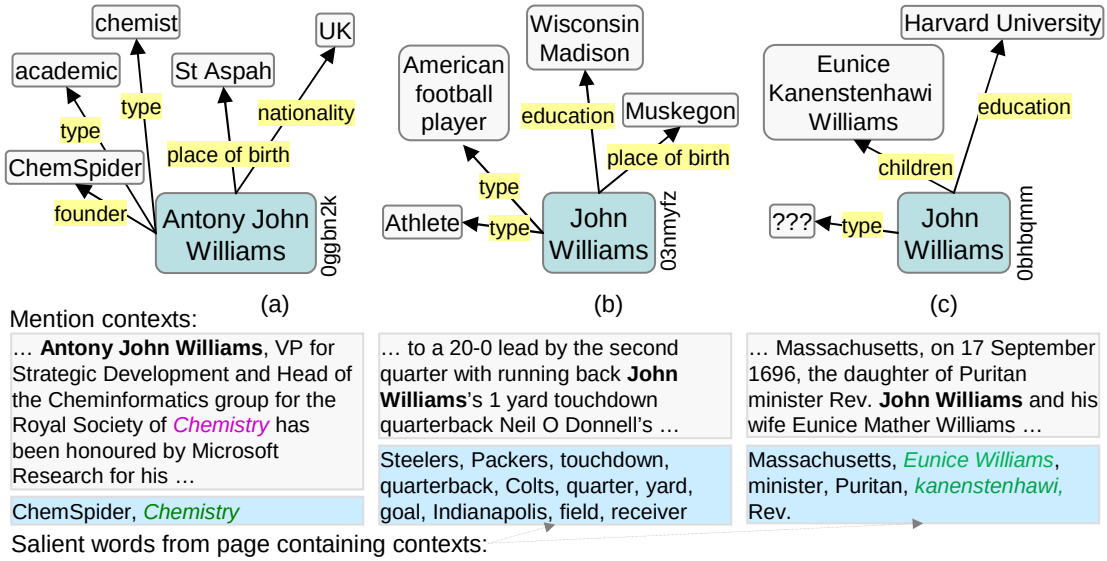


Fig. 1: Signal synergies. Three of the many people mentioned as "John Williams" on the Web are shown, with Freebase MIDs. (a) Easy case where Freebase neighbors of 0ggbn2k match snippet and salient text, and type links are also available. (b) No match exists between Freebase neighborhood and snippet, but type links help attach the snippet to 03nmvfz. (c) Freebase provides no types, but we can provide types from YAGO based on snippet and salient text, which also match neighbors of 0bhbqmm.

of emerging entities (related to our secondary goal). In concurrent work, (Nakashole et al., 2013) propose integer linear program formulations for inferring types of emerging entities from the way their mentions are embedded in curated relation-revealing phrases. (Lin et al., 2012) earlier approached the problem using weight propagation in a bipartite graph connecting unknown to known entities via textual relation patterns. Both note that this can boost mention disambiguation accuracy.

The closest work to ours is by (Zheng et al., 2012): they use semisupervised learning to annotate a corpus with Freebase entities. Like (Milne and Witten, 2008), they depend on unambiguous entity-mention pairs to bootstrap a classifier, then apply it to unlabeled, ambiguous mentions, creating more training data. They use a per-type language model like us (§3.3), but this is used as a secondary cause for (word) feature generation, supplementing and smoothing entity-specific language models. In contrast, we use a rigorous graphical model to combine new signals, not depending on naturally unambiguous mentions. Finally, in the interest of fully automated evaluation, they limit their experiments to $F \cap W$ and Wikipedia corpus, thus differing critically from our human evaluation on F and a Web corpus.

(Gabrilovich et al., 2013) have recently released FACC1: annotations of ClueWeb09 and ClueWeb12 with Freebase entities. Their algorithm is not yet public. They report: "Due to the sheer size of the data, it was not possible to verify all the automatic annotations manually. Based on a small-scale human evaluation, the precision ... is believed to be around 80–85%. Estimating the recall is of course difficult; however, it is believed to be around 70-85%." In §5, we will see that, for people entities, TMI greatly increases recall beyond FACC1, keeping precision unimpaired.

3 The three signals

Fig. 1 shows three Freebase entities mentioned as "John Williams" in Web text, represented as nodes with Freebase "MID"s e = 0ggbn2k, 03nmvfz, and 0bhbqmm, embedded in their Freebase graph neighborhoods. Owing to larger size and higher flux, Freebase shows less editorial uniformity than Wikipedia. This shows up in missing or non-standard relation edges. Unlike YAGO, where each entity is attached to one or more types, e = 0bhbqmm does not have a type link. Many people have a link labeled *profession*, which is a second

kind of type link. Entities like e = 0bhbqmm also have small, uninformative graph neighborhoods.

Also shown are three mention contexts, each represented by the snippet immediately surrounding the mention, and salient words from the documents containing each snippet. (Salient words may be extracted as words that contribute the largest components to the document represented as a TFIDF vector.) (a) shows a favorable but relatively rare case where e = 0ggbn2k has reliable type links. We cannot assume there will be a 1-to-1 correspondence between Freebase and YAGO types, but in §3.2 we will describe how to learn associations between Freebase paths around entities and their YAGO types. The snippet and salient words show reasonable overlap with $N(e)$. In §3.4 we will describe features that characterize such overlap. In (b), e = 03nmvfz is reliably typed, but there is no direct match between $N(e)$ and the snippet. Nevertheless, the snippet can be reliably annotated with 03nmvfz if we can learn associations between types *American football player* and *Athlete* (or their approximate YAGO target types) and several context/salient words (see §3.3). In (c), e = 0bhbqmm is not reliably typed. However, there are matches between $N(e)$ and context/salient words. Once the snippet-to-entity association is established, it is easier to assign 0bhbqmm to suitable types in YAGO.

In this section we will first describe our design of the target type space, and then the tree signals that will be used in our joint inference.

3.1 Designing the target type space

By typing two people called John Williams as actor and footballer, we may also disambiguate their mentions accurately. Therefore, we need a well-organized type space where the types

- collectively cover most entities of interest,
- offer reasonable type prediction accuracy, and
- can be selected algorithmically, for any domain.

Wikipedia and Freebase have many obscure types like "people born in 1937" or "artists from Ontario" which satisfy none of the above requirements. YAGO, on the other hand, has a clean hierarchy of over 200,000 types with broad coverage and fine differentiation. Most entities in $F \setminus W$ can be accurately attached to one or more YAGO types.

if t has $< N_{\text{low}} = 5000$ member entities **then**
  reject t from our type space
  return
if t has $> N_{\text{high}} = 25000$ member entities **then**
  for each immediate child $t' \subset t$ **do**
    call $\mathbf{ChooseTypes}(t')$
else
  accept t into our type space (but do not recurse)

Fig. 2: Procedure $\mathbf{ChooseTypes}(t)$.

YAGO lists around 37,000 subtypes of *person*. To satisfy the three requirements above, we called $\mathbf{ChooseTypes}(\textit{person})$ (Fig. 2); this resulted in 130 suitable types being selected. These directly covered 80% of Freebase people; the rest could mostly be attached to slightly over-generic types within our selection.

3.2 Predicting types from entity neighborhood

There will generally not be a simple mapping between Freebase and target types. E.g., entity e may be known as a *Mayor* in Freebase, but the closest YAGO type may be *Politician*. Edge and node labels from the Freebase graph neighborhood $N(e)$ can embed many clues for assigning e a target type. E.g., e = 03nmvfz may have an edge labeled *playedFor* to a node representing the Wikipedia entity `Pittsburgh_Steelers`, which has a type link to *NFL team*. This two-hop link label sequence would repeat for a large number of players, and can be used as a feature in a classifier.

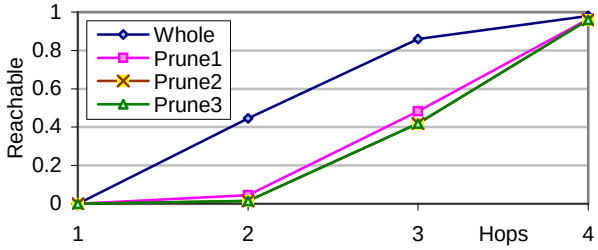


Fig. 3: Freebase has small diameter despite graph thinning. Prune1 removes paths from the whole Freebase graph that pass through nodes */user/root* and */common/topic*, Prune2 also removes node */people/person*, and Prune3 removes several other high degree hubs.

Two further refinements are needed to make this work. First, we have to collect path labels around negative instances we well, and submit positive and negative path labels to a binary classifier to can-

cel the effect of frequent but non-informative path types. Second, indiscriminate expansion around e is infeasible because the Freebase graph has very small diameter. Even after substantial pruning, paths of length 3 and 4 reach over 40% and 96% of all nodes (Fig. 3). This increases computational burden and floods us with noisy and spurious paths. We remedy the problem using an idea from PathRank (Lao and Cohen, 2010). Instead of trying to explore *all* paths originating (or terminating) at e, where e may or may not belong to a target type, we focus on paths *between* e and other known members of the target type.

3.3 Type "language model"

To exploit the second signal, shown in Fig. 1(b), we need to model the association between target YAGO types and the mention contexts of Wikipedia entities known to belong to those types. This model component is in the same spirit as (Zheng et al., 2012).

For each target YAGO type t, we sample positive entities $e \in F \cap W$, and for each e, we collect, from Wikipedia annotated text, a corpus of snippets mentioning e. We remove the mention words and retain the rest. We also collect salient words from the entire Wikipedia document containing the snippet, as shown in Fig. 1.

At this point each target type is associated with a "corpus" of contexts, each represented by snippet words. We compute the IDF of all words in this corpus[3], and then represent each type as a TFIDF vector (Salton and McGill, 1983). A test context is turned into a similar vector, and its score with respect to t is the cosine between these two vectors. This simple approach was found superior to building a more traditional smoothed multinomial unigram model (Zhai, 2008) for each type. Given the output of this component feeds into an outer discriminative inference mechanism, a strict probabilistic model is not necessary.

3.4 Entity neighborhood match with snippet

The third signal is a staple of any disambiguation work: match the occurrence context against the neighborhood in the structured representation. In word sense disambiguation (WSD), support for assigning a word in context to a synset comes from matches between, say, other words in the context and the WordNet neighborhood of the proposed synset. As in WSD, many approaches to Wikification measure some local consistency between a mention m and the neighborhood $N(e)$ of a candidate entity e. $N(e)$ is again limited by a maximum path length ℓ. From snippet m we extract all phrases $P(m)$ excluding the mention words. For each phrase $p \in P(m)$, if p occurs *at least once*[4] in any node of $N(e)$, then we accumulate a credit of $|p| \sum_{w \in p} \text{IDF}(w)$, where w ranges over words in p, $\text{IDF}(w)$ is its inverse document frequency (Salton and McGill, 1983) in Wikipedia, and $|p|$ is the length of the phrase. This rewards exact phrase matches.

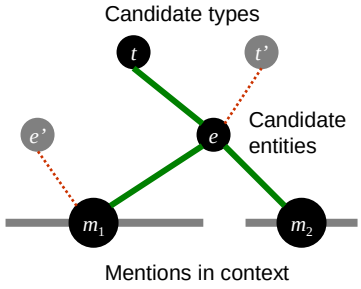


Fig. 4: Tripartite assignment problem.

4 Unified model

Figure 4 abstracts out the signals shown in Figure 1 into a tripartite assignment problem. Each mention like m_1 has to choose at most one entity from among candidate aliasing entities like e and e'. Each entity $e \in F \setminus W$ has to choose one type (for simplicity we ignore zero or more than one types as possibilities) from candidates like t and t'.

The configuration of thick (green) edges should be preferred to alternative dotted (red) edges under these considerations:

- There is high local affinity or compatibility between e and t, based on associations between t and $N(e)$ as discussed in §3.2.
- There are better textual matches between $N(e)$ and m_1, as compared to $N(e')$ and m_1.
- In aggregate, the non-mention tokens in the context of m_1, m_2 (shown as gray horizontal lines) match well the language model associated with mentions of entities of type t (rather than t').

[3] Generic IDF from Wiki text does not work.

[4] Incorporating term frequency often polluted the score.

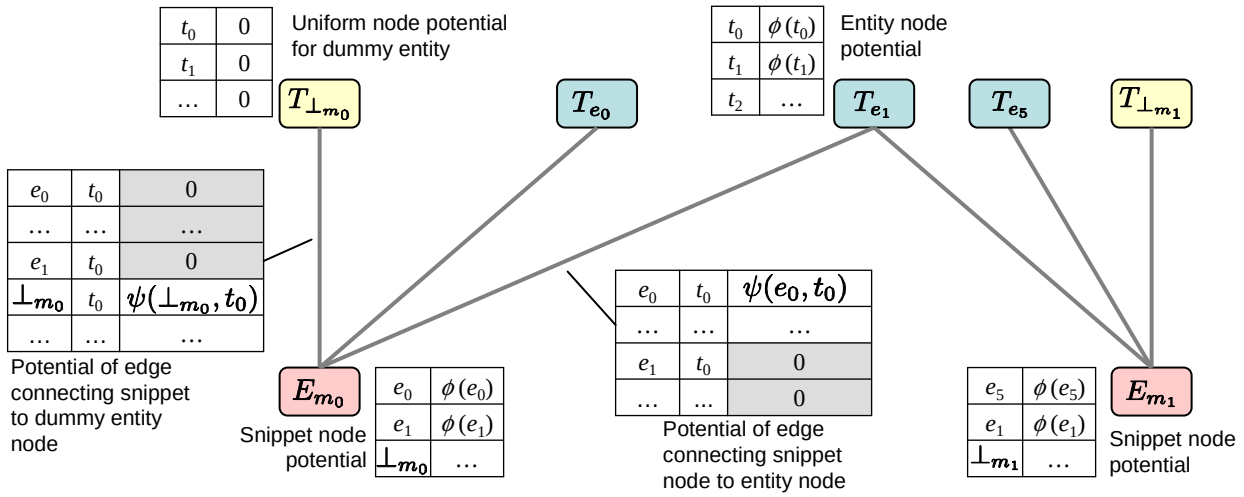


Fig. 5: Illustration of the proposed bipartite graphical model, with tables for node and edge potentials and synthetic $\perp$-entity nodes to implement the reject option.

We now present a unified model that combines the signals and solves the two proposed tasks jointly. We model two kinds of decision variables, which will be represented by nodes in a graphical model. Associated with each entity $e \in F \setminus W$ there is a hidden type variable (node) T_e, which can take on a value from (some subset of) the type catalog $\mathcal{T}$. Associated with each mention m (along with its snippet context and all its observable features) there is a hidden entity variable E_m, which can take on values from some subset of entities. (For simplicity, we assume that entities in $F \cap W$ have already been annotated in the corpus, and no m of interest mentions such entities.)

We will model the probability of a joint assignment of values to T_e, E_m as

$$\log \Pr(\vec{t}, \vec{e}) = \alpha \sum_{e} \phi_e(t_e) + \beta \sum_{m} \phi_m(e_m) + \sum_{e,m} \psi_{e,m}(t_e, e_m) - \text{const.}, \quad (1)$$

where node log potentials are called ϕ_e, ϕ_m, edge log potentials are called $\psi_{e,m}$, and α, β are tuned constants. The log partition function, written "const." above, will not be of interest in inference, where we will seek to choose $\vec{t}, \vec{e}$ to maximize $\Pr(\vec{t}, \vec{e})$. In this section, we will design the node and edge log potentials.

4.1 Node log potentials

Each node T_e is associated with a node potential table, mapping from possible types in $\mathcal{T}_e$ to nonnegative potentials. The potential values are supplied from §3.2 as the classifier output scores.

Each node E_m is associated with a node potential table, mapping from possible entities in $\mathcal{E}_m$ to nonnegative potentials. The potential values are supplied from §3.4.

4.2 Edge log potentials

Suppose we assign $E_m = e$, and $T_e = t$. Then we would like the non-mention context words of m to be highly compatible with the type "language model" developed in §3.3.

If e is among the set of values $\mathcal{E}_m$ that E_m can take, then nodes T_e and E_m are joined by an edge. This edge is associated with an edge potential table $\psi_{e,m} : \mathcal{T}_e \times \mathcal{E}_m \to \mathbb{R}^+$. $\psi_{e,m}(\cdot, e')$ will be set to zero (cells shaded gray in Fig. 5) when $e \neq e'$. $\psi_{e,m}(t, e)$ is set to the cosine match score described in §3.3.

4.3 The reject (a.k.a. null, nil, NA, $\perp$) option

An algorithm may reject many snippets, i.e., refrain from annotating them. This could be because the snippet mentions an entity outside F (and outside W), or the system wishes to ensure high precision at some cost to recall.

Rejection is modeled by adding, for each snippet m, a pseudo or "null" entity $\perp_m$ (also called "no an-

notation" NA, null or nil in the TAC-KBP[5] community). For simplicity, we assume that $\perp_m$ and $\perp_{m'}$ are incomparable or distinct for $m \neq m'$. I.e., we do not offer to cluster mentions of unknown entities. These remain separate, unconnected nodes in the (augmented) Freebase graph.

If we choose $E_m = \perp_m$, we get zero credit for matching non-mention text in m to $N(\perp_m)$, because $N(\perp_m) = \varnothing$ and §3.4 has no information to contribute. I.e., we set $\phi_m(\perp_m) = 0$. In general, $\phi_m(\cdot) \geq 0$, so $\perp_m$ gets the lowest possible credit (but this will be modified shortly).

There is also a type variable $T_{\perp_m}$. To what type should we assign "entity" $\perp_m$? Because $\perp_m$ has no connections in the Freebase graph, no hint can come from §3.2. Put differently, $\phi_{\perp_m}(t)$ will be constant (say, zero) for all snippets m and types t.

Even if we do not know the entity mentioned in m, the non-mention text in m will have differential affinity to different types, obtained from §3.3. This means that, if $E_m = \perp_m$ is chosen, $T_{\perp_m}$ will be $\arg\max_t \psi_{\perp_m,m}(t, \perp_m)$, which explains the non-mention words in m using the best available language model associated with some type. For a different entity $E_m = e_0$ and type $T_{e_0} = t$ assignment to win, $\alpha\phi_m(e_0)+\beta\phi_{e_0}(t)+\psi_{e_0,m}(t, e_0)$ must exceed the null score above. This provides a usable recall/precision handle: we modify $\phi_m(\perp_m)$ to a tuned number; making it smaller generally gives higher recall and lower precision.

4.4 Inference and training

The goal of collective inference will be to assign a type value to each T_e and an entity value to each E_m. We seek the maximum a-posteriori (MAP) label, for which we use tree-reweighted message passing (TRW-S) (Kolmogorov, 2006). Our graph has plenty of bipartite cycles, so inference is approximate. Given the sparsity of data, we preferred to *delexicalize* our objective (1), i.e., avoid word-level features and pre-aggregate their signals via time-tested aggregators (such as TFIDF cosine). As a result we have only two free parameters α, β in (1), which we tune via grid search. A more principled training regimen is left for future work.

[5] http://www.nist.gov/tac/2013/KBP/

5 Experiments

We focus our experiments on one broad type of entities, *people*, that is more challenging for disambiguators than typical showcase examples of distinguishing (Steve) Jobs from employment and Apple Inc. from fruit.

We report on three sets of experiments. In §5.1, we restrict to entities from $F \cap W$ and Wikipedia text, for which ground truth annotation is available. In §5.2, we evaluate TMI and baselines on ClueWeb12 and entities from Freebase, not limited to Wikipedia. In §5.3, we compare TMI with Google's recently published FACC1 annotations (Gabrilovich et al., 2013).

5.1 Reference corpus C_W with $F \cap W$ entities

Limited to people, $|F| = 2323792$, $|W| = 807381$, $|F \setminus W| = 1544942$, and $|F \cap W| = 778850$. It is easiest to evaluate TMI and others on Wikipedia entities. They have known YAGO types. Wikipedia text has explicit (disambiguated) entity annotations. For these reasons, the few known systems for Freebase-based annotation (Zheng et al., 2012) are evaluated exclusively on $F \cap W$.

5.1.1 Seeding and setup

People in $F \cap W$ are known by one or more mention words/phrases. From these, we collect mention phrases along with the candidate entity set for each phrase. The number of candidates is the phrase's *degree of ambiguity*. We sort phrases by ambiguity and draw a sample over the ambiguity range. This gives us seed phrases with representative ambiguity. Then we collect all entities mentioned by these phrases. Overall we collect about 1100 entities and 5500 distinct mentions.

Contrast this with (Zheng et al., 2012), who sample entities from much fewer than 130, and largely well-separated types: professional athletes, academics, actors, films, books, hotels, and tourist attractions. If there were only two namesakes, an actor and a politician, the politician disappears, leaving a naturally unambiguous alias. I.e., (Zheng et al., 2012) did not "complete" their entity sets with aliased entities. For all these reasons, Z0 numbers here are not directly comparable to those in their paper.

5.1.2 Tasks and baselines

The structure of TMI suggests two natural baselines to compare against it. TMI solves two tasks simultaneously: assign types to entities and entities to snippets. So the first baseline, **T0**, is one that solves the typing task separately, and the second, **A0**, does snippet annotation separately. A third baseline, **Z0**, from (Zheng et al., 2012) does only snippet annotation; they do not consider typing entities.

While evaluating types output by TMI and T0 against ground truth, we may wish to assign partial credit for overlapping types, e.g., *athlete* vs. *soccer player*, because our types form an incomplete hierarchy. We use the standard "M&W" score of semantic similarity between types (Milne and Witten, 2008) for this.

As regards snippet annotation, Z0 (Zheng et al., 2012) does not specify any mechanism for handling $\perp$. Therefore we run two sets of experiments. In one we eliminate all snippets with ground truth $\perp$. A0, and TMI are also debarred from returning $\perp$ for any snippet. In the other, snippets marked $\perp$ in ground truth are included. A0 and TMI are enabled to return $\perp$, but Z0 cannot.

	TMI	A0	Z0
0/1 snippet accuracy	**0.827**	0.699	0.627

Fig. 6: Snippet annotation on C_W corpus, $F \cap W$ entities, $\perp$ not allowed.

	TMI	A0	Z0
0/1 snippet accuracy	**0.7307**	0.651	0.622
Snippet precision	**0.858**	0.843	0.622
Snippet recall	**0.777**	0.692	0.639
Snippet F1	**0.815**	0.760	0.630

Fig. 7: Snippet annotation on C_W corpus, $F \cap W$ entities, $\perp$ allowed.

5.1.3 Snippet annotation results

Fig. 6 shows snippet annotation accuracy (fraction of snippets labeled with the correct entity) when $\perp$ is not allowed as an entity. As two uninformed refernces, uniform random choice gives an accuracy of 0.423 and choosing the entity with the largest prior gives an accuracy of 0.767. TMI is considerably better than A0, which is better than Z0 and the uninformed references. This is despite training Z0's per-type topic models not only on unambiguous snippets, but also on a disjoint fraction of $F \cap W$, as a surrogate for Wikipedia's containment in Freebase.

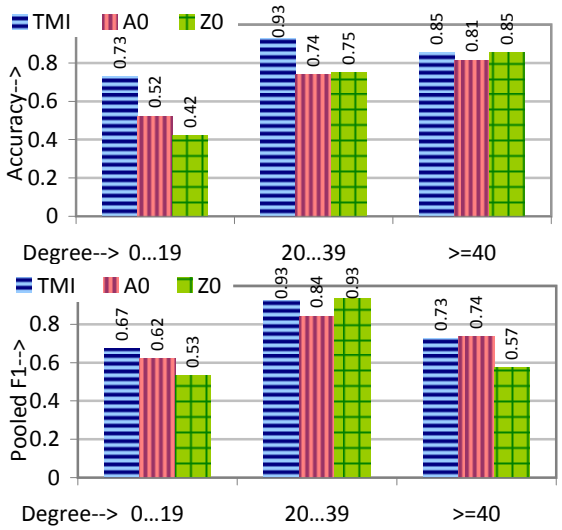


Fig. 8: Bucketed comparison between TMI and baselines, $F \cap W$, $\perp$ allowed.

	TMI	T0
0/1 type accuracy	0.80	0.81
M&W type accuracy	0.82	0.83

Fig. 9: Type inference, C_W corpus, $F \cap W$ entities.

Fig. 7 repeats the experiment while allowing $\perp$. Here, in 0/1 accuracy, $\perp$ is regarded as just another entity. Again, we see that TMI has a clear advantage. Z0's performance here is worse than in (Zheng et al., 2012). This is explained by our much larger and difficult-to-separate type system.

We disaggregate the summary results into buckets, shown in Fig. 8. Each bucket covers a range of degrees of entity nodes in Freebase, while roughly balacing the number of snippets in each bucket. TMI generally shows larger gains for low-degree buckets.

5.1.4 Type prediction results

We also compared the type inference accuracy of TMI and T0; (Zheng et al., 2012) do not infer types. The summary is in Fig. 9. Two uninformed baselines are worth mentioning. Uniform random choice over 130 types gave only 2% accuracy. Chossing the type with largest prior probability gave 28.2% accuracy. TMI is much better, but offers no significant benefit (or degradation) compared to T0. We verified, partly by way of debugging, that there do exist entities e with small degree but a modest number of assigned snippets, for which snippet-to-$N(e)$ matches

angus mcdonald, chris robinson, christopher henry, elizabeth cameron, george woods , henry barnes, jack scott, jeremy robert, john sherman, leonard thomas, marc anthony, mitchell donald, morrison mark, parker edward, richard andrew , simon scott, stephen ross, stuart baron, tom clark, whitney john, austin scott, barbara johnson, brian peterson, carlos rivero, david berman, david johns, donald fraser, george davies, george fisher, graham smith, john pepper, jonathan edwards, kevin brown, kevin hughes, matt johnson, michael davidson, nancy johnson, paul holmes, pedro martins, peter frank, peter mitchell, peter mullen, robert stern, roger edwards, stuart walker, terry evans, tony angelo, tony ward, william jarvis, william sampson

Fig. 10: Seed mentions for confusion clusters for Web corpus C and entities in F.

provide a boost to type prediction accuracy (about 50%), as compared to T0 (about 20% for these instances). Therefore, the flow of information between type and entity assignments is, in principle, bidirectional, in the regime of such entities.

5.2 Payload corpus C with entities in F

Recall our main goal is to annotate payload corpus C with entities in all of F. Experience with $F \cap W$ and C_W may not be representative of F and C. Entities in $F \setminus W$ may not come with reference mentions, and their type and entity neighborhoods may be sparse. Furthermore, compared to the closed world of $F \cap W$, evaluating TMI and baselines over C and $F \setminus W$ is challenging. Entities in $F \setminus W$ do not have ground truth types in the type catalog T (here, YAGO), nor snippets labeled by humans as mentioning them. Therefore, we need human editorial judgment, which is scarce. Even though TMI can be *applied* at Web scale, the scale if *evaluation* is limited by editorial input.

5.2.1 Seeding and data setup

There are about 2.3 million Freebase entities connected to */people/person* via *type* links. Similar to §5.1.1, we chose phrases (Fig. 10) with diverse degree of ambiguity (Fig. 11), to seed confusion clusters. Then we completed the clusters by including aliased entities, as before, so as not to artifically reduce the degree of ambiguity. Note that entities in W can and do contend with entities in $F \setminus W$. The cluster size distribution is shown in Fig. 12. Limited by editorial budget, we finished with 634 entities, 238 distinct aliases and 4,500 snippets.

We used the 700-million-page ClueWeb12 Web corpus. All phrases in the expanded clusters are

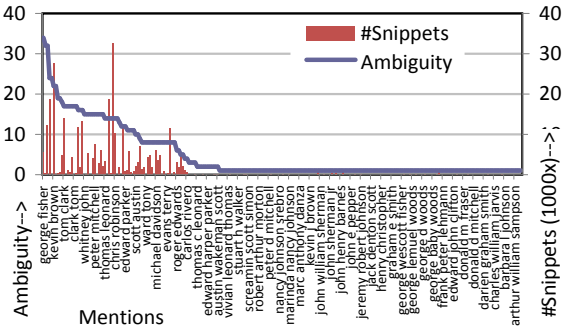


Fig. 11: Ambiguity distribution for Web corpus C. Unambiguous names are usually fully expanded and very rare, if at all present, in evaluated snippets.

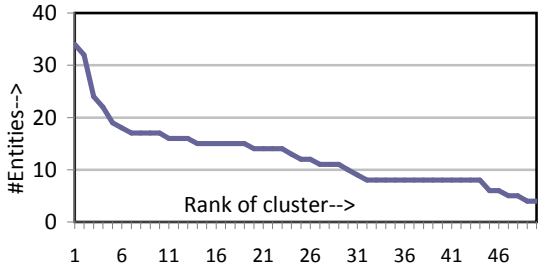


Fig. 12: Confusion cluster size distribution for Web corpus C.

loaded into a trie used by a map-reduce job to extract documents, then snippets, from the corpus. Some phrases in Figs. 10 and 11 have overwhelming numbers of pages with matches. In production, we naturally want all of them to be annotated. But human editorial judgment being the bottleneck, we sampled 50% or 50,000 snippets, whichever was smaller. Starting with about 752,450 pages, we ran the Stanford NER (Finkel et al., 2005) to mark person spans. Pages with fewer than five non-person tokens per person were discarded; this effectively discarded long list pages without any informative text to disambiguate anyone, and left us with 574,135 pages. From these we collected 304,309 snippets where the mention phrase is marked by the NER as a person. Each seed phrase leads to one cluster on which TMI and A0 are run. Note that $\perp$ must be allowed on the open Web.

5.2.2 Editorial judgment

Finally, for each algorithm, about 634 entity-type and about 4500 snippet-entity assignments are randomly sampled and sent to 20 editors in a commercial search engine company, who judged each assignment as correct or incorrect, *without knowing which algorithm* produced the annotation, to avoid

	TMI entity	A0
0/1 accuracy	**.714**	.562
Pooled recall	.764	**.869**
Precision	**.714**	.562
F1	**.738**	.683

	$e \in W$ (0/1 acc.)	$e \in F \setminus W$ (0/1)
TMI	**.75**	**.62**
A0	.65	.42

Fig. 13: Snippet summary for F and payload corpus C.

bias. Because the editors are trained professionals (unlike Mechanical Turks), we increased our evaluation coverage by having each type or entity assignment reviewed by one editor.

Pooling: Ideally, editors can be asked to find the best type or entity for each entity or snippet, but, given the size and diversity of Freebase, the cognitive burden would be unacceptable. In the Wikipedia corpus C_W, a snippet marked $\perp$ (no entity) by an algorithm can be judged a loss of recall if Wikipedia ground truth annotates it with an entity. Unfortunately, this is no longer practical for Web corpus C, because 8,217 snippets marked $\perp$ would have to be manually inspected and compared with a large number of candidate entities in Freebase. Therefore, we adopt pooling as in TREC. (Although the pool is small, A0 has very high recall.) Recall is evaluated with respect to the union of snippets annoted with a non-$\perp$ entity by *at least one* competing algorithm, with agreement in case of more than one.

0/1 Type accuracy: Editors judged each proposed type as correct or not. Unlike in §5.1, where the true and proposed types could be compared via M&W (Milne and Witten, 2008), they could not be asked to objectively estimate relatedness between types. Therefore we present only their reported post-hoc 0/1 accuracy for types: T0 and TMI have 0/1 type accuracy of 0.828 and 0.818.

5.2.3 Snippet annotation results

Given the large gap between TMI and Z0 in the easier setup in §5.1, we no longer consider Z0, and instead focus on TMI vs. A0. The summary comparison of A0 vs. TMI is shown in Fig. 13. Here TMI's absolute gains in 0/1 accuracy and F1 are even larger than in §5.1. To understand TMI's performance across a diversity of Freebase entity nodes e, as a function of 1. the size and richness of $N(e)$, and 2. the number of snippets claimed to mention e, we disaggregate the data of Fig. 13 into buckets of

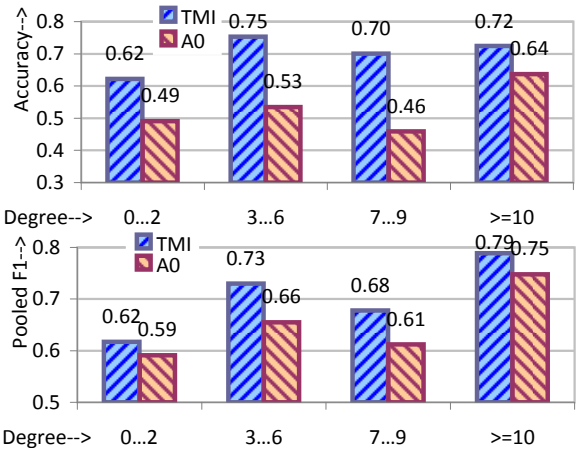


Fig. 14: 0/1 accuracy and F1 for snippets, payload corpus C and entities in F.

Snippet label judgements	%
TMI ok, FACC1 ok, neither $\perp$	22
TMI ok, FACC1 wrong, neither $\perp$	6
TMI$\neq \perp$ ok, FACC1=$\perp$ wrong	**40**
TMI$\neq \perp$ wrong, FACC1=$\perp$	23
TMI wrong, FACC1 wrong, neither $\perp$	2
TMI$= \perp$ wrong, FACC1$\neq \perp$ correct	4
TMI$= \perp$, FACC1$\neq \perp$, wrong	3
(TMI$= \perp$, FACC1$= \perp$, not judged)	-

Fig. 15: TMI vs. FACC1 comparison.

consecutive degrees, roughly balancing the number of snippets per bucket, as shown in Fig. 14. At the very low end of almost disconnected entity nodes, no algorithm does very well, because these entities are also hardly ever mentioned. When the entity is popular and well-connected, TMI's benefits are relatively modest. TMI's gains are best in the mid-range of degrees. The gap narrows for large-degree nodes, which is expected.

5.3 Comparison with FACC1

After collecting our pool of snippets as in §5.2.2, we consulted FACC1 (Gabrilovich et al., 2013), and passed on FACC1 annotations to our editors. As before, the identity of the algorithm was concealed. Results are shown in Fig. 15. In a large 40% of cases, TMI labels correctly while FACC1 backs off. The converse, where FACC1 backs off and TMI makes a mistake, is about half as frequent. These preliminary numbers suggest that TMI is able to push recall beyond FACC1 while also giving better precision.

6 Conclusion

We presented a formal model for bootstrapping from YAGO types and entities annotated in Wikipedia to two tasks, 1. annotating Web snippets with Freebase entities, and 2. associating Freebase entities with YAGO types. We presented TMI, a system to solve the two tasks jointly. Experiments show that TMI's snippet annotation accuracy, especially for relatively weakly-connected Freebase entities, is superior to baselines. We aim to extend from people to all major Freebase categories, and larger Web crawls.

Acknowledgment: We are grateful to Shrikant Naidu, Muthusamy Chelliah, and the editors from Yahoo! for their generous support. Shashank Gupta helped process FACC1 data.

References

S. Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP Conference*, pages 708–716.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL Conference*, pages 363–370.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora. `http://lemurproject.org/clueweb12/`, June. Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *SIGIR Conference*, pages 267–274. ACM.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in Web text: A graph-based method. In *SIGIR Conference*, pages 765–774.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP Conference*, pages 782–792, Edinburgh, Scotland, UK, July. SIGDAT.

Vladimir Kolmogorov. 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE PAMI*, 28(10):1568–1583, October.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of Wikipedia entities in Web text. In *SIGKDD Conference*, pages 457–466.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, October.

Xiaonan Li, Chengkai Li, and Cong Yu. 2010. EntityEngine: Answering entity-relationship queries using shallow semantics. In *CIKM*, October. (demo).

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *EMNLP Conference*, pages 893–903.

R Mihalcea and A Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242.

David Milne and Ian H Witten. 2008. Learning to link with Wikipedia. In *CIKM*, pages 509–518.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *ACL Conference*.

Patrick Pantel, Thomas Lin, and Michael Gamon. 2012. Mining entity types from query logs via user intent modeling. In *ACL Conference*, pages 563–571, Jeju Island, Korea, July.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *ACL Conference*, ACL/HLT, pages 1375–1384, Portland, Oregon.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP Conference*, pages 1524–1534, Edinburgh, UK. ACL.

G Salton and M J McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Uma Sawant and Soumen Chakrabarti. 2013. Learning joint query interpretation and response ranking. In *WWW Conference*, Brazil.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *WWW Conference*, pages 697–706. ACM Press.

ChengXiang Zhai. 2008. Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, March.

Zhicheng Zheng, Xiance Si, Fangtao Li, Edward Y. Chang, and Xiaoyan Zhu. 2012. Entity disambiguation with Freebase. In *Web Intelligence Conference*, WI-IAT '12, pages 82–89.

Effectiveness and Efficiency of Open Relation Extraction

Filipe Mesquita **Jordan Schmidek** **Denilson Barbosa**
Department of Computing Science, University of Alberta, Canada
{mesquita,schmidek,denilson}@ualberta.ca

Abstract

A large number of Open Relation Extraction approaches have been proposed recently, covering a wide range of NLP machinery, from "shallow" (e.g., part-of-speech tagging) to "deep" (e.g., semantic role labeling–SRL). A natural question then is what is the trade-off between NLP depth (and associated computational cost) versus effectiveness. This paper presents a fair and objective experimental comparison of 8 state-of-the-art approaches over 5 different datasets, and sheds some light on the issue. The paper also describes a novel method, EXEMPLAR, which adapts ideas from SRL to less costly NLP machinery, resulting in substantial gains both in efficiency and effectiveness, over binary and n-ary relation extraction tasks.

1 Introduction

Open Relation Extraction (ORE) (Banko and Etzioni, 2008) has become prevalent over traditional relation extraction methods, especially on the Web, because of the intrinsic difficulty in training individual extractors for every single relation. Broadly speaking, existing ORE approaches can be grouped according to the level of sophistication of the NLP techniques they rely upon: (1) shallow parsing, (2) dependency parsing and (3) semantic role labelling (SRL). Shallow methods annotate the sentences with part-of-speech (POS) tags and the ORE approaches in this category, such as ReVerb (Fader et al., 2011) and SONEX (Merhav et al., 2012), identify relations by matching patterns over such tags. Dependency parsing gives unambiguous relations among each word in the sentence, and the ORE approaches in this category such as PATTY (Nakashole et al., 2012), OLLIE (Mausam et al., 2012), and TreeKernel (Xu et al., 2013) identify whole subtrees connecting the relation predicate and its arguments. Finally, semantic annotators, such as Lund (Johansson and Nugues, 2008) and SwiRL (Surdeanu et al., 2003), add roles to each node in a parse tree, enabling ORE approaches that identify the precise connection between each argument and the predicate in a relation, independently.

The first contribution of the paper is an objective and fair experimental comparison of the state-of-the-art in ORE, on 5 datasets with varying degree of "difficulty". Of these, 4 datasets were annotated manually, covering both well-formed sentences, from the New York Times (NYT) and the Penn Treebank, as well as mixed-quality sentences from a popular Web corpus. A much larger corpus with 12,000 sentences from NYT, automatically annotated is also used. Another experiment focuses on n-ary relation extractions separately. The results show, as expected, that the three broad classes above are separated by orders of magnitude when it comes to throughput. Shallow methods handle ten times more sentences than dependency parsing methods, which in turn handle ten times more sentences than semantic parsing methods. Nevertheless, the cost-benefit trade-off is not as simple; and the higher computation cost of dependency or semantic parsing does not always pays off with higher effectiveness.

The second contribution of the paper is a new ORE method, called EXEMPLAR, which applies a key idea in semantic approaches (namely, to iden-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 447–457,
Seattle, Washington, USA, 18-21 October 2013.

tify the precise connection between the argument and the predicate words in a relation) over a dependency parse tree (i.e., without applying SRL). The goal is to achieve the higher accuracy of the semantic approaches at the lower computational cost of the dependency parsing approaches. EXEMPLAR is a rule-based system derived from a careful study of all dependency types identified by the Stanford parser. (Note, however, that other parsers can be used, as shown later on.) EXEMPLAR works for both binary and n-ary relations, and is evaluated separately in each case. For binary relations, EXEMPLAR outperforms all previous methods in terms of accuracy, losing to the shallow methods only in terms of throughput. As for n-ary relations, EXEMPLAR outperforms the methods that support this kind of extraction.

2 Related Work

Others have pointed out the importance of understanding the trade-off between "shallow" versus "deep" NLP in ORE. One side of the argument favors shallow methods, claiming deep NLP costs orders of magnitude more and provide much less dramatic gains in terms of effectiveness (Christensen et al., 2011). The counterpoint, illustrated with a recent analysis on a industrial-scale Web crawl (Dalvi et al., 2012), is that the diversity with which information is encoded in text is too high. Framing the debate as "shallow" versus "deep" is perhaps convenient, but nevertheless an oversimplification. This paper sheds more light into the debate by comparing the state-of-the-art from three broad classes of approaches.

Shallow ORE. TextRunner (Banko and Etzioni, 2008) and its successor ReVerb (Fader et al., 2011) are based on the idea that most relations are expressed using few syntactic patterns. ReVerb, for example, detects only three types of relations ("verb", "verb+preposition" and "verb+noun+preposition"). Following a similar approach, SONEX (Merhav et al., 2012) extends ReVerb by detecting patterns with appositions and possessives.

ORE via dependency parsing. PATTY (Nakashole et al., 2012) extracts textual patterns from sentences based on paths in the dependency graph. For all pairs of named entities, PATTY finds the shortest path in the dependency graph that connects the two named entities. They limit the search to only paths that start with one of these dependencies: nsubj, rcmod and partmod.

Dataset	Source	# Sentences	# Relations
WEB-500	Search Snippets	500	461
NYT-500	New York Times	500	150
PENN-100	Penn Treebank	100	51

Table 1: Binary relation datasets.

OLLIE (Mausam et al., 2012) also extracts relations between two entities. It applies pattern templates over the dependency subtree containing pairs of entities. Pattern templates are learned automatically from a large training set that is bootstrapped from high confidence extractions from ReVerb. OLLIE merges binary relations that differ only in the preposition and second argument to produce n-ary extractions, as in: (A, "met with", B) and (A, "met in", C) leading to (A, "met", [with B, in C]).

The TreeKernel (Xu et al., 2013) method uses a dependency tree kernel to classify whether candidate tree paths are indeed instances of relations. The shortest path between the two entities along with the shortest path between relational words and an entity are used as input to the tree kernel. An expanded set of syntactic patterns based on those from ReVerb are used to generate relation candidates.

ORE via semantic parsing. Recently, a method based on SRL, called SRL-IE, has shown that the effectiveness of ORE methods can be improved with semantic features (Christensen et al., 2011). We implemented our version of SRL-IE by relying on the output of two SRL systems: Lund (Johansson and Nugues, 2008) and SwiRL (Surdeanu et al., 2003). SwiRL is trained on PropBank and expands upon the syntactic features used in previous work. One of its major limitations is that it is only able to label arguments with verb predicates. Lund, on the other hand, is based on dependency parsing and is trained on both PropBank and NomBank, making it able to extract relations with both verb and noun predicates.

3 Experimental Study

This section compares the effectiveness and efficiency of the following ORE methods: ReVerb,

Method	NYT-500				WEB-500				PENN-100			
	Time	P	R	F	Time	P	R	F	Time	P	R	F
ReVerb	**0.02**	0.70	0.11	0.18	**0.01**	0.92	0.29	0.44	**0.02**	0.78	0.14	0.23
SONEX	0.04	0.77	0.22	0.34	0.02	**0.98**	0.30	0.45	0.04	**0.92**	0.43	0.59
OLLIE	0.05	0.62	0.27	0.38	0.04	0.81	0.29	0.43	0.14	0.81	0.43	0.56
EXEMPLAR[M]	0.08	0.70	**0.39**	0.50	0.06	0.95	0.44	0.61	0.16	0.83	0.49	**0.62**
EXEMPLAR[S]	1.03	0.73	**0.39**	**0.51**	0.47	0.96	**0.46**	**0.62**	0.62	0.79	**0.51**	**0.62**
PATTY	1.18	0.49	0.23	0.32	0.48	0.71	0.48	0.57	0.66	0.46	0.24	0.31
SwiRL	2.96	0.63	0.10	0.17	1.73	0.97	0.34	0.50	2.17	0.89	0.16	0.27
Lund	11.40	**0.78**	0.24	0.37	2.69	0.91	0.37	0.52	5.21	0.86	0.35	0.50
TreeKernel	–	–	–	–	–	–	–	–	0.85	0.85	0.33	0.48

Table 2: Results for the task of extracting binary relations. Methods are ordered by computing time per sentence (in seconds). Best results for each column are underlined and marked in bold, for clarity.

SONEX, OLLIE, PATTY, TreeKernel, SwiRL, Lund and our two variants of our method, EXEMPLAR, explained in detail in Appendix A: (1) EXEMPLAR[S] uses the Stanford parser (Klein and Manning, 2003) and (2) EXEMPLAR[M] uses the Malt parser (Nivre and Nilsson, 2004).

3.1 Binary Relations – Setup

We start by evaluating the extraction of binary relations. Table 1 shows our experimental datasets. WEB-500 is a commonly used dataset, developed for the TextRunner experiments (Banko and Etzioni, 2008). These sentences are often incomplete and grammatically unsound, representing the challenges of dealing with web text. NYT-500 represents the other end of the spectrum with formal, well written new stories from the New York Times Corpus (Sandhaus, 2008). PENN-100 contains sentences from the Penn Treebank recently used in an evaluation of the TreeKernel method (Xu et al., 2013). We manually annotated the relations for WEB-500 and NYT-500 and use the PENN-100 annotations provided by TreeKernel's authors (Xu et al., 2013).

We annotate each sentence manually as follows. We identify exactly two entities and a *trigger* (a single token indicating a relation–see Section A.3) for the relation between them, if one exists. In addition, we specify a *window* of tokens allowed to be in a relation, including modifiers of the trigger and prepositions connecting triggers to their arguments. For each sentence annotated with two entities, a system must extract a string representing the relation between them. Our evaluation method deems an extraction as correct if it contains the trigger and allowed tokens only.

In our annotated sentences, entities are enclosed in triple square brackets, triggers and enclosed in triple curly brackets and the window of allowed tokens is defined by arrows ("--->" and "<---"), as in this example:

```
I've got a media call about
[[[ORG Google]]] --->'s
{{{acquisition}}} of<--- [[[ORG
YouTube]]] --->today<---.
```

where "Google" and "YouTube" are entities of the type organization, "acquisition" is the trigger and the allowed tokens are "acquisition", "'s" and "of". We include time and location modifiers (e.g., "today", "here") in the list of allowed tokens since OLLIE extracts them as part of the relation. OLLIE's extractions may also include auxiliary verbs and prepositions that are not present in the original sentence. To be fair with OLLIE, we remove auxiliary verbs and prepositions from OLLIE extractions.

Our benchmarks are available upon request.

Ensuring entities are recognized properly. Since every method uses a different tool to recognize entities, we try to ensure every method is able to recognize the entities marked by our annotators. We replace the original entities by a single word, preventing any system from recognizing only part of an entity. Entities are replaced by "Europe" and "Asia", since we empirically found that, for 99.7% of the sentences in our experiment, all methods were able to recognize "Europe" and "Asia" as entities (or nouns, for systems that do not use a NER tool). In addition, we did not find any occurrence of

"Europe" and "Asia" in the original sentences that could conflict with our entity placeholders.

For methods that extract relations between noun phrases (ReVerb, OLLIE, SwiRL and Lund), there is the additional task of identifying whether a noun phrase containing additional words surrounding "Europe" and "Asia" is still a reference to the annotated entity. For example, "the beautiful Europe" refers to the entity, while "Europe's enemy" does not. In our evaluation, we ignore noun phrases that do not reference the annotated entity. For SwiRL and Lund, we ignore any noun phrase that do not present "Europe" or "Asia" as its head word. For ReVerb and OLLIE, we ignore noun phrases that do not contain these words in the end of the phrase.

Metrics. Our evaluation focuses on sentence-level extractions. Therefore, we only apply the steps of each method that perform this task. Additional steps to merge relations and remove infrequent relations are not applied. In addition, we assume there is only one relation between a pair of entities in a sentence. The number of entity pairs with more than one relation was insignificant in our datasets (less than 0.5%).

The metrics used in this analysis are precision (P), recall (R) and f-measure (F), defined as usual:

$$P = \frac{\text{\# correct}}{\text{\# extractions}}, R = \frac{\text{\# correct}}{\text{\# relations}}, F = \frac{2PR}{P+R}$$

where "# correct" is the number of extractions deemed as correct.

We also measure the total computing time of each method, excluding initialization or loading any libraries or models in memory. To ensure a fair comparison, we make sure each method runs in a single-threaded mode, thus utilizing a single computing core at all times.

3.2 Binary Relations – Results

Table 2 presents the results for our experiment with binary relations. WEB-500 turned out to contain the easiest sentences as evidenced by the precision of all methods in this dataset. This is because WEB-500 sentences were collected by querying a search engine with known relation instances. The other two datasets, on the other hand, contain randomly chosen sentences. Although WEB-500 is a popular dataset, it perhaps does not represent the challenges found in web text.

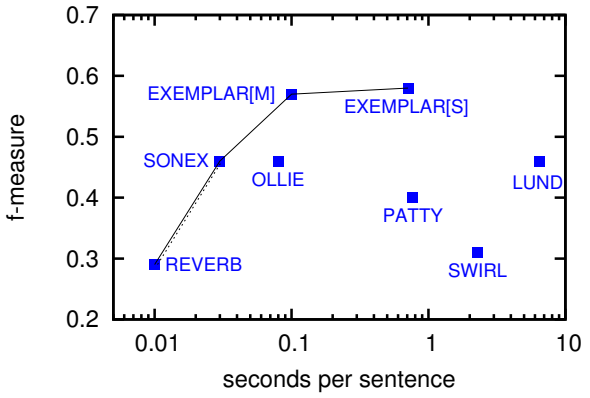


Figure 1: Average f-measure vs average time for NYT-500, WEB-500 and PEN-100.

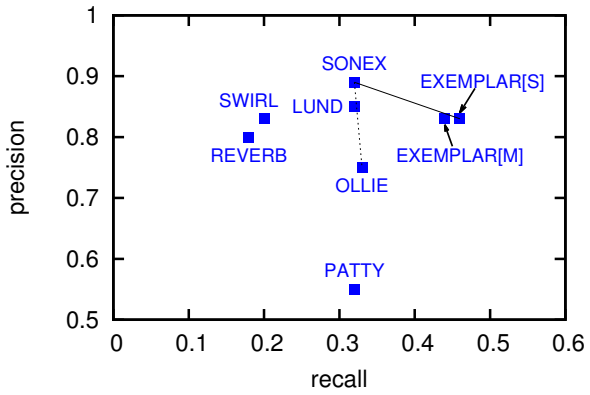


Figure 2: Average precision vs average recall for NYT-500, WEB-500 and PEN-100.

We were unable to run TreeKernel for NYT-500 and WEB-500 for lack of training data. We ran TreeKernel, as trained by its authors, on the same test set used in their paper (Xu et al., 2013).

Comparing methods based on effectiveness (f-measure) or efficiency (computational cost) alone can be misleading. To do so, we compare methods in terms of *dominance*. We say method A dominates method B if A is: (1) *more effective* and as efficient as B; (2) *more efficient* and as effective as B; or (3) both more effective and more efficient than B. The methods that are not dominated by any other form the state-of-the-art.

Figure 1 plots the effectiveness and efficiency of

all methods, averaged over all datasets (TreeKernel was not included due to missing results). For efficiency, there is a clear separation of approximately one order of magnitude among methods based on shallow parsing (ReVerb and SONEX), dependency parsing (OLLIE, EXEMPLAR[M], EXEMPLAR[S], and PATTY) and semantic parsing (SwiRL and Lund). The lines in the plot identify the state-of-the-art before (dashed) and after (solid) EXEMPLAR. (Note: Although not clear in the figure, OLLIE and Lund are dominated by SONEX as they tie in terms of effectiveness.)

In terms of efficiency, EXEMPLAR[M] and EXEMPLAR[S] closely match OLLIE and PATTY, respectively, since they use the same dependency parsers. EXEMPLAR outperforms both systems by covering a larger number of relational patterns. This is possible because EXEMPLAR looks at each argument separately, as opposed to the whole subtree connecting two arguments. As it turns out, this design choice greatly simplifies the task of designing good patterns.

The poor performance of PATTY is due to its rather permissive sentence-level extraction of relations, which looks at the shortest path between arguments. PATTY relies on redundancy of extractions to normalize the produced relations in order to recover from mistakes done in the sentence-level.

Figure 2 illustrates the dominance relation differently, using precision versus recall. Again, the dashed line shows the previous state-of-the-art, and the solid line shows the current situation. SONEX dominates PATTY and Lund, since they tied in recall. OLLIE, however, achieved greater recall than SONEX. Somewhat surprisingly, EXEMPLAR presents 44% more recall than the more sophisticated Lund, at a close level of precision. This can be explained by Lund's dependency on training data, which contains only a subset of all possible predicates and roles. The importance of relations triggered by nouns is illustrated by the higher recall of SONEX and Lund when compared, respectively, to ReVerb and SwiRL, similar methods that handle verb triggers only.

3.3 Binary Relations – Discussion

Differences in annotation. It is worth noting some differences between our annotations (WEB-500 and NYT-500) and the annotations from PEN-100. The first difference concerns the definition of an entity. Consider the following sentence from PEN-100:

> "... says Leslie Quick Jr., chairman of the Quick & Reilly discount brokerage firm."

Unlike our annotation style, the original annotation defines that "Leslie Quick Jr." is the chairman of "the Quick & Reilly discount brokerage firm", as opposed to "Quick & Reilly". While we consider the words surrounding "Quick & Reilly" as apposition, the original consider them as part of the entity.

Another difference concerns the definition of the RE task. We assume that RE methods are responsible for resolving co-references when necessary to identify a relation. For example, consider the sentence:

> "It also marks P&G's growing concern that its Japanese rivals, such as Kao Corp., may bring their superconcentrates to the U.S."

According to our annotation style, there is a relation "rivals" between "P&G" and "Kao Corp." in this sentence. On the other hand, the original annotations for PENN-100 consider only the relation between "Kap Corp." and the pronoun "it", leaving the task of resolving the coreference between "P&G" and "it" as a posterior step.

These differences in annotation illustrate the challenges of producing a benchmark for open relation extraction.

Differences in evaluation methodology. A *sentence-level* evaluation like ours focuses on each sentence, separately. On the other hand, the evaluations of SONEX, ReVerb, PATTY, TreeKernel and OLLIE are performed at *the corpus level*. Corpus-level evaluations consider an extracted relation as correct regardless of whether a method was able to identify one or all sentences that describe this relation.

Creating a ground truth for corpus-level evaluations is extremely hard, since one has to identify and curate (e.g., merge near-duplicate relations and co-referential entities) all relations described in a corpus. As a consequence, most corpus-level evaluations perform only a manual inspection of a

Method	NYT n-ary			
	Time	P	R	F
EXEMPLAR[M]	**0.11**	0.94	**0.40**	**0.56**
OLLIE	0.12	0.87	0.14	0.25
EXEMPLAR[S]	0.88	0.92	0.39	0.55
SwiRL	2.90	0.94	0.30	0.45
Lund	9.20	**0.95**	0.36	0.53

Table 3: Results for n-ary relations.

Method	NYT 12K			
	Time	P	R	F
ReVerb	0.01	0.84	0.11	0.19
OLLIE	0.02	0.85	0.22	0.35
SONEX	0.03	**0.87**	0.20	0.32
EXEMPLAR[M]	0.05	**0.87**	0.26	0.40
EXEMPLAR[S]	1.20	0.86	**0.29**	**0.43**
PATTY	1.29	0.86	0.18	0.30
SwiRL	3.58	**0.87**	0.16	0.27
Lund	11.28	0.86	0.21	0.33

Table 4: Results for binary relations automatically annotated using Freebase and WordNet.

method's extractions. This manual inspection measures a method's precision, but is unable to measure recall.

Other differences in methodology are as follows. PATTY's evaluation concerns relation patterns (e.g., "wrote hits for") and their type signatures (e.g. Musician–Musician), as opposed to the relation itself, which includes its two arguments. The evaluations of ReVerb and OLLIE consider any noun phrase as a potential argument, while the evaluations of TreeKernel and SONEX consider named entities only.

Due to the lack of a ground truth and differences in evaluation methodology, results from different papers are usually not comparable. This work tries to alleviate this problem by providing reusable annotations that are flexible and can be used to evaluate a wide range of methods.

3.4 n-ary Relations

The goal of this experiment is to evaluate the accuracy and performance of our method when extracting n-ary relations ($n > 2$). For this experiment, we manually tagged 222 sentences with n-ary relations from the New York Times. Every sentence is annotated with a single relation trigger and its arguments.

This experiment measures precision and recall over the extracted arguments. For each sentence, a system can extract a number of relations of the form $r(a_1, a_2, \ldots, a_n)$, where r is the relation name and a_i is an argument. We only use the extracted relation whose name contains the annotated trigger, if it exists. An argument of such relation is deemed a correct extraction if it is annotated in the sentence; otherwise, it is deemed incorrect.

Precision and recall are now defined as follows:

$$P = \frac{\text{\# correct}}{\text{\# extracted args}}, R = \frac{\text{\# correct}}{\text{\# annotated args}}.$$

where "# correct" is the number of arguments deemed as correct. There are 765 annotated arguments in total. Table 3 reports the results for our experiment with n-ary relations. EXEMPLAR[M] shows a 6% increase in f-measure over Lund, the second best system, while being almost two orders of magnitude faster.

3.5 Automatically Annotated Sentences

The creation of datasets for open RE is an extremely time-consuming task. In this section we investigate whether external data sources such as Freebase[1] and WordNet[2] can be used to automatically annotate a dataset, leading to a useful benchmark.

Our *automatic annotator* identifies pairs of entities and a trigger of the relation between them. It does so by first trying to link all entities to Wikipedia (and consequently to Freebase, since Freebase is linked to Wikipedia) by using the method proposed by (Cucerzan, 2007). Given two entities appearing within 10 tokens of each other in a sentence, our annotator checks whether there is a relation connecting them in Freebase. If such a relation exists, the annotator looks for a trigger in the sentence. A trigger must be a synonym for the Freebase relation (according to WordNet) and its distance to the nearest entity cannot be more than 5 tokens.

We applied this method for the New York Times and were able to annotate over 60,000 sentences with over 13,000 distinct entity pairs. For our experiments, we randomly selected one sentence for each entity pair and separated a thousand for development and over 12,000 for test.

[1] http://www.freebase.com
[2] http://wordnet.princeton.edu/

Comparing with human annotators. Although we expect our automatic annotator to be less accurate than a human annotator, we are interested to measure the difference in accuracy between them. To do so, two authors of this paper looked at our development set and marked each sentence as correct or incorrect. The agreement (that is, the percentage of matching answers) between the humans was 82%. On other hand, the agreement between our automatic annotator and each human was 71% and 72%. This shows that our annotator's accuracy is not too far below human's level of accuracy.

Table 4 shows the results for the test sentences. Both EXEMPLAR[S] and EXEMPLAR[M] outperformed all systems in recall, while keeping the same level of precision.

4 Conclusion

This work presents a fair and objective evaluation of several ORE methods, shedding some light on the trade-offs between f-measure and computational as well as precision and recall. Our evaluation is able to assess the effectiveness of different methods by specifying a trigger and a window of allowed tokens for each relation.

EXEMPLAR's promising results indicate that rule-based methods may still be very competitive, especially if rules are applied to each argument separately. Looking at the recall levels of different methods, we conjecture that EXEMPLAR outperforms machine learning methods like Ollie and TreeKernel, because its rules apply in cases not trained by these methods. From a pragmatic point of view, EXEMPLAR is also preferable because it doesn't require training data.

An interesting research question is whether machine learning can be used to learn more rules for EXEMPLAR in order to improve recall without loss in precision. Rules could be learned from both dependency parsing and shallow parsing, or just shallow parsing if computing time is extremely limited.

The next step for our experimental study is to evaluate corpus-level extractions, where an automatic annotator is essential due to the massive number of annotations required for even one relation, let alone thousands.

Acknowledgements

We thank the anonymous reviewers for useful suggestions to improve the paper, and the Natural Sciences and Engineering Council of Canada, through the NSERC Business Intelligence Network, for financial support.

A EXEMPLAR

ORE methods must recognize relations from the text alone. To do this, each method tries to model how relations are expressed in English. Banko and Etzioni claim that more than 90% of binary relations are expressed through a few syntactic patterns, such as "verb" and "noun+preposition" (Banko and Etzioni, 2008). It is unclear, however, whether n-ary relations follow.

This section presents a study focusing on how n-ary relations ($n > 2$) are expressed in English, based on 100 distinct relations manually annotated from a random sample of 514 sentences in the New York Times Corpus (Sandhaus, 2008).

Table 5 shows six syntactic patterns that cover 86% of our n-ary relations. These patterns are slightly different from those used for binary relations. In binary relations, the pattern implicitly defines the roles of the two arguments. For instance, a relation "met with" indicates that the first argument is the subject and the second one is the object of "with". In order to represent n-ary relations, our patterns do not contain prepositions, possessives or any other word connecting the relation to the argument. For instance, the sentence "Obama met with Putin in Russia" contains the relation "meet" along three arguments: "Obama" (subject), "Putin" (object of preposition with) and "Russia" (object of preposition in).

Relation types. A single relation can be represented in different ways using the patterns shown in Table 5. For instance, the relation "donate" can be expressed as an active verb ("donates"), passive voice ("was donated by") and normalized verb ("donation"). In addition, an apposition+noun relation can be expressed as an copula+noun relation by replacing apposition for the copula verb "be". By merging these patterns, we have that most relations fall into one of the following types: *verb*, *verb+noun*

Pattern	Frequency	Example
Verb	30%	Hingis **beat** Steffi Graf in the Italian Open two weeks ago.
Apposition+noun	19%	Jaden and Willow, the **famous children** of Will Smith, ...
Passive verb	14%	Jumbo the Elephant **was exported** from Sudan to Paris.
Verb+noun	14%	D-League will **move its offices** from Greenville to New York.
Copula+noun	5%	Kimball **was a** Fulbright **scholar** at the University of Heidelberg.
Nominalized verb	4%	Thousands died in Saddam Hussein's **attack** on Halabja in 1988.

Table 5: Patterns representing 86% of the relations with three or more arguments. Frequencies collected from 100 relations from the New York Times Corpus. Relation triggers are highlighted in bold.

Relation Type	Freq.	Example	Variations
Verb	48%	beat	Pass. verb, nom. verb
Copula+noun	24%	is son	Apposition+noun
Verb+noun	14%	sign deal	–

Table 6: Relation types recognized by EXEMPLAR.

and *copula+noun*.

Table 6 present the relation types found through our analysis. We developed EXEMPLAR to specifically recognize these relation types, including their variations.

Argument roles. An *argument role* defines how an argument participates in a relation. In ORE, the roles for each relation are not provided and must also be recognized from the text.

We use the following roles: `subject`, `direct_object` and `prep_object`. An argument has a role `prep_object` when its connected to the relation by a preposition. The roles of prepositional objects consist of their preposition and the suffix "_object", indicating that each preposition corresponds to a different role. In the sentence "Obama is the president ***of*** the U.S.", "U.S." is an object of the preposition "of" and has the role `of_object`.

Multiple entities can play the same role in a relation instance. For instance, in the sentence "Obama and Putin discuss the Syria conflict", both "Obama" and "Putin" have the `subject` role. Furthermore, some relations accept less roles than others. Verb relations accept all three roles, while copula+noun and verb+noun relations accept `subject` and `prep_object` only.

Our roles are different from those used in SRL. SRL roles carry semantic information across different relations. This information is unavailable for ORE systems, and for this reason, we rely on syntactic roles. An open problem is to determine which syntactic roles correspond to the same semantic role across different relations (Chambers and Jurafsky, 2011). However, this problem is out of the scope of this work.

```
  subject: "NFL"
 relation: "approve new stadium"
of_object: "Falcons"
in_object: "Atlanta"
```

Figure 3: A relation instance extracted by EXEMPLAR for the sentence "NFL approves Falcons' new stadium in Atlanta".

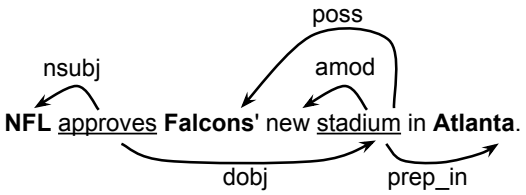


Figure 4: A input sentence after pre-processing. Entities are in bold, triggers are underline and arrows represent dependencies.

A.1 The method

EXEMPLAR takes a stream of textual documents and extracts instances of n-ary relations as illustrated in Figure 3.

A.2 Preprocessing

Given a document, EXEMPLAR extracts its syntactic structure by applying a pipeline of NLP tools provided by the Stanford Parser (Klein and Manning, 2003). Our method converts the original text into sentences, each containing a list of tokens. Each token is tagged with part of speech, lemma and dependencies. EXEMPLAR also works with other dependency parsers based on Stanford's dependencies,

such as the Malt parser (Nivre and Nilsson, 2004).

Figure 4 illustrates our running example where each word is a token and arrows represent dependencies among tokens. In this example, "stadium" depends on "approves" and the arrow connecting them can be read as "the direct object of *approves* is *stadium*".

Extracting Named Entities. EXEMPLAR employs the Stanford NER (Finkel et al., 2005) to recognize named entities. We consider these types of entities: people, organization, location, miscellaneous and date. Figure 4 shows entities highlighted in bold.

A.3 Detecting triggers

After recognizing entities, EXEMPLAR detects *relation triggers*. A trigger is a single token that indicates the presence of a relation. A relation may present one or two triggers. For instance, the relation in our running example has two triggers. EXEMPLAR also uses triggers to determine the relation name, as discussed later.

A trigger can be any noun or verb that was not tagged as being part of an entity mention. Other requirements are enforced for each canonical pattern.

Verb relations. A verb relation is triggered by a verb that does not include a noun as its direct object. The name of the relation is the trigger's lemma.

A noun must be a nominalized verb to be a trigger for verb relations. To identify nominalized verbs, EXEMPLAR checks if a noun is filed under the type "event" in Wordnet's Morphosemantic Database[3]. Doing so may generate false positives; however, EXEMPLAR has a filtering step to eliminate these false positives, as discussed later.

The name of a relation triggered by a nominalized verb is the trigger's original verb (before nominalization). For instance, "donation" triggers the relation "donate".

Copula+noun relations. Only noun triggers are accepted for copula+noun relations. The copula used in the relation name can be a verb with copula dependency to the trigger, or the verb "be" for appositions. The relation name is the concatenation of the copula's lemma and the trigger's lemma along its modifiers. For instance, the relation in the sentence "Jaden and Willow, the famous children of Will Smith" is "be famous child".

Verb+noun relations. EXEMPLAR recognizes two triggers for each verb+noun relation: a verb and a noun acting as its direct object. The relation name is defined by concatenating the verb's lemma with the the noun and its modifiers. In our running example, "approves" and "stadium" trigger the relation "approve new stadium".

A.4 Detecting candidate arguments

After relation triggers are identified, EXEMPLAR proceeds to detect their *candidate arguments*. For this, we look at the dependency between each entity and a trigger separately. EXEMPLAR relies on two observations: (1) an argument is often adjacent to a trigger in the dependency graph, and (2) the type of the dependency can accurately predict whether an entity is an argument for the relation or not.

Table 7 enumerates 12 types of dependencies (from a total of 53) that often connect arguments and triggers. EXEMPLAR identifies as a candidate argument every entity that is connected to trigger, as long as their dependency type is listed in Table 7.

Our observations can be seen in our running example. The entities "NFL" and "Atlanta" depends on the trigger "approves" and "Falcons" depend on the trigger "stadium". Since their dependency types are listed in Table 7, these entities are marked as candidate arguments.

A.5 Role Detection

EXEMPLAR determines the role of an argument based on the trigger type (noun or verb), the type of dependency between the trigger and argument and the direction of the dependency. To take into account the dependency direction, we prefix each dependency type with "$>$" when an entity depends on the trigger and "$<$" when the trigger depends on the entity.

Table 8 shows EXEMPLAR's rules that assign roles to arguments for each relation type. Rules are triples ($trigger$, $dependency$, $role$) whose meaning is as follows:

[3] http://wordnetcode.princeton.edu/standoff-files/morphosemantic-links.xls

Dependency	Example
nsubj (Subject)	**Romeo** loves Juliet
dobj (Direct Object)	The **Prince** exiles Romeo
nsubjpass (Pass. Subj.)	**Romeo** was seen in Verona
agent (Pass. Voice Obj.)	Juliet is loved by **Romeo**
iobj (Indirect Object)	Romeo gave **Juliet** a kiss
poss (Possessive)	**Romeo**'s father Montague
appos (Apposition)	**Capulet**, Juliet's father,
amod (Adj. Modifier)	The **Italian** city of Verona
nn (Noun Comp. Mod.)	Romeo's cousin **Benvolio**
prep_* (Prep. Modifier)	Romeo lived in **Verona**
partmod (Participal Mod.)	**Romeo**, born in Italy
rcmod (Rel. Clause Mod.)	**Juliet**, who loved Romeo

Table 7: Dependencies connecting arguments and triggers. Arguments are in bold and triggers are underlined.

> If trigger type = *trigger* and dependency type = *dependency* then assign *role*.

For example, the first rule in Table 8a specifies that an argument must be assigned the role of a *subject* if this argument depends on a *verb* trigger and the dependency type is *>nsubj*.

Rules are ordered by descending priority in Table 8. In case several rules can be applied for an argument, we apply only the rule with higher priority. If none of the rules applies to an argument, EXEMPLAR removes this argument from the relation.

Exceptions. There are three exceptions for the rules above. The first exception concerns arguments of verb relations whose dependency type is <partmod or <rcmod. EXEMPLAR chooses the role of direct_object (as oppose to subject) for these arguments *when the verb trigger is in passive form*. For instance, in the sentence "Barbie, (which was) invented by Handler", "Barbie" has the role direct_object because "invented" is in passive form.

The second exception is for nominalized verbs followed by the preposition "by", such as in "Georgian invasion by Russia". Arguments of this type of trigger with dependency types >nn, >amod or >poss are assigned the role direct_object.

Finallly, there is an exception for copula+noun relations expressed with close appositions of the form: "determiner entity noun". An example is "the Greenwood Heights section of Brooklyn". Here, EXEMPLAR assigns the subject role to the entity between the determiner and the noun.

Trigger Type	Dependency Type	Role
Verb	>nsubj	subject
Verb	>agent	subject
Verb	<partmod	subject
Verb	<rcmod	subject
Verb	>dobj	direct_object
Verb	>subjpass	direct_object
Verb	>iobj	to_object
Verb	>prep_*	prep_object
Noun	>prep_by	subject
Noun	>amod	subject
Noun	>nn	subject
Noun	>poss	subject
Noun	>prep_of	direct_object
Noun	>prep_*	prep_object

(a) Rules for verb relations.

Trigger Type	Dependency Type	Role
Noun	>nsubj	subject
Noun	>appos	subject
Noun	<appos	subject
Noun	<partmod	subject
Noun	<rcmod	subject
Noun	>prep_of	of_object
Noun	>amod	of_object
Noun	>nn	of_object
Noun	>poss	of_object
Noun	>prep_*	prep_object

(b) Rules for copula+noun relations.

Trigger Type	Dependency Type	Role
Verb	>nsubj	subject
Verb	>agent	subject
Verb	<partmod	subject
Verb	<rcmod	subject
Verb	>iobj	to_object
Verb	>prep_*	prep_object
Noun	>amod	of_object
Noun	>nn	of_object
Noun	>poss	of_object
Noun	>prep_*	prep_object

(c) Rules for verb+noun relations.

Table 8: Rules for assigning roles to arguments.

A.6 Filtering Relations

The final step in EXEMPLAR is to remove incomplete relations. EXEMPLAR removes relations with less than two arguments and relations that do not present subject and direct_object.

For EXEMPLAR, Lund and SwiRL, which extract n-ary relations, our evaluation needs to convert n-ary relations into binary ones. This is done by selecting all pairs of arguments from a n-ary relation and creating a new (binary) relation for each of them. Binary relations containing two prepositional objects (or equivalent for SRL systems) are removed.

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 28–36, Columbus, Ohio, June. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 976–986, Stroudsburg, PA, USA. Association for Computational Linguistics.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, pages 113–120. Association for Computational Linguistics.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'12, pages 708–716.

Nilesh Dalvi, Ashwin Machanavajjhala, and Bo Pang. 2012. An analysis of structured data on the web. *Proc. VLDB Endow.*, 5(7):680–691.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, EMNLP'12.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL'05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'08, pages 69–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL'03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'12.

Yuval Merhav, Filipe de Sá Mesquita, Denilson Barbosa, Wai Gen Yee, and Ophir Frieder. 2012. Extracting information networks from the blogosphere. *TWEB*, 6(3):11.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'12, pages 1135–1145, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of Computational Natural Language Learning*, CoNLL'04.

Evan Sandhaus. 2008. The new york times annotated corpus. `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2008T19`.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 8–15. Association for Computational Linguistics.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 868–877, Atlanta, Georgia, June. Association for Computational Linguistics.

Automatic Feature Engineering for Answer Selection and Extraction

Aliaksei Severyn
DISI, University of Trento
38123 Povo (TN), Italy
severyn@disi.unitn.it

Alessandro Moschitti
Qatar Computing Research Institue
5825 Doha, Qatar
amoschitti@qf.org.qa

Abstract

This paper proposes a framework for automatically engineering features for two important tasks of question answering: answer sentence selection and answer extraction. We represent question and answer sentence pairs with linguistic structures enriched by semantic information, where the latter is produced by automatic classifiers, e.g., question classifier and Named Entity Recognizer. Tree kernels applied to such structures enable a simple way to generate highly discriminative structural features that combine syntactic and semantic information encoded in the input trees. We conduct experiments on a public benchmark from TREC to compare with previous systems for answer sentence selection and answer extraction. The results show that our models greatly improve on the state of the art, e.g., up to 22% on F1 (relative improvement) for answer extraction, while using no additional resources and no manual feature engineering.

1 Introduction

Question Answering (QA) systems are typically built from three main macro-modules: (i) search and retrieval of candidate passages; (ii) reranking or selection of the most promising passages; and (iii) answer extraction. The last two steps are the most interesting from a Natural Language Processing viewpoint since deep linguistic analysis can be carried out as the input is just a limited set of candidates.

Answer sentence selection refers to the task of selecting the sentence containing the correct answer among the different sentence candidates retrieved by a search engine.

Answer extraction is a final step, required for factoid questions, consisting in extracting multiwords constituting the synthetic answer, e.g., *Barack Obama* for a question: *Who is the US president?* The definition of rules for both tasks is conceptually demanding and involves the use of syntactic and semantic properties of the questions and its related answer passages.

For example, given a question from TREC QA[1]:

> *Q: What was Johnny Appleseed's real name?*

and a relevant passage, e.g., retrieved by a search engine:

> *A: Appleseed, whose real name was John Chapman, planted many trees in the early 1800s.*

a rule detecting the semantic links between *Johnny Appleseed's real name* and the correct answer *John Chapman* in the answer sentence has to be engineered. This requires the definition of other rules that associate the question pattern `real_name_?(X)` with `real_name_is(X)` of the answer sentence. Although this can be done by an expert NLP engineer, the effort for achieving the necessary coverage and a reasonable accuracy is not negligible.

An alternative to manual rule definition is the use of machine learning, which often shifts the problem

[1] We use it as our running example in the rest of the paper.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 458–467,
Seattle, Washington, USA, 18-21 October 2013.

to the easier task of feature engineering. Unfortunately, when the learning task is semantically difficult such as in QA, e.g., features have to encode combinations of syntactic and semantic properties. Thus their extraction modules basically assume the shape of high-level rules, which are, in any case, essential to achieve state-of-the-art accuracy. For example, the great IBM Watson system (Ferrucci et al., 2010) uses a learning to rank algorithm fed with hundreds of features. The extraction of some of the latter requires articulated rules/algorithms, which, in terms of complexity, are very similar to those constituting typical handcrafted QA systems. An immediate consequence is the reduced adaptability to new domains, which requires a substantial reengineering work.

In this paper, we show that tree kernels (Collins and Duffy, 2002; Moschitti, 2006) can be applied to automatically learn complex structural patterns for both answer sentence selection and answer extraction. Such patterns are syntactic/semantic structures occurring in question and answer passages. To make such information available to the tree kernel functions, we rely on the shallow syntactic trees enriched with semantic information (Severyn et al., 2013b; Severyn et al., 2013a), e.g., Named Entities (NEs) and question focus and category, automatically derived by machine learning modules, e.g., question classifier (QC) or focus classifier (FC).

More in detail, we (i) design a pair of shallow syntactic trees (one for the question and one for the answer sentence); (ii) connect them with relational nodes (i.e., those matching the same words in the question and in the answer passages); (iii) label the tree nodes with semantic information such as question category and focus and NEs; and (iv) use the NE type to establish additional semantic links between the candidate answer, i.e., an NE, and the focus word of the question. Finally, for the task of answer extraction we also connect such semantic information to the answer sentence trees such that we can learn factoid answer patterns.

We show that our models are very effective in producing features for both answer selection and extraction by experimenting with TREC QA corpora and directly comparing with the state of the art, e.g., (Wang et al., 2007; Yao et al., 2013). The results show that our methods greatly improve on both tasks yielding a large improvement in Mean Average Precision for answer selection and in F1 for answer extraction: up to 22% of relative improvement in F1, when small training data is used. Moreover, in contrast to the previous work, our model does not rely on external resources, e.g., WordNet, or complex features in addition to the structural kernel model.

The reminder of this paper is organized as follows, Sec. 2 describes our kernel-based classifiers, Sec. 3 illustrates our question/answer relational structures also enriched with semantic information, Sec. 4 describes our model for answer selection and extraction, Sec. 5 illustrates our comparative experiments on TREC data, Sec. 6 reports on our error analysis, Sec. 7 discusses the related work, and finally, Sec. 8 derives the conclusions.

2 Structural Kernels for classification

This section describes a kernel framework where the input question/answer pairs are handled directly in the form of syntactic/semantic structures.

2.1 Feature vector approach to object pair classification

A conventional approach to represent a question/answer pairs in linear models consists in defining a set of similarity features $\{x_i\}$ and computing the simple scalar product $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_i w_i x_i$, where $\boldsymbol{w}$ is the model weight vector learned on the training data. Hence, the learning problem boils down to estimating individual weights of each of the similarity features x_i. Such features often encode various types of lexical, syntactic and semantic similarities shared between a question and its candidate. Previous work used a rich number of distributional semantic, knowledge-based, translation and paraphrase resources to build explicit feature vector representations. One evident potential downside of using feature vectors is that a great deal of structural information encoded in a given text pair is lost.

2.2 Pair Classification using Structural Kernels

A more versatile approach in terms of the input representation relies on kernels. A typical kernel machine, e.g., SVM, classifies a test input $\boldsymbol{x}$ using the following prediction function: $h(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, where α_i are the model parameters estimated from the training data, y_i are target

variables, $\boldsymbol{x}_i$ are support vectors, and $K(\cdot,\cdot)$ is a kernel function. The latter can measure the *similarity* between question and answer pairs.

We define each question/answer pair $\boldsymbol{x}$ as a triple composed of a question tree $\boldsymbol{T}_q$ and answer sentence tree $\boldsymbol{T}_s$ and a similarity feature vector $\boldsymbol{v}$, i.e., $\boldsymbol{x} = \langle \boldsymbol{T}_q, \boldsymbol{T}_s, \boldsymbol{v} \rangle$. Given two triples $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$, we define the following kernel:

$$\begin{aligned} K(\boldsymbol{x}^i, \boldsymbol{x}^j) &= K_{\text{TK}}(\boldsymbol{T}_q^i, \boldsymbol{T}_q^j) \\ &+ K_{\text{TK}}(\boldsymbol{T}_s^i, \boldsymbol{T}_s^j) \\ &+ K_{\text{v}}(\boldsymbol{v}^i, \boldsymbol{v}^j), \end{aligned} \quad (1)$$

where K_{TK} computes a structural kernel, e.g., tree kernel, and K_{v} is a kernel over feature vectors, e.g., linear, polynomial, gaussian, etc. Structural kernels can capture the structural representation of a question/answer pair whereas traditional feature vectors can encode some sort of similarity, e.g., lexical, syntactic, semantic, between a question and its candidate answer.

We prefer to split the kernel computation over a question/answer pair into two terms since tree kernels are very efficient and there are no efficient graph kernels that can encode exhaustively all graph fragments. It should be noted that the tree kernel sum does not capture feature pairs. Theoretically, for such purpose, a kernel product should be used. However, our experiments revealed that using the product is actually worse in practice. In contrast, we solve the lack of feature pairing by annotating the trees with relational tags which are supposed to link the question tree fragments with the related fragments from the answer sentence.

Such relational information is very important to improve the quality of the pair representation as well as the implicitly generated features. In the next section, we show simple structural models that we used in our experiments for question and answer pair classification.

2.3 Partial Tree Kernels

The above framework can use any kernel for structural data. We use the Partial Tree Kernel (PTK) (Moschitti, 2006) to compute $K_{\text{TK}}(\cdot,\cdot)$ as it is the most general convolution tree kernel, which at the same time shows rather *good* efficiency. PTK can be effectively applied to both constituency and dependency parse trees. It generalizes the syntactic tree kernel (STK) (Collins and Duffy, 2002), which maps a tree into the space of all possible tree fragments constrained by the rule that sibling nodes cannot be separated. In contrast, the PTK fragments can contain any subset of siblings, i.e., PTK allows for breaking the production rules in syntactic trees. Consequently, PTK generates an extremely rich feature space, which results in higher generalization ability.

3 Relational Structures

This section introduces relational structures designed to encode syntactic and shallow semantic properties of question/answer pairs. We first define a simple to construct shallow syntactic tree representation derived from a shallow parser. Next, we introduce a relational linking scheme based on a plain syntactic matching and further augment it with additional semantic information.

3.1 Shallow syntactic tree

Our shallow tree structure is a two-level syntactic hierarchy built from word lemmas (leaves), part-of-speech tags that organized into chunks identified by a shallow syntactic parser (Fig. 1). We defined a similar structure in (Severyn and Moschitti, 2012) for answer passage reranking, which improved on feature vector baselines.

This simple linguistic representation is suitable for building a rather expressive *answer sentence selection* model. Moreover, the use of a shallow parser is motivated by the need to generate text spans to produce candidate answers required by an *answer extraction* system.

3.2 Tree pairs enriched with relational links

It is important to establish a correspondence between question and answer sentence aligning related concepts from both. We take on a two-level approach, where we first use plain lexical matching to connect common lemmas from the question and its candidate answer sentence. Secondly, we establish semantic links between NEs extracted from the answer sentence and the question focus word, which encodes the expected lexical answer type (LAT). We use the question categories to identify NEs that have

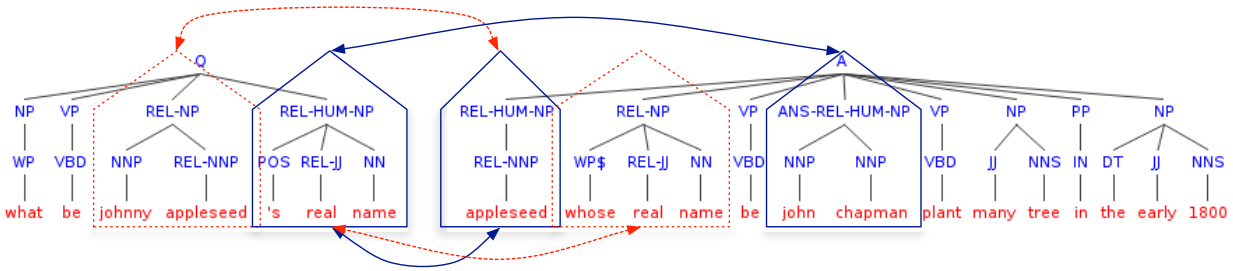


Figure 1: Shallow tree representation of the example q/a pair from Sec. 1. Dashed arrows (red) indicate the tree fragments (red dashed boxes) in the question and its answer sentence linked by the relational REL tag, which is established via syntactic match on the word lemmas. Solid arrows (blue) connect a question focus word *name* with the related named entities of type *Person* corresponding to the question category (HUM) via a relational tag REL-HUM. Additional ANS tag is used to mark chunks containing candidate answer (here the correct answer *John Chapman*).

higher probability to be correct answers following a mapping defined in Table 1.

Next, we briefly introduce our tree kernel-based models for building question focus and category classifiers.

Lexical Answer Type. Question Focus represents a central entity or a property asked by a question (Prager, 2006). It can be used to search for semantically compatible candidate answers, thus greatly reducing the search space (Pinchak, 2006). While several machine learning approaches based on manual features and syntactic structures have been recently explored, e.g. (Quarteroni et al., 2012; Damljanovic et al., 2010; Bunescu and Huang, 2010), we opt for the latter approach where tree kernels handle automatic feature engineering.

To build an automatic Question Focus detector we use a tree kernel approach as follows: we (i) parse each question; (ii) create a set of positive trees by labeling the node exactly covering the focus with FC tag; (iii) build a set of negative trees by labeling any other constituent node with FC; (iii) we train the FC node classifier with tree kernels. At the test time, we try to label each constituent node with FC generating a set of candidate trees. Finally, we select the tree and thus the constituent associated with the highest SVM score.

Question classification. Our question classification model is simpler than before: we use an SVM multi-classifier with tree kernels to automatically extract the question class. To build a multi-class classifier we train a binary SVM for each of the classes and apply a one-vs-all strategy to obtain the predicted class. We use constituency trees as our input representation.

Table 1: Expected Answer Type (EAT) → named entity types.

EAT	Named Entity types
HUM	Person
LOCATION	Location
ENTITY	Organization, Person, Misc
DATE	Date, Time, Number
QUANTITY	Number, Percentage
CURRENCY	Money, Number

Our question taxonomy is derived from the UIUIC dataset (Li and Roth, 2002) which defines 6 coarse and 50 fine grain classes. In particular, our set of question categories is formed by adopting 3 coarse classes: HUM (human), LOC (location), ENTY (entities) and replacing the NUM (numeric) coarse class with 3 fine-grain classes: CURRENCY, DATE, QUANTITY[2]. This set of question categories is sufficient to capture the coarse semantic answer type of the candidate answers found in TREC. Also using fewer question classes results in a more accurate multi-class classifier.

Semantic tagging. Question focus word specifies the lexical answer type capturing the target information need posed by a question, but to make this piece of information effective, the focus word needs to be linked to the target candidate answer. The focus word can be lexically matched with words present in

[2]This class is composed by including all the fine-grain classes from NUMERIC coarse class except for CURRENCY and DATE.

the answer sentence, or the match can be established using semantic information. Clearly, the latter approach is more appealing since it helps to alleviate the lexical gap problem, i.e., it improves the coverage of the näive string matching of words between a question and its answer.

Hence, we propose to exploit a question focus along with the related named entities (according to the mapping from Table 1) of the answer sentence to establish relational links between the tree fragments. In particular, once the question focus and question category are determined, we link the focus word w_{focus} in the question, with all the named entities whose type matches the question class (Table 1). We perform tagging at the chunk level and use a relational tag typed with a question class, e.g., `REL-HUM`. Fig. 1 shows an example q/a pair where the typed relational tag is used in the shallow syntactic tree representation to link the chunk containing the question focus *name* with the named entities of the corresponding type *Person*, i.e., *Appleseed* and *John Chapman*.

4 Answer Sentence Selection and Answer Keyword Extraction

This section describes our approach to (i) answer sentence selection used to select the most promising answer sentences; and (ii) answer extraction which returns the answer keyword (for factoid questions).

4.1 Answer Sentence Selection

We cast the task of answer sentence selection as a classification problem. Considering a supervised learning scenario, we are given a set of questions $\{q_i\}_{i=1}^{N}$ where each question q_i is associated with a list of candidate answer sentences $\{(r_i, s_i)\}_{i=1}^{N}$, with $r_i \in \{-1, +1\}$ indicating if a given candidate answer sentence s_i contains a correct answer $(+1)$ or not (-1). Using this labeled data, our goal is to learn a classifier model to predict if a given pair of a question and an answer sentence is correct or not. We train a binary SVM with tree kernels[3] to train an answer sentence classifier. The prediction scores obtained from a classifier are used to rerank the answer candidates (pointwise reranking), s.t. the sentences that are more likely to contain correct answers will be ranked higher than incorrect candidates. In addition to the structural representation, we augment our model with basic bag-of-word features (unigram and bigrams) computed over lemmas.

[3] disi.unitn.it/moschitti/Tree-Kernel.htm

4.2 Answer Sentence Extraction

The goal of answer extraction is to extract a text span from a given candidate answer sentence. Such span represents a correct answer phrase for a given question. Different from previous work that casts the answer extraction task as a tagging problem and apply a CRF to learn an answer phrase tagger (Yao et al., 2013), we take on a simpler approach using a kernel-based classifier.

In particular, we rely on the shallow tree representation, where text spans identified by a shallow syntactic parser serve as a source of candidate answers. Algorithm 1 specifies the steps to generate training data for our classifier. In particular, for each example representing a triple $\langle a, T_q, T_s \rangle$ composed of the answer a, the question and the answer sentence trees, we generate a set of training examples E with every candidate chunk marked with an `ANS` tag (one at a time). To reduce the number of generated examples for each answer sentence, we only consider `NP` chunks, since other types of chunks, e.g., `VP`, `ADJP`, typically do not contain factoid answers. Finally, an original untagged tree is used to generate a positive example (line 8), when the answer sentence contains a correct answer, and a negative example (line 10), when it does not contain a correct answer.

At the classification time, given a question and a candidate answer sentence, all NP nodes of the sentence are marked with `ANS` (one at a time) as the possible answer, generating a set of tree candidates. Then, such trees are classified (using the kernel from Eq. 1) and the one with the highest score is selected. If no tree is classified as positive example we do not extract any answer.

5 Experiments

We provide the results on two related yet different tasks: answer sentence selection and answer extraction. The goal of the former is to learn a model scoring correct question and answer sentence pairs to bring in the top positions sentences containing the correct answers. Answer extraction derives the cor-

Algorithm 1 Generate training data for answer extraction

for all $\langle a, T_q, T_s \rangle \in \boldsymbol{D}$ **do**
  $E \leftarrow \emptyset$
  for all $chunk \in extract\_chunks(T_s)$ **do**
    if not $chunk ==$ NP **then**
      continue
    $T'_s \leftarrow tagAnswerChunk(T_s, chunk)$
    if $contains\_answer(a, chunk)$ **then**
      label $\leftarrow +1$
    else
      label $\leftarrow -1$
    $e \leftarrow build\_example(T_q, T'_s, label)$
    $E \leftarrow E \cup \{e\}$
return E

rect answer keywords, i.e., a text span such as multi-words or constituents, from a given sentence.

5.1 Semantic Annotation

We briefly describe the experiments of training automatic question category and focus classifiers, which are more extensively described in (Severyn et al., 2013b).

Question Focus detection. We used three datasets for training and evaluating the performance of our focus detector: SeCo-600 (Quarteroni et al., 2012), Mooney GeoQuery (Damljanovic et al., 2010) and the dataset from (Bunescu and Huang, 2010). The SeCo dataset contains 600 questions. The Mooney GeoQuery contains 250 question targeted at geographical information in the U.S. The first two datasets are very domain specific, while the dataset from (Bunescu and Huang, 2010) is more generic containing the first 2,000 questions from the answer type dataset from Li and Roth annotated with focus words. We removed questions with implicit and multiple focuses.

Question Classification. We used the UIUIC dataset (Li and Roth, 2002) which contains 5,952 factoid questions [4] to train a multi-class question classifier.

Table 2 summarizes the results of question focus and category classification.

[4]We excluded questions from TREC to ensure there is no overlap with the data used for testing models trained on TREC QA.

Table 2: Accuracy (%) of focus (FC) and question classifiers (QC) using PTK.

TASK	SET	PTK
FC	MOONEY	80.5
	SECO-600	90.0
	BUNESCU	96.9
QC	UIUIC	85.9
	TREC 11-12	78.1

5.2 Answer Sentence Selection

We used the train and test data from (Wang et al., 2007) to enable direct comparison with previous work on answer sentence selection. The training data is composed by questions drawn from TREC 8-12 while questions from TREC 13 are used for testing. The data provided for training comes as two sets: a small set of 94 questions (TRAIN) that were manually curated for errors[5] and 1,229 questions from the entire TREC 8-12 that contain at least one correct answer sentence (ALL). The latter set represents a more noisy setting, since many answer sentences are marked erroneously as correct as they simply match a regular expression. Table 3 summarizes the data used for training and testing.

Table 4 compares our kernel-based structural model with the previous state-of-the-art systems for answer sentence selection. In particular, we compare with four most recent state of the art answer sentence reranker models (Wang et al., 2007; Heilman and Smith, 2010; Wang and Manning, 2010; Yao et al., 2013), which report their performance on the same questions and candidate sets from TREC 13 as provided by (Wang et al., 2007).

Our simple shallow tree representation (Severyn and Moschitti, 2012) delivers state-of-the-art accuracy largely improving on previous work. Finally, augmenting the structure with semantic linking (Severyn et al., 2013b) yields additional improvement in MAP and MRR. This suggests the utility of using supervised components, e.g., question focus and question category classifiers coupled with NERs, to establish semantic mapping between words in a q/a pair.

[5]In TREC correct answers are identified by regex matching using the provided answer pattern files

Table 3: Summary of TREC data for answer extraction used in (Yao et al., 2013).

data	questions	candidates	correct
TRAIN	94	4718	348
ALL	1229	53417	6410
TEST	89	1517	284

Table 4: Answer sentence reranking on TREC 13.

System	MAP	MRR
Wang et al. (2007)	0.6029	0.6852
Heilman & Smith (2010)	0.6091	0.6917
Wang & Manning (2010)	0.5951	0.6951
Yao et al. (2013)	0.6319	0.7270
+ WN	0.6371	0.7301
shallow tree (S&M, 2012)	0.6485	0.7244
+ semantic tagging	**0.6781**	**0.7358**

It is worth noting that our kernel-based classifier is conceptually simpler than approaches in the previous work, as it relies on the structural kernels, e.g., PTK, to automatically extract salient syntactic patterns relating questions and answers. Our model only includes the most basic feature vector (uni- and bi-grams) and does not rely on external sources such as WordNet.

5.3 Answer Extraction

Our experiments on answer extraction replicate the setting of (Yao et al., 2013), which is the most recent work on answer extraction reporting state-of-the-art results.

Table 5 reports the accuracy of our model in recovering correct answers from a set of candidate answer sentences for a given question. Here the focus is on the ability of an answer extraction system to recuperate as many correct answers as possible from each answer sentence candidate. The set of extracted candidate answers can then be used to select a single best answer, which is the final output of the QA system for factoid questions. Recall (R) encodes the percentage of correct answer sentences for which the system correctly extracts an answer (for TREC 13 there are a total of 284 correct answer sentences), while Precision (P) reflects how many answers extracted by the system are actually correct. Clearly, having a high recall system, allows for correctly answering more questions. On the other hand, a high precision system would attempt to answer less questions (extracting no answers at all) but get them right.

We compare our results to a CRF model of (Yao et al., 2013) augmented with WordNet features (without forced voting) [6]. Unlike the CRF model which obtains higher values of precision, our system acts as a high recall system able to recover most of the answers from the correct answer sentences. Having higher recall is favorable to high precision in answer extraction since producing more correct answers can help in the final voting scheme to come up with a single best answer. To solve the low recall problem of their CRF model, Yao et al. (2013) apply fairly complex outlier resolution techniques to force answer predictions, thus aiming at increasing the number of extracted answers.

To further boost the number of answers produced by our system we exclude negative examples (answer sentences not containing the correct answer) from training, which slightly increases the number of pairs with correctly recovered answers. Nevertheless, it has a substantial effect on the number of questions that can be answered correctly (assuming perfect single best answer selection). Clearly, our system is able to recover a large number of answers from the correct answer sentences, while low precision, i.e., extracting answer candidates from sentences that do not contain a correct answer, can be overcome by further applying various best answer selection strategies, which we explore in the next section.

5.4 Best Answer Selection

Since the final step of the answer extraction module is to select for each question a single best answer from a set of extracted candidate answers, an answer selection scheme is required.

We adopt a simple majority voting strategy, where we aggregate the extracted answers produced by our answer extraction model. Answers sharing similar lemmas (excluding stop words) are grouped together. The prediction scores obtained by the an-

[6] We could not replicate the results obtained in (Yao et al., 2013) with the forced voting strategy. Thus such result is not included in Table 5.

Table 5: Results on answer extraction. P/R - precision and recall; pairs - number of QA pairs with a correctly extracted answer, q - number of questions with at least one correct answer extracted, F1 sets an upper bound on the performance assuming the selected best answer among extracted candidates is always correct. *-marks the setting where we exclude incorrect question answer pairs from training.

set	P	R	pairs	q	F1
Yao et al. (2013)	25.7	23.4	73	33	-
+ WN	26.7	24.3	76	35	-
TRAIN	29.6	64.4	183	58	65.2
TRAIN*	15.7	71.8	204	66	74.1
Yao et al. (2013)	35.2	35.1	100	38	-
+ WN	34.5	34.7	98	38	-
ALL	29.4	74.6	212	69	77.5
ALL*	15.8	76.7	218	73	82.0

Table 6: Results on finding the best answer with voting.

system	set	P	R	F1
Yao et al. (2013)	TRAIN	55.7	43.8	49.1
+ forced		54.5	53.9	54.2
+ WN		55.2	53.9	54.5
this work		**66.2**	**66.2**	**66.2**
Yao et al. (2013)	ALL	67.2	50.6	57.7
+ forced		60.9	59.6	60.2
+ WN		63.6	62.9	63.3
this work		**70.8**	**70.8**	**70.8**

swer extraction classifier are used as votes to decide on the final rank to select the best single answer.

Table 6 shows the results after the majority voting is applied to select a single best answer for each candidate. A rather naïve majority voting scheme already produces satisfactory outcome demonstrating better results than the previous work. Our voting scheme is similar to the one used by (Yao et al., 2013), yet it is much simpler since we do not perform any additional hand tuning to account for the weight of the "forced" votes or take any additional steps to catch additional answers using outlier detection techniques applied in the previous work.

6 Discussion and Error Analysis

There are several sources of errors affecting the final performance of our answer extraction system: (i) chunking, (ii) named entity recognition and semantic linking, (iii) answer extraction, (iv) single best answer selection.

Chunking. Our system uses text spans identified by a chunker to extract answer candidates, which makes it impossible to extract answers that lie outside the chunk boundaries. Nevertheless, we found this to be a minor concern since for 279 out of total 284 candidate sentences from TREC 13 the answers are recoverable within the chunk spans.

Semantic linking. Our structural model relies heavily on the ability of NER to identify the relevant entities in the candidate sentence that can be further linked to the focus word of the question. While our answer extraction model is working on all the NP chunks, the semantic tags from NER serve as a strong cue for the classifier that a given chunk has a high probability of containing an answer. Typical off-the-shelf NER taggers have good precision and low recall, s.t. many entities as potential answers are missed. In this respect, a high recall entity linking system, e.g., linking to wikipedia entities (Ratinov et al., 2011), is required to boost the quality of candidates considered for answer extraction. Finally, improving the accuracy of question and focus classifiers would allow for having more accurate input representations fed to the learning algorithm.

Answer Extraction. Our answer extraction model acts as a high recall system, while it suffers from low precision in extracting answers for many incorrect sentences. Improving the precision without sacrificing the recall would ease the successive task of best answer selection, since having less incorrect answer candidates would result in a better final performance. Introducing additional constraints in the form of semantic tags to allow for better selection of answer candidates could also improve our system.

Best Answer Selection. We apply a naïve majority voting scheme to select a single best answer from a set of extracted answer candidates. This step has a dramatic impact on the final performance of the answer extraction system resulting in a large drop of recall, i.e., from 82.0 to 70.8 before and after voting respectively. Hence, a more involved model, i.e.,

performing joint answer sentence re-ranking and answer extraction, is required to yield a better performance.

7 Related Work

Tree kernel methods have found many applications for the task of answer reranking which are reported in (Moschitti, 2008; Moschitti, 2009; Moschitti and Quarteroni, 2008; Severyn and Moschitti, 2012). However, their methods lack the use of important relational information between a question and a candidate answer, which is essential to learn accurate relational patterns. In this respect, a solution based on enumerating relational links was given in (Zanzotto and Moschitti, 2006; Zanzotto et al., 2009) for the textual entailment task but it is computationally too expensive for the large dataset of QA. A few solutions to overcome computational issues were suggested in (Zanzotto et al., 2010).

In contrast, this paper relies on structures directly encoding the output of question and focus classifiers to connect focus word and good candidate answer keywords (represented by NEs) of the answer passage. This provides more effective relational information, which allows our model to significantly improve on previous rerankers. Additionally, previous work on kernel-based approaches does not target answer extraction.

One of the best models for answer sentence selection has been proposed in (Wang et al., 2007). They use the paradigm of quasi-synchronous grammar to model relations between a question and a candidate answer with syntactic transformations. (Heilman and Smith, 2010) develop an improved Tree Edit Distance (TED) model for learning tree transformations in a q/a pair. They search for a good sequence of tree edit operations using complex and computationally expensive Tree Kernel-based heuristic. (Wang and Manning, 2010) develop a probabilistic model to learn tree-edit operations on dependency parse trees. They cast the problem into the framework of structured output learning with latent variables. The model of (Yao et al., 2013) has reported an improvement over the Wang's et al. (2007) system. It applies linear chain CRFs with features derived from TED and WordNet to automatically learn associations between questions and candidate answers.

Different from previous approaches that use tree-edit information derived from syntactic trees, our kernel-based learning approach also use tree structures but with rather different learning methods, i.e., SVMs and structural kernels, to automatically extract salient syntactic patterns relating questions and answers. In (Severyn et al., 2013c), we have shown that such relational structures encoding input text pairs can be directly used within the kernel learning framework to build state-of-the-art models for predicting semantic textual similarity. Furthermore, semantically enriched relational structures, where automatic have been previously explored for answer passage reranking in (Severyn et al., 2013b; Severyn et al., 2013a). This paper demonstrates that this model also works for building a reranker on the sentence level, and extends the previous work by applying the idea of automatic feature engineering with tree kernels to answer extraction.

8 Conclusions

Our paper demonstrates the effectiveness of handling the input structures representing QA pairs directly vs. using explicit feature vector representations, which typically require substantial feature engineering effort. Our approach relies on a kernel-based learning framework, where structural kernels, e.g., tree kernels, are used to handle automatic feature engineering. It is enough to specify the desired type of structures, e.g., shallow, constituency, dependency trees, representing question and its candidate answer sentences and let the kernel learning framework learn to use discriminative tree fragments for the target task.

An important feature of our approach is that it can effectively combine together different types of syntactic and semantic information, also generated by additional automatic classifiers, e.g., focus and question classifiers. We augment the basic structures with additional relational and semantic information by introducing special tag markers into the tree nodes. Using the structures directly in the kernel learning framework makes it easy to integrate additional relational constraints and semantic information directly in the structures.

The comparison with previous work on a public

benchmark from TREC suggests that our approach is very promising as we can improve the state of the art in both answer selection and extraction by a large margin (up to 22% of relative improvement in F1 for answer extraction). Our approach makes it relatively easy to integrate other sources of semantic information, among which the use of Linked Open Data can be the most promising to enrich the structural representation of q/a pairs.

To achieve state-of-the-art results in answer sentence selection and answer extraction, it is sufficient to provide our model with a suitable tree structure encoding relevant syntactic information, e.g., using shallow, constituency or dependency formalisms. Moreover, additional semantic and relational information can be easily plugged in by marking tree nodes with special tags. We believe this approach greatly eases the task of tedious feature engineering that will find its applications well beyond QA tasks.

Acknowledgements

This research is partially supported by the EU's 7th Framework Program (FP7/2007-2013) (#288024 LIMOSINE project) and an Open Collaborative Research (OCR) award from IBM Research. The first author is supported by the Google Europe Fellowship 2013 award in Machine Learning.

References

Razvan Bunescu and Yunfeng Huang. 2010. Towards a general model of answer typing: Question focus identification. In *CICLing*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *ACL*.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In *LREC*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3).

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

A. Moschitti and S. Quarteroni. 2008. Kernels on Linguistic Structures for Answer Extraction. In *ACL*.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *CIKM*.

Alessandro Moschitti. 2009. Syntactic and semantic kernels for short text pair categorization. In *EACL*.

Christopher Pinchak. 2006. A probabilistic answer type model. In *In EACL*.

John M. Prager. 2006. Open-domain question-answering. *Foundations and Trends in Information Retrieval*, 1(2):91–231.

Silvia Quarteroni, Vincenzo Guerrisi, and Pietro La Torre. 2012. Evaluating multi-focus natural language queries over data services. In *LREC*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Building structures from classifiers for passage reranking. In *CIKM*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning adaptable patterns for passage reranking. In *CoNLL*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013c. Learning semantic textual similarity with structural representations. In *ACL*.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *ACL*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitaura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP*.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

F. M. Zanzotto and A. Moschitti. 2006. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *COLING*.

F. M. Zanzotto, M. Pennacchiotti, and A. Moschitti. 2009. A Machine Learning Approach to Recognizing Textual Entailment. *Natural Language Engineering*, Volume 15 Issue 4, October 2009:551–582.

F. M. Zanzotto, L. Dell'Arciprete, and A. Moschitti. 2010. Efficient graph kernels for textual entailment recognition. *FUNDAMENTA INFORMATICAE*, 2010.

Improving Web Search Ranking by Incorporating Structured Annotation of Queries*

Xiao Ding[1], Zhicheng Dou[2], Bing Qin[1], Ting Liu[1], Ji-Rong Wen[3]

[1]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China

[2]Microsoft Research Asia, Beijing 100190, China

[3]Renmin University of China, Beijing, China

[1]{xding, qinb, tliu}@ir.hit.edu.cn;
[2]zhichdou@microsoft.com; [3]jirong.wen@gmail.com

Abstract

Web users are increasingly looking for structured data, such as lyrics, job, or recipes, using unstructured queries on the web. However, retrieving relevant results from such data is a challenging problem due to the unstructured language of the web queries. In this paper, we propose a method to improve web search ranking by detecting *Structured Annotation* of queries based on top search results. In a structured annotation, the original query is split into different units that are associated with semantic attributes in the corresponding domain. We evaluate our techniques using real world queries and achieve significant improvement.

1 Introduction

Search engines are getting more sophisticated by utilizing information from multiple diverse sources. One such valuable source of information is structured and semi-structured data, which is not very difficult to access, owing to information extraction (Wong et al., 2009; Etzioni et al., 2008; Zhai and Liu 2006) and semantic web efforts.

Driving the web search evolution are the user needs. Users usually have a template in mind when formulating queries to search for information. Agarwal et al., (2010) surveyed a search log of 15 million queries from a commercial search engine. They found that 90% of queries follow certain templates. For example, by issuing the query "taylor swift lyrics falling in love", the users are actually seeking for the lyrics of the song "Mary's Song (oh my my my)" by artist Taylor Swift. The words "falling in love" are actually part of the lyrics they are searching for. However, some top search results are irrelevant to the query, although they contain all the query terms. For example, the first top search result shown in Figure 1(a) does not contain the required lyrics. It just contains the lyrics of another song of Taylor Swift, rather than the song that users are seeking.

A possible way to solve the above ranking problem is to understand the underlying query structure. For example, after recognizing that "taylor swift" is an *artist name* and "falling in love" are part of the *lyrics*, we can improve the ranking by comparing the structured query with the corresponding structured data in documents (shown in Figure 1(b)). Some previous studies investigated how to extract structured information from user queries, such as query segmentation (Bergsma and Wang, 2007). The task of query segmentation is to separate the query words into

*Work was done when the first author was visiting Microsoft Research Asia

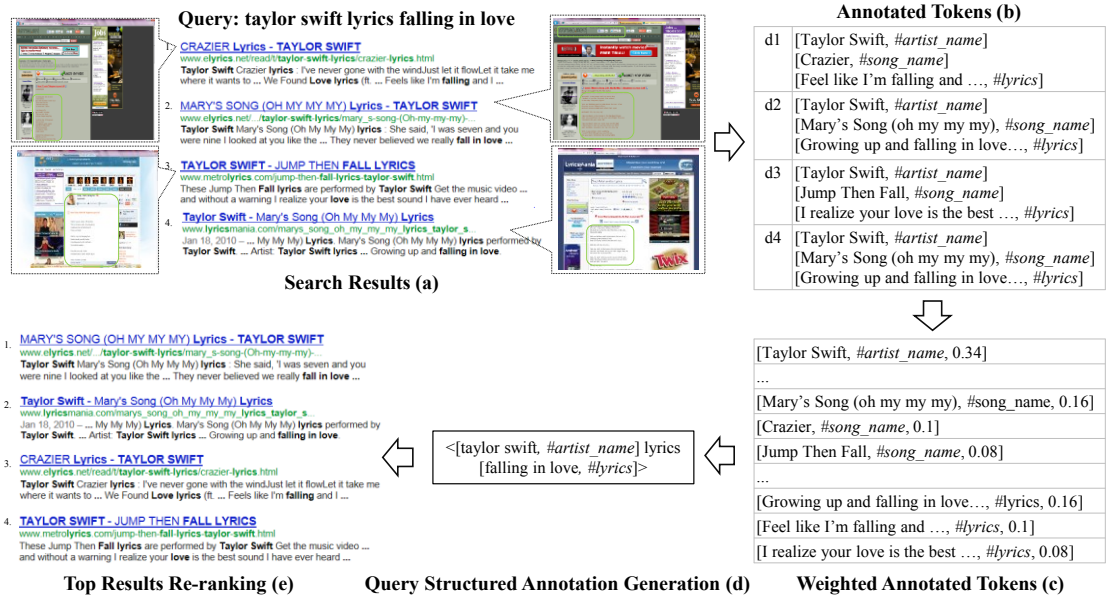


Figure 1. Search results of query "taylor swift lyrics falling in love" and processing pipeline

disjointed segments so that each segment maps to a semantic unit (Li et al., 2011). For example, the segmentation of the query "taylor swift lyrics falling in love" can be "taylor swift | lyrics | falling in love". Since query segmentation cannot tell "talylor swift" is an artist name and "falling in love" are part of lyrics, it is still difficult for us to judge whether each part of the query segmentations matches the right field of the documents or not (such as judge whether "talylor swift" matches the artist name in the document). Recently, a lot of work (Sarkas et al., 2010; Li et al., 2009) proposed the task of structured annotation of queries which aims to detect the structure of the query and assign a specific label to it. However, to our knowledge, the previous methods do not exploit an effective approach for improving web search ranking by incorporating structured annotation of queries.

In this paper, we investigate the possibility of using structured annotation of queries to improve web search ranking. Specifically, we propose a greedy algorithm which uses the structured data (named annotated tokens in Figure 1(b)) extracted from the top search results to annotate the latent structured semantics in web queries. We then compute matching scores between the annotated query and the corresponding structured information contained in documents. The top search results can be re-ranked according to the matching scores. However, it is very difficult to extract structured data from all of the search results. Hence, we propose a relevance feedback based re-ranking model. We use these structured documents whose matching scores are greater than a threshold as feedback documents, to effectively re-rank other search results to bring more relevant and novel information to the user.

Experiments on a large web search dataset from a major commercial search engine show that the F-Measure of structured annotation generated by our approach is as high as 91%. On this dataset, our re-ranking model using the structured annotations significantly outperforms two baselines.

The main contributions of our work include:

1. We propose a novel approach to generate structured annotation of queries based on top search results.
2. Although structured annotation of queries has been studied previously, to the best of our knowledge this is the first paper that attempts to improve web search ranking by incorporating structured annotation of queries.

The rest of this paper is organized as follows. We briefly introduce related work in Section 2. Section 3 presents our method for generating structured annotation of queries. We then propose two novel re-ranking models based on structured annotation in Section 4. Section 5 introduces the data used in this paper. We report experimental results in Section 6. Finally we conclude the work in Section 7.

2 Related Work

There is a great deal of prior research that identifies query structured information. We summarize this research according to their different approaches.

2.1 Structured Annotation of Queries

Recently, a lot of work has been done on understanding query structure (Sarkas et al., 2010; Li et al., 2009; Bendersky et al., 2010). One important method is structured annotation of queries which aims to detect the structure of the query and assign a specific label to it. Li et al., (2009) proposed web query tagging and its goal is to assign to each query term a specified category, roughly corresponding to a list of attributes. A semi-supervised Conditional Random Field (CRF) is used to capture dependencies between query words and to identify the most likely joint assignment of words to "categories." Comparing with previous work, the advantages of our approach are on the following aspects. **First**, we generate structured annotation of queries based on top search results, not some global knowledge base or query logs. **Second**, they mainly focus on the method of generating structured annotation of queries, rather than leverage the generated query structures to improve web search rankings. In this paper, we not only offer a novel solution for generating structured annotation of queries, but also propose a re-ranking approach to improve Web search based on structured annotation of queries. Bendersky et al., (2011) also used top search results to generate structured annotation of queries. However, the annotations in their definition are capitalization, POS tags, and segmentation indicators, which are different from ours.

2.2 Query Template Generation

The concept of query template has been discussed in a few recent papers (Agarwal et al., 2010; Pasca 2011; Liu et al., 2011; Szpektor et al., 2011). A query template is a sequence of terms, where each term could be a word or an attribute. For example, <#artist_name lyrics #lyrics> is a query template, "#artist_name" and "#lyrics" are attributes, and "lyrics" is a word. Structured annotation of queries is different from query template, as a query template can instantiate multiple queries while a structured annotation only serves for a specific query. Unlike query template, our work is ranking-oriented. We aim to automatically annotate query structure based on top search results, and further use these structured annotations to re-rank top search results for improving search performance.

Table 1. Example domain schemas

Domain	Schema	Example structured annotations
lyrics	*#artist_name #song_name #lyrics*	<lyrics of [hey jude, *#song_name*] [beatles, *#artist_name*]>
job	*#category #location*	<[teacher, *#category*] job in [America, *#location*]>
recipe	*#directions #ingredients*	<[baking, # *directions*] [bread, # *ingredients*] recipe>

2.3 Query Segmentation

The task of query segmentation is to separate the query words into disjointed segments so that each segment maps to a semantic unit (Li et al., 2011). Query segmentation techniques have been well studied in recent literature (Tan and Peng, 2008; Yu and Shi, 2009). However, structured annotation of queries cannot only separate the query words into disjoint segments but can also assign each segment a semantic label which can help the search engine to judge whether each part of query segmentation matches the right field of the documents or not.

2.4 Entity Search

The problem of entity search has received a great deal of attention in recent years (Guo et al., 2009; Bron et al., 2010; Cheng et al., 2007). Its goal is to answer information needs that focus on entities. The problem of structured annotation of queries is related to entity search because for some queries, structured annotation items are entities or attributes. Some existing entity search approaches also exploit knowledge from the structure of webpages (Zhao et al., 2005). Annotating query structured information differs from entity search in the following aspects. **First**, structured annotation based ranking is applicable for all queries, rather than just entity related queries. **Second**, the result of an entity search is usually a list of entities, their attributes, and associated homepages, whereas our work uses the structured information from webpages to annotate query structured information and further leverage structured annotation of queries to re-rank top search results.

3 Structured Annotation of Queries

3.1 Problem Definition

We start our discussion by defining some basic concepts. A *token* is defined as a sequence of words including space, i.e., one or more words. For example, the bigram "taylor swift" can be a single token. As our objective is to find structured annotation of queries in a specific domain, we begin with a definition of domain schema.

Definition 1 (Domain Schema): For a given domain of interest, the domain schema is the set of attributes. We denote the domain schema as $A = \{a_1, a_2, \cdots, a_n\}$, where each a_i is the name of an *attribute* of the domain. Sample domain schemas are shown in Table 1. In contrast to previous methods (Agarwal et al., 2010), our definition of domain schema does not need attribute values. For the sake of simplicity, this paper assumes that attributes in domain schema are available. However, it is not difficult to pre-specify attributes in a specific domain.

Definition 2 (Annotated Token): An *annotated token* in a specific domain is a pair $[v, a]$, where v is a token and a is a corresponding *attribute* for v in this domain. [hey jude, *#song_name*] is an example of an annotated token for the "lyrics" domain shown in Table 1. The words "hey jude" comprise a token, and its corresponding attribute name is *#song_name*. If a token does not have any corresponding attributes, we denote it as *free token*.

Definition 3 (Structured Annotation): A *structured annotation p* is a sequence of terms $< s_1, s_2, \cdots, s_k >$, where each s_i could be a free token or an annotated token, and at least one of the terms is an annotated token, i.e., $\exists i \in [1, k]$ for which s_i is an annotated token.

Given the schema for the domain "lyrics", <[taylor swift, *#artist_name*] lyrics [falling in love, *#lyrics*]> is a possible structured annotation for the query "taylor swift lyrics falling in love". In this annotation, [taylor swift, *#artist_name*] and [falling in love, *#lyrics*] are two annotated tokens. The word "lyrics" is a free token.

Intuitively, a structured annotation corresponds to an interpretation of the query as a request for some structured information from documents. The set of annotated tokens expresses the information need of the documents that have been requested. The free tokens may provide more diverse information. Annotated tokens and free tokens together cover all query terms, reflecting the complete user intent of the query.

Algorithm 1: Query Structured Annotation Generation

Input: a list of weighted annotated tokens $T = \{t_1, \ldots, t_m\}$;
a query $q =$ "$w_1, \ldots, w_n$" where $w_i \in$ W;
a pre-defined threshold score δ.
Output: a query structured annotation $p = <s_1, \ldots, s_k>$.

1: Set $p = q = \{s_1, \ldots, s_n\}$, where $s_i = w_i$
2: for u = 1 to *T.size* do
3: compute $Match(p, t_u)$
$= Match(p, t_u.v)$
$= t_u.w \times max_{0 \le i < j \le n} Sim(p_{ij}, t_u.v)$,
where $p_{ij} = s_i, \ldots, s_j$, s.t. $s_l \in$ W for $l \in [i, j]$. *//p_{ij} is just in the remaining query words*
4: end for
5: find the maximum matching t_u with
$t_{max} = argmax_{1 \le u \le m} Match(p, t_u)$
6: if $Match(p, t_{max}) > \delta$ then
7: replace $s_i, \ldots, s_j$ in p with $[s_i, \ldots, s_j, t_{max}.a]$
8: remove t_{max} from T
9: n ← n – (j - i)
10: go to step 2
11: else
12: return p
13: end if

3.2 Generating Structured Annotation

In this paper, given a domain schema A, we generate structured annotation for a query q based on the top search results of q. We propose using top search results, rather than some global knowledge base or query logs, because:

(1) Top search results have been proven to be a successful technique for query explanation (Bendersky et al., 2010).

(2) We have observed that in most cases, a reasonable percentage of the top search results are relevant to the query. By aggregating structured information from the top search results, we can get more query-dependent annotated tokens than using global data sources which may contain more noise and outdated.

(3) Our goal for generating structured annotation is to improve the ranking quality of queries. Using top search results enables simultaneous and consistent detection of structured information from documents and queries.

As mentioned in Section 3.1, we generate structured annotation of queries based on annotated tokens, which are actually structured data (shown in Figure 1(b)) embedded in web documents. In this paper, we assume that the annotated tokens are

available and we mainly focus on how to use these annotated tokens from top search results to generate structured annotation of queries. The approach is comprised of two parts, one for weighting annotated tokens and the other for generating structured annotation of queries based on the weighted annotated tokens.

Weighting: As shown in Figure 1, annotated tokens extracted from top results may be inconsistent, and hence some of the extracted annotated tokens are less useful or even useless for generating structured annotation.

We assume that a better annotated token should be supported by more top results; while a worse annotated token may appear in fewer results. Hence we aggregate all the annotated tokens extracted from top search results, and evaluate the importance of each unique one by a ranking-aware voting model as follows. For an annotated token [*v*, *a*], its weight *w* is defined as:

$$w = \frac{1}{N}\sum_{1\le j\le N} w_j \quad (1)$$

where w_j is a voting from document d_j, and

$$w_j = \begin{cases} \frac{N-j+1}{N}, & \text{if } [v,a] \in d_j \\ 0, & \text{else} \end{cases}$$

Here, *N* is the number of top search results and *j* is the ranking position of document d_j. We then generate a weighted annotated token [*v*, *a*, *w*] for each original unique token [*v*, *a*].

Generating: The process by which we map a query *q* to *Structured Annotation* is shown in Algorithm 1. The algorithm takes as input a list of weighted annotated tokens and the query *q*, and outputs the structured annotation of the query *q*. The algorithm first partitions the query *q* by comparing each sub-sequence of the query with all the weighted annotated tokens, and find the maximum matching annotated token (line 1 to line 5). Then, if the degree of match is greater than the threshold δ which is a pre-defined threshold score for fuzzy string matching, the query substring will be assigned the attribute label of the maximum matching annotated token (line 6 to line 8). The algorithm stops when all the weighted annotated tokens have been scanned, and outputs the structured annotation of the query.

Note that in some cases, the query may fail to exactly match with the annotated tokens, due to spelling errors, acronyms or abbreviations in users' queries. For example, in the query "broken and **beatuful** lyrics", "broken and **beatuful**" is a misspelling of "broken and **beautiful**." We adopt a fuzzy string matching function for comparing a sub-sequence string *s* with a token *v*:

$$Sim(s,v) = 1 - \frac{EditDistance(s,v)}{\max(|s|,|v|)} \quad (2)$$

where *EditDistance*(*s*, *v*) measures the edit distances of two strings, |*s*| is the length of string *s* and |*v*| is the length of string *v*.

4 Ranking with Structured Annotation

Given a domain schema $A = \{a_1, a_2, \cdots, a_n\}$, and a query *q*, suppose that $p = < s_1, s_2, \ldots, s_k >$ is the structured annotation for query *q* obtained using the method introduced in the above sections. *p* can better reflects the user's real search intent than the original *q*, as it presents the structured semantic information needed instead of a simple word string. Therefore, a document d_i can better satisfy a user's information need if it contains corresponding structured semantic information in *p*. Suppose that T_i is the set of annotated tokens extracted from document d_i, we compute a re-ranking score, denoted by *RScore*, for document d_i as follows:

$$\begin{aligned} RScore(q, d_i) &= Match(q, d_i) \\ &= Match(p, T_i) \\ &= \sum_{1\le j\le k}\sum_{t\in T_i} Match(s_j, t) \end{aligned}$$

where

$$Match(s_j, t) = \begin{cases} Sim(s_j.v_j, t.v), & \text{if } s_j.a_j = t.a \\ 0, & \text{else} \end{cases} \quad (3)$$

where s_j is an annotated token in *p* and *t* is an annotated token in d_i. We use Equation (2) to compute the similarity between values in query annotated tokens and values in document annotated tokens. We propose two re-ranking models, namely the conservative re-ranking model, to re-rank top results based on *RScore* and relevance feedback based re-ranking model.

4.1 Conservative Re-ranking Model

A nature way to re-rank top search results is according to their *RScore*. However, we fail to obtain annotated tokens from some retrieved documents, and hence the *RScore* of these documents are not available. In the conservative re-ranking model, we only re-rank search results that have an *RScore*. For example, suppose there are five retrieved documents {d_1, d_2, d_3, d_4, d_5} for query *q*, we can extract structured information from document d_3 and d_4 and $RScore(q, d_4) > RScore(q, d_3)$. Note that we cannot obtain

structured information from d_1, d_2, and d_5. In the conservative re-ranking method, d_1, d_2, and d_5 retain their original positions; while d_3 and d_4 will be re-ranked according to their *RScore*. Therefore, the final ranking generated by our conservative re-ranking model should be $\{d_1, d_2, d_4, d_3, d_5\}$, in which the documents are re-ranked among the affected positions.

There is also useful information in the documents without structured data, such as community question answering websites. However, in the conservative re-ranking model they will not be re-ranked. This may hurt the performance of our re-ranking model. One reasonable solution is relevance feedback model.

4.2 Relevance Feedback based Re-ranking Model

The disadvantage of the conservative re-ranking model is that it only can re-rank those top search results with structured data. To make up its limitation, we propose a relevance feedback based re-ranking model. The key idea of this model is based on the observation that the search results with the corrected annotated tokens could give implicit feedback information. Hence, we use these structured documents whose *RScore* are greater than a threshold γ (empirically set it as 0.6) as feedback documents, to effectively re-rank other search results to bring more relevant and novel information to the user.

Formally, given a query Q and a document collection C, a retrieval system returns a ranked list of documents D. Let d_i denote the i-th ranked document in the ranked list. Our goal is to study how to use these feedback documents, $J \subseteq \{d_1, \ldots, d_k\}$, to effectively re-rank the other r search results: $U \subseteq \{d_{k+1}, \ldots, d_{k+r}\}$. A general formula of relevance feedback model (Salton et al, 1990) R is as follows:

$$R(Q') = (1 - \alpha)L_q(\mathrm{Q}) + \alpha L_d(\mathrm{J}) \qquad (4)$$

where $\alpha \in [0, 1]$ is the feedback coefficient, and L_q and L_d are two models that map a query and a set of relevant documents, respectively, into some comparable representations. For example, they can be represented as vectors of weighted terms or language models.

In this paper, we explore the problem in the language model framework, particularly the KL-divergence retrieval model and mixture-model feedback method (Zhai and Lafferty, 2001), mainly because language models deliver state-of-the-art retrieval performance and the mixture-model based feedback is one of the most effective feedback techniques which outperforms Rocchio feedback.

4.2.1 The KL-Divergence Retrieval Model

The KL-divergence retrieval model was introduced in Lafferty and Zhai, (2001) as a special case of the risk minimization retrieval framework and can support feedback more naturally. In this model, queries and documents are represented by unigram language models. Assuming that these language models can be appropriately estimated, KL-divergence retrieval model measures the relevance value of a document D with respect to a query Q by computing the negative Kullback-Leibler divergence between the query language model θ_Q and the document language model θ_D as follows:

$$S(Q, D) = -D(\theta_Q || \theta_D) = -\sum_{w \in V} p(w|\theta_Q) log \frac{p(w|\theta_Q)}{p(w|\theta_D)} \qquad (5)$$

where V is the set of words in our vocabulary. Intuitively, the retrieval performance of the KL-divergence relies on the estimation of the document model θ_D and the query model θ_Q.

For the set of k relevant documents, the document model θ_D is estimated as $p(\mathrm{w}|\theta_D) = \frac{1}{k}\sum_{i=1}^{k} \frac{c(w,r_i)}{|r_i|}$, where $c(w, r_i)$ is the count of word w in the i-th relevant document, and $|r_i|$ is the total number of words in that document. The document model θ_D needs to be smoothed and an effective method is Dirichlet smoothing (Zhai et al., 2001).

The query model intuitively captures what the user is interested in, and thus would affect retrieval performance. With feedback documents, θ_Q is estimated by the mixture-model feedback method.

4.2.2 The Mixture Model Feedback Method

As the problem definition in Equation (4), the query model can be estimated by the original query model $p(w|\theta_Q) = \frac{c(w,Q)}{|Q|}$ (where $c(w,Q)$ is the count of word w in the query Q, and $|Q|$ is the total number of words in the query) and the feedback document model. Zhai and Lafferty, (2001) proposed a mixture model feedback method to estimate the feedback document model. More specifically, the model assumes that the feedback documents can be generated by a background language model $p(w|C)$ estimated using the whole collection and an unknown topic language model

Table 2. Domain queries used in our experiment

Domain	Containing Keyword	Queries	Structured Annotation%
lyrics	"lyrics"	196	95%
job	"job"	124	92%
recipe	"recipe"	76	93%

Table 3. Quality of Structured Annotation. All the improvements are significant ($p < 0.05$)

Domain	Method	Precision	Recall	F-Measure
lyrics	Baseline	90.06%	84.92%	87.41%
	Our	95.45%	89.83%	92.55%
job	Baseline	89.62%	80.14%	84.62%
	Our	95.31%	84.93%	89.82%
recipe	Baseline	83.96%	84.23%	84.09%
	Our	89.68%	88.44%	89.06%
All	Baseline	87.88%	83.10%	85.42%
	Our	**93.61%**	**88.45%**	**90.96%**

θ_F to be estimated. Formally, let $F \subset C$ be a set of feedback documents. In this paper, F is comprised of documents that *RScore* are greater than γ. The log-likelihood function of the mixture model is:

$$log(F|\theta_F) = \sum_{D \in F} \sum_{w \in V} c(w, D) \log[(1-\lambda)p(w|\theta_F) + \lambda p(w|C)] \quad (6)$$

where $\lambda \in [0,1)$ is a mixture noise parameter which controls the weight of the background model. Given a fixed λ, a standard EM algorithm can then be used to estimate $p(w|\theta_F)$, which is then interpolated with the original query model $p(w|Q)$ to obtain an improved estimation of the query model:

$$p(w|\theta_Q) = (1-\alpha)p(w|Q) + \alpha p(w|\theta_F) \quad (7)$$

where α is the feedback coefficient.

5 Data

We used a dataset composed of 12,396 queries randomly sampled from query logs of a search engine. For each query, we retrieved its top 100 results from a commercial search engine. The documents were judged by human editors. A five-grade (from 0 to 4 meaning from bad to perfect) relevance rating was assigned for each document.

We used a proprietary query domain classifier to identify queries in three domains, namely "lyrics," "recipe," and "job," from the dataset. The statistics about these domains are shown in Table 2. To investigate how many queries may potentially have structured annotations, we manually created structured annotations for these queries. The last column of Table 2 shows the percentage of queries

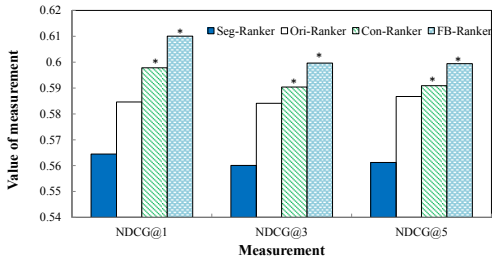


Figure 2. Ranking Quality (* indicates significant improvement)

that have structured annotations created by annotators. We found that for each domain, there was on average more than 90% of queries identified by us that had a certain structured annotation. This indicates that a large percentage of these queries contain structured information, as we expected.

6 Experimental Results

In this section, we present the structured annotation of queries and further re-rank the top search results for the three domains introduced in Section 5. We used the ranking returned by a commercial search engine as our one of the **Baselines**. Note that as the baseline already uses a large number of ranking signals, it is very difficult to improve it any further.

We evaluate the ranking quality using the widely used Normalized Discounted Cumulative Gain measure (NDCG) (Javelin and Kekalainen., 2000). We use the same configuration for NDCG as (Burges et al. 2005). More specifically, for a given query q, the NDCG@K is computed as:

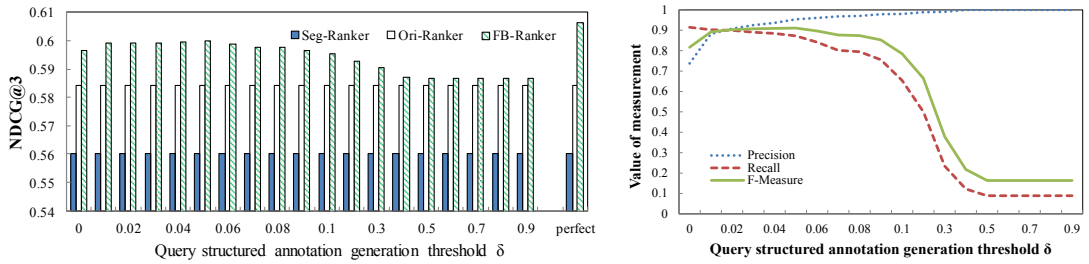
$$N_q = \frac{1}{M_q} \frac{\sum_{j=1}^{K} (2^{r(j)} - 1)}{\log(1+j)} \quad (4)$$

M_q is a normalization constant (the ideal NDCG) so that a perfect ordering would obtain an NDCG of 1; and $r(j)$ is the rating score of the *j-th* document in the ranking list.

6.1 Overall Results

6.1.1 Quality of Structured Annotation of Queries

We generated the structured annotation of queries based on the top 10 search results and used $\delta = 0.04$ for Algorithm 1. We used several existing metrics, P (Precision), R (Recall), and F-Measure to evaluate the quality of the structured annotation. As a query structured annotation may contain more than one annotated token, we concluded that the



(a) Quality of re-ranking (b) Quality of query structured annotation

Figure 3. Quality of re-ranking and quality of query structured annotation with different number of search results

Table 4. Detailed ranking results on three domains. All the improvements are significant ($p < 0.05$)

Domain	Ranking Method	NDCG@1	NDCG@3	NDCG@5
lyrics	Seg-Ranker	0.572	0.574	0.575
	Ori-Ranker	0.621	0.628	0.636
	FB-Ranker	**0.637**	**0.639**	**0.647**
recipe	Seg-Ranker	0.629	0.631	0.634
	Ori-Ranker	0.678	0.687	0.696
	FB-Ranker	**0.707**	**0.704**	**0.709**
job	Seg-Ranker	0.438	0.413	0.408
	Ori-Ranker	0.470	0.453	0.442
	FB-Ranker	**0.504**	**0.474**	**0.459**

annotation was correct only if the entire annotation was **completely the same** as the annotation labeled by annotators. Otherwise we treated the structured annotation as incorrect. Experimental results for the three domains are shown in Table 3. We compare our approach with Xiao Li, (2010) (denoted as baseline), on the dataset described in Section 5. They labeled the semantic structure of noun phrase queries based on semi-Markov CRFs. Our approach achieves better performance than the baseline (about 5.5% significant improvement on F-Measure). This indicates that the approach of generating structured annotation based on the top search results is more effective. With the high-quality structured annotation of queries in hand, it may be possible to obtain better ranking results using our proposed re-ranking models.

6.1.2 Re-ranking Result

We used the models introduced in Section 4 to re-rank the top 10 search results, based on structured annotation of queries and annotated tokens.

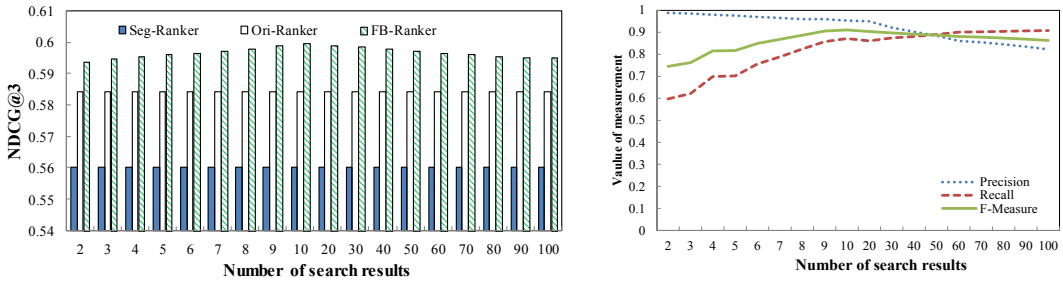
Recall that our goal is to quantify the effectiveness of structured annotation of queries for real web search. One dimension is to compare with the original search results of a commercial search engine (denoted as **Ori-Ranker**). The other is to compare with the query segmentation based re-ranking model (denoted as **Seg-Ranker**; Li et al., 2011) which tries to improve web search ranking by incorporating query segmentation. Li et al., (2011) incorporated query segmentation in the BM25, unigram language model and bigram language model retrieval framework, and bigram language model achieved the best performance. In this paper, Seg-Ranker integrates bigram language model with query segmentation.

The ranking results of these models are shown in Figure 2. This figure shows that all our two rankers significantly outperform the Ori-Ranker– the original search results of a commercial search engine. This means that using high-quality structured annotation does help better understanding of user intent. By comparing these structured annotations and the annotated tokens in documents, we can re-rank the more relevant results higher and yield better ranking quality.

Figure 2 also suggests that structured annotation based re-ranking models outperform query segmentation based re-ranking model. This is mainly because structured annotation can not only separate the query words into disjoint segments but can also assign each segment a semantic label. Taking full advantage of the semantic label can lead to better ranking performance.

Furthermore, Figure 2 shows that FB-Ranker outperforms Con-Ranker. The main reason is that in Con-Ranker, we can only reasonably re-rank the search results with structured data. However, in FB-Ranker we can not only re-rank the structured search results but also can re-rank other documents by incorporating implicit information from those structured documents.

On average, FB-Ranker achieves the best ranking performance. Table 4 shows more detailed



(a) Quality of re-ranking (b) Quality of query structured annotation

Figure 4. Quality of re-ranking and quality of query structured annotation with different number of search results

results for the three selected domains. This table shows that FB-Ranker consistently outperforms the two baseline rankers on these domains. In the remaining part of this paper, we will only report the results for this ranker, due to space limitations. Table 4 also indicates that we can get robust ranking improvement in different domains, and we will consider applying it to more domains.

6.2 Experiment with Different Thresholds of Query Structured Annotation Algorithm

As introduced in Algorithm 1, we pre-defined a threshold δ for fuzzy string matching. We evaluated the quality of re-ranking and query structured annotation with different settings for δ. The results are shown in Figure 3. We found that:

(1) When we use $\delta = 0$, which means that the structured annotations can be generated no matter how small the similarity between the query string and a weighted annotated token is, we can get a significant NDCG@3 gain of 2.15%. Figure 3(b) shows that the precision of the structured annotation is lowest when $\delta = 0$. However, the precision is still as high as 0.7375, and the highest recall is obtained in this case. This means that the quality of the generated structured annotations is still reasonable, and hence we can get a ranking improvement when $\delta = 0$, as shown in Figure 3(a).

(2) Figure 3(a) suggests that the quality of re-ranking increases when the threshold δ increases from 0 to 0.05. It then decreases when δ increases from 0.06 to 0.5. Comparing these two figures shows that the trend of re-ranking performance adheres to the quality of the structured annotation. The settings for δ dramatically affect the recall and precision of the structured annotation; and hence the ranking quality is impacted. The larger δ is, the lower the recall of the structured annotation is.

(3) Since the re-ranking performance dramatically changes along with the quality of the structured annotation, we conducted a re-ranking experiment with perfect structured annotations (F-Measure equal to 1.0). Perfect structured annotations mean the annotations created by annotators as introduced in Section 5. The results are shown in the last bar of Figure 3(a). We did not find a large space for ranking improvement. The NDCG@3 when using perfect structured annotations was 0.606, which is just slightly better than our best result (yield when δ=0.05). It indicates that our structured annotation generation algorithm is already quite effective.

(4) Figure 3(a) shows that our approach outperforms the two baseline approaches with most settings for δ. This indicates that our approach is relatively stable with different settings for δ.

6.3 Experiment with Number of Top Search Results

The above experiments are conducted based on the top 10 search results. In this section, by adjusting the number of top search results, ranging from 2 to 100, we investigate whether the quality of structured annotation of queries and the performance of re-ranking are affected by the quantity of search results. The results shown in Figure 4 indicate that the number of search results does affect the quality of structured annotation of queries and the performance of re-ranking. Structured annotations of queries become better when more search results are used from 2 to 20. This is because more search results cover more websites in our domain list, and hence can generate more annotated tokens. More results also provide more evidence for voting the importance of

annotated tokens, and hence can improve the quality of structured annotation of queries.

In addition, we also found that structured annotation of queries become worse when too many lower ranked results are used (e.g, using results ranked lower than 20). This is because the lower ranked results are less relevant than the higher ranked results. They may contain more irrelevant or noisy annotated tokens than higher ranked documents; and hence using them may harm the precision of the structured annotations. Figure 4 also indicates that the quality of ranking and the accuracy of structured annotations are correlated.

7 Conclusions

In this paper, we studied the problem of improving web search ranking by incorporating structured annotation of queries. We proposed a systematic solution, first to generate structured annotation of queries based on top search results, and then launching two structured annotation based re-ranking models. We performed a large-scale evaluation over 12,396 queries from a major search engine. The experiment results show that the F-Measure of query structured annotation generated by our approach is as high as 91%. In the same dataset, our structured annotation based re-ranking model significantly outperforms the original ranker – the ranking of a major search engine, with improvements 5.2%.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) via grant 61273321, 61133012 and the Nation-al 863 Leading Technology Research Project via grant 2012AA011102.

References

G. Agarwal, G. Kabra, and K. C.-C. Chang. Towards rich query interpretation: walking back and forth for mining query templates. In *Proc. of WWW '10*.

M. Bendersky, W. Bruce Croft and D. A. Smith. Joint Annotation of Search Queries, In *Proc. of ACL-HLT 2011*.

M. Bendersky, W. Bruce Croft and D. A. Smith. Structural Annotation of Search Queries Using Pseudo-Relevance Feedback, In *Proc. Of CIKM 2010.*

S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *Proceedings of EMNLP-CoNLL'07*.

M. Bron, K. Balog, and M. de Rijke. Ranking related entities: components and analyses. In *Proc. of CIKM '10*.

C. Buckley. Automatic query expansion using SMART. In*Proc. of TREC-3, pages 69–80, 1995*.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank usinggradient descent. In *Proceedings of ICML '05*.

T. Cheng, X. Yan, and K. C.-C. Chang. Supporting entity search: a large-scale prototype search engine. In *Proc. of SIGMOD '07*.

O. Etzioni, M. Banko, S. Soderland, and D.S. Weld, (2008). Open Information Extraction from the Web, *Communications of the ACM*, 51(12): 68-74.

J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proc. Of SIGIR' 2009.*

K. Jarvelin and J. Kekalainen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR '00*.

J. Lafferty and C. Zhai, Document language models, query models, and risk minimization for information retrieval, In *Proceedings of SIGIR'01*, pages 111-119, 2001.

V. Lavrenko and W. B. Croft. Relevance based language models. In *Proc. of SIGIR, pages 120–127, 2001*.

Y. Li, BJP. Hsu, CX. Zhai and K. Wang. Unsupervised Query Segmentation Using Clickthrough for Information Retrieval. In *Proc. of SIGIR'11*.

X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proc. of SIGIR'09*.

Y. Liu, X. Ni, J-T. Sun, Z. Chen. Unsupervised Transactional Query Classification Based on Webpage Form Understanding. In *Proc. of* CIKM *'11*.

Y. Liu, M. Zhang, L. Ru, and S. Ma. Automatic query type identification based on click-through information. In *LNCS*, 2006.

M. Pasca. Asking What No One Has Asked Before: Using Phrase Similarities To Generate Synthetic Web Search Queries. In *Proc. of CIKM '11*.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41(4):288-297, 1990.

N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of web queries. In *Proc. of SIGMOD'*10.

I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *Proc. of WWW '11*

B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In WWW'08.

T.-L. Wong, W. Lam, and B. Chen. Mining employment market via text block detection and adaptive cross-domain information extraction. In *Proc. SIGIR*, pages 283–290, 2009.

X. Yu and H. Shi. Query segmentation using conditional random fields. In *Proceedings of* KEYS *'*09.

C. Zhai and J. Lafferty, Model-based feedback in the language modeling approach to information retrieval , In *Proceedings of CIKM'01*, pages 403-410, 2001.

C. Zhai and J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, In *Proceedings of SIGIR'01*, pages 334-342, 2001.

Y. Zhai and B. Liu. Structured data extraction from the Web based on partial tree alignment. *IEEE Trans. Knowl. Data Eng.*, 18(12):1614−1628, Dec. 2006.

H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *Proceedings of WWW '05.*

S. Zheng, R. Song, J.-R. Wen, and D. Wu. Joint optimization of wrapper generation and template detection. In *Proc. of SIGKDD'*07.

Building Specialized Bilingual Lexicons Using Large-Scale Background Knowledge

Dhouha Bouamor[1], **Adrian Popescu**[1], **Nasredine Semmar**[1], **Pierre Zweigenbaum**[2]
[1] CEA, LIST, Vision and Content Engineering Laboratory, 91191
Gif-sur-Yvette CEDEX, France; firstname.lastname@cea.fr
[2]LIMSI-CNRS, F-91403 Orsay CEDEX, France; pz@limsi.fr

Abstract

Bilingual lexicons are central components of machine translation and cross-lingual information retrieval systems. Their manual construction requires strong expertise in both languages involved and is a costly process. Several automatic methods were proposed as an alternative but they often rely on resources available in a limited number of languages and their performances are still far behind the quality of manual translations. We introduce a novel approach to the creation of specific domain bilingual lexicon that relies on Wikipedia. This massively multilingual encyclopedia makes it possible to create lexicons for a large number of language pairs. Wikipedia is used to extract domains in each language, to link domains between languages and to create generic translation dictionaries. The approach is tested on four specialized domains and is compared to three state of the art approaches using two language pairs: French-English and Romanian-English. The newly introduced method compares favorably to existing methods in all configurations tested.

1 Introduction

The plethora of textual information shared on the Web is strongly multilingual and users' information needs often go well beyond their knowledge of foreign languages. In such cases, efficient machine translation and cross-lingual information retrieval systems are needed. Machine translation already has a decades long history and an array of commercial systems were already deployed, including Google Translate [1] and Systran [2]. However, due to the intrinsic difficulty of the task, a number of related problems remain open, including: the gap between text semantics and statistically derived translations, the scarcity of resources in a large majority of languages and the quality of automatically obtained resources and translations. While the first challenge is general and inherent to any automatic approach, the second and the third can be at least partially addressed by an appropriate exploitation of multilingual resources that are increasingly available on the Web.

In this paper we focus on the automatic creation of domain-specific bilingual lexicons. Such resources play a vital role in Natural Language Processing (NLP) applications that involve different languages. At first, research on lexical extraction has relied on the use of parallel corpora (Och and Ney, 2003). The scarcity of such corpora, in particular for specialized domains and for language pairs not involving English, pushed researchers to investigate the use of comparable corpora (Fung, 1998; Chiao and Zweigenbaum, 2003). These corpora include texts which are not exact translation of each other but share common features such as domain, genre, sampling period, etc.

The basic intuition that underlies bilingual lexicon creation is the *distributional hypothesis* (Harris, 1954) which puts that words with similar meanings occur in similar contexts. In a multilingual formulation, this hypothesis states that the translations of a word are likely to appear in similar lexical environments across languages (Rapp, 1995). The *standard approach* to bilingual lexicon extraction builds

[1] http://translate.google.com/
[2] http://www.systransoft.com/

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 479–489,
Seattle, Washington, USA, 18-21 October 2013.

on the distributional hypothesis and compares context vectors for each word of the source and target languages. In this approach, the comparison of context vectors is conditioned by the existence of a seed bilingual dictionary. A weakness of the method is that poor results are obtained for language pairs that are not closely related (Ismail and Manandhar, 2010). Another important problem occurs whenever the size of the seed dictionary is small due to ignoring many context words. Conversely, when dictionaries are detailed, ambiguity becomes an important drawback.

We introduce a bilingual lexicon extraction approach that exploits Wikipedia in an innovative manner in order to tackle some of the problems mentioned above. Important advantages of using Wikipedia are:

- The resource is available in hundreds of languages and it is structured as unambiguous concepts (i.e. articles).
- The languages are explicitly linked through concept translations proposed by Wikipedia contributors.
- It covers a large number of domains and is thus potentially useful in order to mine a wide array of specialized lexicons.

Mirroring the advantages, there are a number of challenges associated with the use of Wikipedia:

- The comparability of concept descriptions in different languages is highly variable.
- The translation graph is partial since, when considering any language pair, only a part of the concepts are available in both languages and explicitly connected.
- Domains are unequally covered in Wikipedia (Halavais and Lackaff, 2008) and efficient domain targeting is needed.

The approach introduced in this paper aims to draw on Wikipedia's advantages while appropriately addressing associated challenges. Among the techniques devised to mine Wikipedia content, we hypothesize that an adequate adaptation of Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) is fitted to our application context. ESA was already successfully tested in different NLP tasks, such as word relatedness estimation or text classification, and we modify it to mine specialized domains, to characterize these domains and to link them across languages.

The evaluation of the newly introduced approach is realized on four diversified specialized domains (*Breast Cancer*, *Corporate Finance*, *Wind Energy* and *Mobile Technology*) and for two pairs of languages: French-English and Romanian-English. This choice allows us to study the behavior of different approaches for a pair of languages that are richly represented and for a pair that includes Romanian, a language that has fewer associated resources than French and English. Experimental results show that the newly introduced approach outperforms the three state of the art methods that were implemented for comparison.

2 Related Work

In this section, we first give a review of the standard approach and then introduce methods that build upon it. Finally, we discuss works that rely on Explicit Semantic Analysis to solve other NLP tasks.

2.1 Standard Approach (SA)

Most previous approaches that address bilingual lexicon extraction from comparable corpora are based on the standard approach (Fung, 1998; Chiao and Zweigenbaum, 2002; Laroche and Langlais, 2010). This approach is composed of three main steps:

1. **Building context vectors**: Vectors are first extracted by identifying the words that appear around the term to be translated W_{cand} in a window of n words. Generally, association measures such as the mutual information (Morin and Daille, 2006), the log-likelihood (Morin and Prochasson, 2011) or the Discounted Odds-Ratio (Laroche and Langlais, 2010) are employed to shape the context vectors.
2. **Translation of context vectors**: To enable the comparison of source and target vectors, source vectors are translated intoto the target language by using a seed bilingual dictionary. Whenever several translations of a context word exist,

all translation variants are taken into account. Words not included in the seed dictionary are simply ignored.

3. **Comparison of source and target vectors**: Given W_{cand}, its automatically translated context vector is compared to the context vectors of all possible translations from the target language. Most often, the cosine similarity is used to rank translation candidates but alternative metrics, including the weighted Jaccard index (Prochasson et al., 2009) and the city-block distance (Rapp, 1999), were studied.

2.2 Improvements of the Standard Approach

Most of the improvements of the standard approach are based on the observation that the more representative the context vectors of a candidate word are, the better the bilingual lexicon extraction is. At first, additional linguistic resources, such as specialized dictionaries (Chiao and Zweigenbaum, 2002) or transliterated words (Prochasson et al., 2009), were combined with the seed dictionary to translate context vectors.

The ambiguities that appear in the seed bilingual dictionary were taken into account more recently. (Morin and Prochasson, 2011) modify the standard approach by weighting the different translations according to their frequency in the target corpus. In (Bouamor et al., 2013), we proposed a method that adds a word sense disambiguation process relying on semantic similarity measurement from WordNet to the standard approach. Given a context vector in the source language, the most probable translation of polysemous words is identified and used for building the corresponding vector in the target language. The most probable translation is identified using the monosemic words that appear in the same lexical environment.

On specialized French-English comparable corpora, this approach outperforms the one proposed in (Morin and Prochasson, 2011), which is itself better than the standard approach. The main weakness of (Bouamor et al., 2013) is that the approach relies on WordNet and its application depends on the existence of this resource in the target language. Also, the method is highly dependent on the coverage of the seed bilingual dictionary.

2.3 Explicit Semantic Analysis

Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2007) is a method that maps textual documents onto a structured semantic space using classical text indexing schemes such as TF-IDF. Examples of semantic spaces used include Wikipedia or the Open Directory Project but, due to superior performances, Wikipedia is most frequently used. In the original evaluation, ESA outperformed state of the art methods in a word relatedness estimation task.

Subsequently, ESA was successfully exploited in other NLP tasks and in information retrieval. Radinsky and al. (2011) added a temporal dimension to word vectors and showed that this addition improves the results of word relatedness estimation. (Hassan and Mihalcea, 2011) introduced Salient Semantic Analysis (SSA), a development of ESA that relies on the detection of salient concepts prior to mapping words to concepts. SSA and the original ESA implementation were tested on several word relatedness datasets and results were mixed. Improvements were obtained for text classification when comparing SSA with the authors' in-house representation of the method. ESA has weak language dependence and was already deployed in multilingual contexts. (Sorg and Cimiano, 2012) extended ESA to other languages and showed that it is useful in cross-lingual and multilingual retrieval task. Their focus was on creating a language independent conceptual space in which documents would be mapped and then retrieved.

Some open ESA topics related to bilingual lexicon creation include: (1) the document representation which is simply done by summing individual contributions of words, (2) the adaptation of the method to specific domains and (3) the coverage of the underlying resource in different language.

3 ESA for Bilingual Lexicon Extraction

The main objective of our approach is to devise lexicon translation methods that are easily applicable to a large number of language pairs, while preserving the overall quality of results. A subordinated objective is to exploit large scale background multilingual knowledge, such as the encyclopedic content available in Wikipedia. As we mentioned, ESA

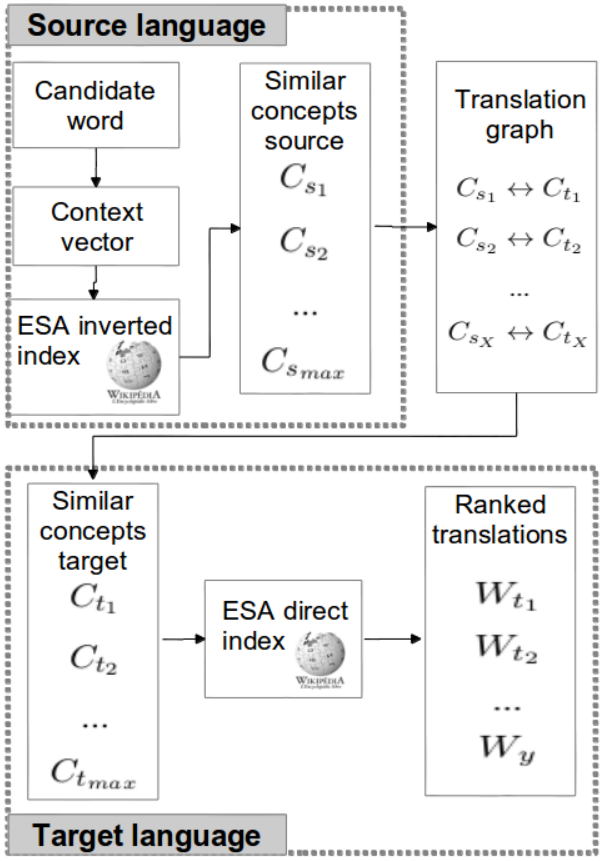


Figure 1: Overview of the Explicit Semantic Analysis enabled bilingual lexicon extraction.

(Gabrilovich and Markovitch, 2007) was exploited in a number of NLP tasks but not in bilingual lexicon extraction.

Figure 1 shows the overall architecture of the lexical extraction process we propose. The process is completed in the following three steps:

1. Given a word to be translated and its context vector in the source language, we derive a ranked list of similar Wikipedia concepts (i.e. articles) using the ESA inverted index.

2. Then, a translation graph is used to retrieve the corresponding concepts in the target language.

3. Candidate translations are found through a statistical processing of concept descriptions from the ESA direct index in the target language.

In this section, we first introduce the elements of the original formulation of ESA necessary in our approach. Then, we detail the three steps that compose the main bilingual lexicon extraction method illustrated in Figure 1. Finally, as a complement to the main method we introduce a measure for domain word specificity and present a method for extracting generic translation lexicons.

3.1 ESA Word and Concept Representation

Given a semantic space structured using a set of M concepts and including a dictionary of N words, a mapping between words and concepts can be expressed as the following matrix:

$$\begin{matrix} w(W_1,C_1) & w(W_2,C_1) & ... & w(W_N,C_1) \\ w(W_1,C_2) & w(W_2,C_2) & ... & w(W_N,C_2) \\ ... & ... & ... & \\ w(W_1,C_M) & w(W_2,C_M) & ... & w(W_N,C_M) \end{matrix}$$

When Wikipedia is exploited concepts are equated to Wikipedia articles and the texts of the articles are processed in order to obtain the weights that link words and concepts. In (Gabrilovich and Markovitch, 2007), the weights w that link words and concepts were obtained through a classical TF-IDF weighting of Wikipedia articles. A series of tweaks destined to improve the method's performance were used and disclosed later[3]. For instance, administration articles, lists, articles that are too short or have too few links are discarded. Higher weight is given to words in the article title and more longer articles are favored over shorter ones. We implemented a part of these tweaks and tested our own version of ESA with the Wikipedia version used in the original implementation. The correlation with human judgments of word relatedness was 0.72 against 0.75 reported by (Gabrilovich and Markovitch, 2007). The ESA matrix is sparse since the N size of the dictionary, is usually in the range of hundreds of thousands and each concept is usually described by hundreds of distinct words. The direct ESA index from Figure 1 is obtained by reading the matrix horizontally while the inverted ESA index is obtained by reading the matrix vertically.

[3] https://github.com/faraday/wikiprep-esa/wiki/roadmap

Terme	Concepts
action	*évaluation d'action, communisme, actionnaire activiste, socialisme, dévelopement durable ...*
déficit	*crise de la dette dans la zone euro, dette publique, règle d'or budgétaire, déficit, trouble du déficit de l'attention ...*
cisaillement	*taux de cisaillement, zone de cisaillement, cisaillement, contrainte de cisaillement, viscoanalyseur ...*
turbine	*ffc turbine potsdam, turbine à gaz, turbine, urbine hydraulique, cogénération ...*
cryptage	*TEMPEST, chiffrement, liaison 16, Windows Vista, transfert de fichiers ...*
protocole	*Ad-hoc On-demand Distance Vector, protocole de Kyoto, optimized link state routing protocol, liaison 16, IPv6 ...*
biopsie	*biopsie, maladie de Horton, cancer du sein, cancer du poumon, imagerie par résonance magnétique ...*
palpation	*cancer du sein, cellulite, examen clinique, appendicite, ténosynovite ...*

Table 1: The five most similar Wikipedia concepts to the French terms *action[share], déficit[deficit], cisaillement[shear], turbine[turbine], cryptage[encryption], biopsie[biopsie]* and *palpation[palpation]* and their context vectors.

3.2 Source Language Processing

The objective of the source language processing is to obtain a ranked list of similar Wikipedia concepts for each candidate word (W_{cand}) in a specialized domain. To do this, a context vector is first built for each W_{cand} from a specialized monolingual corpus. The association measure between W_{cand} and context words is obtained using the Odds-Ratio (defined in equation 5). Wikipedia concepts in the source language C_s that are similar to W_{cand} and to a part of its context words are extracted and ranked using equation 1.

$$Rank(C_s) = (10 * \max(Odds_{W_{s_i}}^{W_{cand}}) * w(W_{cand}, C_s)) + \sum_{i=1}^{n} Odds_{W_{s_i}}^{W_{cand}} * w(W_{s_i}, C_s) \quad (1)$$

where $\max(Odds_{W_{s_i}}^{W_{cand}})$ is the highest Odds-Ratio association between W_{cand} and any of its context words W_{s_i}; the factor 10 was empirically set to give more importance to W_{cand} over context words; $w(W_{cand}, C_s)$ is the weight of the association between W_{cand} and C_s from the ESA matrix; n is the total number of words W_{s_i} in the context vector of W_{cand}; $Odds_{W_{s_i}}^{W_{cand}}$ is the association value between W_{cand} and W_{s_i} and $w(W_{s_i}, C_s)$ are the weights of the associations between each context word W_{s_i} and C_s from the ESA matrix. The use of contextual information in equation 1 serves to characterize the candidate word in the target domain.

In table 1, we present the five most similar Wikipedia concepts to the French terms *action, déficit, cisaillement, turbine, cryptage, biopsie* and *palpation* and their context vectors. These terms are part of the four specialized domains we are studying here. From observing these examples, we note that despite the difference between the specialized domains and word ambiguity (words *action* and *protocole*), our method has the advantage of successfully representing each word to be translated by relevant conceptual spaces.

3.3 Translation Graph Construction

To bridge the gap between the source and target languages, a concept translation graph that enables the multilingual extension of ESA is used. This concept translation graph is extracted from the explicit translation links available in Wikipedia articles and is exploited in order to connect a word's conceptual space in the source language with the corresponding conceptual space in the target language. Only a part of the articles have translations and the size of

the conceptual space in the target language is usually smaller than the space in the source language. For instance, the French-English translation graph contains 940,215 pairs of concepts while the French and English Wikipedias contain approximately 1.4 million articles, respectively 4.25 million articles.

3.4 Target Language Processing

The third step of the approach takes place in the target language. Using the translation graph, we select the 100 most similar concept translations (threshold determined empirically after preliminary experiments) from the target language and use their direct ESA representations in order to retrieve potential translations for the candidate word W_{cand} from source language. These candidate translations W_t are ranked using equation 2.

$$Rank(W_t) = (\sum_{i=1}^{n} \frac{w(W_t, C_{t_i})}{avg(C_{t_i})}) * log(count(W_t, \mathbf{S})) \quad (2)$$

with $w(W_t, C_{t_i})$ is the weight of the translation candidate W_T for concept C_{t_i} from the ESA matrix in the target language; $avg(C_{t_i})$ is the average TF-IDF score of words that appear in C_{t_i}; $\mathbf{S}$ is the set of similar concepts C_{t_i} in the target language and $count(W_t, \mathbf{S})$ accounts for the number of different concepts from $\mathbf{S}$ in which the candidate translation W_T appears.

The accumulation of weights $w(W_t, C_{t_i})$ follows the way original ESA text representations are calculated (Gabrilovich and Markovitch, 2007) and $avg(C_{t_i})$ is used in order to correct the bias of the TF-IDF scheme towards short articles. $log(count(W_t, \mathbf{S}))$ is used to favor words that are associated with a larger number of concepts. log weighting was chosen after preliminary experiments with a wide range of functions.

3.5 Domain Specificity

In previous works, ESA was usually exploited in generic tasks that did not require any domain adaptation. Here we process information from specific domains and we need to measure the specificity of words in those domains. The domain extraction is seeded by using Wikipedia concepts (noted C_{seed}) that best describes the domain in the target language. For instance, in English, the *Corporate Finance* domain is seeded with *https://en.wikipedia.org/wiki/Corporate_finance*. We extract a set of 10 words with the highest TF-IDF score from this article (noted SW) and use them to retrieve a domain ranking of concepts in the target language $Rank_{dom}(C_t)$ with equation 3.

$$Rank_{dom}(C_t) = (\sum_{i=1}^{n} w(W_{t_i}, C_t) * w(C_{seed}, W_{t_i})) * count(SW, C_t) \quad (3)$$

where n is size of the seed list of words (i.e. 10 items), $w(W_{t_i}, C_t)$ is the weight of the domain words in the concept C_t ; $w(C_{seed}, W_{t_i})$ is the weight of W_{t_i} in C_{seed}, the seed concept of the domain, and $count(SW, C_t)$ is the number of distinct seed words from SW that appear in C_t.

The first part of equation 3 sums up the contributions of different words from SW that appear in C_t while the second part is meant to further reinforce articles that contain a larger number of domain keywords from SW.

Domain delimitation is performed by retaining articles whose $Rank_{dom}(C_t)$ is at least 1% or the score of the top $Rank_{dom}(C_t)$ score. This threshold was set up during preliminary experiments. Given the delimitation obtained with equation 3, we calculate a domain specificity score ($specif_{dom}(W_t)$) for each word that occurs in the domain (equation 4). $specif_{dom}(W_t)$ estimates how much of a word's use in an underlying corpus is related to a target domain.

$$specif_{dom}(W_t) = \frac{DF_{dom}(W_t)}{DF_{gen}(W_t)} \quad (4)$$

where DF_{dom} and DF_{gen} stand for the domain and the generic document frequency of the word W_t.

$specif_{dom}(W_t)$ will be used to favor words with greater domain specificity over more general ones when several translations are available in a seed generic translation lexicon. For instance, the French word *action* is ambiguous and has English translations such as *action*, *stock*, *share* etc. In a general case, the most frequent translation is *action* whereas in a *corporate finance* context, *share* or *stock* are more relevant. The specificity of the three translations, from highest to lowest, is: *share*, *stock* and *action* and is used to rank these potential translations.

3.6 Generic Dictionaries

Generic translation dictionaries, already used by existing bilingual lexicon extraction approaches, can also be integrated in the newly proposed approach. The Wikipedia translation graph is transformed into a translation dictionary by removing the disambiguation marks from ambiguous concept titles, as well as lists, categories and other administration pages. Moreover, since the approach does not handle multiword units, we retain only translation pairs that are composed of unigrams in both languages. When existing, unigram redirections are also added in each language.

The obtained dictionaries are incomplete since: (1) Wikipedia focuses on concepts that are most often nouns, (2) specialized domain terms often do not have an associated Wikipedia entry and (3) the translation graph covers only a fraction of the concepts available in a language. For instance, the resulting translation dictionaries have 193,543 entries for French-English and 136,681 entries for Romanian-English. They can be used in addition to or instead of other resources available and are especially useful when there are only few other resources that link the pair of languages processed.

4 Evaluation

The performances of our approach are evaluated against the standard approach and its developments proposed by (Morin and Prochasson, 2011) and (Bouamor et al., 2013). In this section, we first describe the data and resources we used in our experiments. We then present differents parameters needed in the implementation of the different methods tested. Finally, we discuss the obtained results.

4.1 Data and Resources

Comparable corpora

We conducted our experiments on four French-English and Romanian-English specialized comparable corpora: *Corporate Finance, Breast Cancer, Wind Energy* and *Mobile Technology*. For the Romanian-English language pair, we used Wikipedia to collect comparable corpora for all domains since they were not already available. The Wikipedia corpora are harvested using a category-based selection. We consider the topic in the source language (for instance *Cancer Mamar* [*Breast Cancer*]) as a query to Wikipedia and extract all its subtopics (i.e., sub-categories) to construct a domain-specific *category tree*. Then, based on the constructed tree, we collect all Wikipedia articles belonging to at least one of these categories and use *inter-language links* to build the comparable corpora.

Domain	FR	EN
Corporate Finance	402,486	756,840
Breast Cancer	396,524	524,805
Wind Energy	145,019	345,607
Mobile Technology	197,689	144,168
Domain	RO	EN
Corporate Finance	206,169	524,805
Breast Cancer	22,539	322,507
Wind Energy	121,118	298,165
Mobile Technology	200,670	124,149

Table 2: Number of *content words* in the comparable corpora.

Concerning the French-English pair, we followed the strategy described above to extract the comparable corpora related to the *Corporate Finance* and *Breast Cancer* domains since they were otherwise unavailable. For the two other domains, we used the corpora released in the TTC project[4]. All corpora were normalized through the following linguistic preprocessing steps: tokenization, part-of-speech tagging, lemmatization, and function word removal. The resulting corpora[5] sizes are presented in Table 2. The size of the domain corpora vary within and across languages, with the corporate finance domain being the richest in both languages. In Romanian, *Breast Cancer* is particularly small, with approximately 22,000 tokens included. This variability will allow us to test if there is a correlation between corpus size and quality of results.

Bilingual dictionary

The seed generic French-English dictionary used to translate French context vectors consists of an in-house manually built resource which contains approximately 120,000 entries. For Romanian-

[4] http://www.ttc-project.eu/index.php/releases-publications

[5] Comparable corpora will be shared publicly

Domain	FR-EN	RO-EN
Corporate Finance	125	69
Breast Cancer	96	38
Wind Energy	89	38
Mobile Technology	142	94

Table 3: Sizes of the evaluation lists.

English, we used the generic dictionary extracted following the procedure described in Subsection 3.6.

Gold standard

In bilingual terminology extraction from comparable corpora, a reference list is required to evaluate the performance of the alignment. Such lists are usually composed of around 100 single terms (Hazem and Morin, 2012; Chiao and Zweigenbaum, 2002). Reference lists[6] were created for the four specialized domains and the two pairs of languages. For the French-English, reference words from the *Corporate Finance* domain were extracted from the glossary of bilingual micro-finance terms[7]. For *Breast Cancer*, the list is derived from the MESH and the UMLS thesauri[8]. Concerning *Wind Energy* and *Mobile Technology*, lists were extracted from specialized glossaries found on the Web. The Romanian-English gold standard was manually created by a native speaker starting from the French-English lists. Table 3 displays the sizes of the obtained lists. Reference terms pairs were retained if each word composing them appeared at least five times in the comparable domain corpora.

4.2 Experimental setup

Aside from those already mentioned, three parameters need to be set up: (1) the window size that defines contexts, (2) the association measure that measures the strength of the association between words and the (3) similarity measure that ranks candidate translations for state of the art methods. Context vectors are defined using a seven-word window which approximates syntactic dependencies. The association and the similarity measures (Discounted Log-Odds ratio (equation 5) and the cosine similarity) were set following Laroche and Langlais (2010), a comprehensive study of the influence of these parameters on the bilingual alignment.

$$\textit{Odds-Ratio}_{disc} = \log \frac{(O_{11} + \frac{1}{2})(O_{22} + \frac{1}{2})}{(O_{12} + \frac{1}{2})(O_{21} + \frac{1}{2})} \quad (5)$$

where O_{ij} are the cells of the 2×2 contingency matrix of a token s co-occurring with the term S within a given window size.

The F-measure of the Top 20 results (F-Measure@20), which measures the harmonic mean of precision and recall, is used as evaluation metric. Precision is the total number of correct translations divided by the number of terms for which the system returned at least one answer. Recall is equal to the ratio between the number of correct translation and the total number of words to translate (W_{cand}).

4.3 Results and discussion

In addition to the basic approach based on ESA (denoted ESA), we evaluate the performances of a method so-called Dico_{Spec} in which the translations are extracted from a generic dictionary and a method we called ESA_{Spec} which combine ESA and Dico_{Spec}. DICO_{Spec} is based on the generic dictionary we presented in subsection 3.6 and proceeds as follows: we extract a list of translations for each word to be translated from the generic dictionary. The domain specificity introduced in subsection 3.5 is then used to rank these translations. For instance, the french term *port* referring in the *Mobile Technology* domain, to the system that allows computers to receive and transmit information is translated into *port* and *seaport*. According to domain specificity values, the following ranking is obtained: the English term *port* obtain the highest specificity value (0.48). *seaport* comes next with a specificity value of 0.01. In ESA_{Spec}, the translations set out in the translations lists proposed by both ESA and the generic dictionary are weighted according to their domain specificity values. The main intuition behind this method is that by adding the information about the domain specificity, we obtain a new ranking of the bilingual extraction results.

The obtained results are displayed in table 4. The comparison of state of the art method shows that BA13 performs better than STAPP and MP11 for French-English and has comparable performances

[6]Reference lists will be shared publicly

[7]http://www.microfinance.lu/en/

[8]http://www.nlm.nih.gov/

a) FR-EN

Method	F-Measure@20			
	Breast Cancer	Corporate Finance	Wind Eenrgy	Mobile Technology
STAPP	0.49	0.17	0.08	0.06
MP11	0.55	0.33	0.24	0.05
BA13	0.61	0.37	0.30	0.24
Dico$_{spec}$	0.50	0.20	0.36	0.25
ESA	0.74	0.50	0.83	0.72
ESA$_{spec}$	**0.81**	**0.56**	**0.86**	**0.75**

b) RO-EN

Method	F-Measure@20			
	Breast Cancer	Corporate Finance	Wind Eenrgy	Mobile Technology
STAPP	0.21	0.13	0.08	0.16
MP11	0.21	0.13	0.08	0.16
BA13	0.21	0.14	0.08	0.17
Dico$_{spec}$	0.44	0.11	0.21	0.16
ESA	0.76	0.17	**0.58**	0.53
ESA$_{spec}$	**0.78**	**0.24**	**0.58**	**0.55**

Table 4: Results of the specialized dictionary creation on four specific domains, two pairs of languages.Three state of the art methods were used for comparison: STAPP is the standard approach, MP11 is the improvement of the standard approach introduced in (Morin and Prochasson, 2011), BA13 is a recent method that we developed (Bouamor et al., 2013). Dico$_{spec}$ exploits a generic dictionary, combined with the use of domain specificity (see Subsection 3.5). ESA stands for the ESA based approach introduced in this paper (see Figure 1). ESA$_{spec}$ combines the results of Dico$_{spec}$ and ESA.

for RO-EN. Consequently, we will use BA13 as the main baseline for discussing the newly introduced approach. The results presented in Table 4 show that ESA$_{spec}$ clearly outperforms the three baselines for the four domains and the two pairs of languages tested. When comparing ESA$_{spec}$ to BA13 for French-English, improvements range between 0.19 for *Corporate Finance* and 0.56 for *Wind Energy*. For RO-EN, the improvements vary from 0.1 for *Corporate Finance* to 0.5 for *Wind Energy*. Also, except for the *Corporate Finance* domain in Romanian, the performance variation across domains is much smaller for ESA$_{spec}$ than for the three state of the art methods. This shows that ESA$_{spec}$ is more robust to domain change and thus more generic.

The results obtained with ESA are significantly better than those obtained with Dico$_{spec}$ and ESA$_{spec}$, their combination, further improves the results. The main contribution to ESA$_{spec}$ performances comes from ESA, a finding that validates our assumption that the adequate use of a rich multilingual resource such as Wikipedia is appropriate for specialized lexicon translation. Dico$_{spec}$ is a simple method that ranks the different meanings of a candidate word available in a generic dictionary but its average performances are comparable to those of BA13 for FR-EN and higher for RO-EN. This finding advocates for the importance of good quality generic dictionaries in specialized lexicon translation approaches. However, it is clear that such dictionaries are far from being sufficient in order to cover all possible domains. There is no clear correlation between domain size and quality of results. Although richer than the other three domains, *Corporate Finance* has the lowest associated performances. This finding is probably explained by the intrinsic difficulty of each domain. When passing from FR-EN to RO-EN the average performance drop is more significant for BA13 than for the ESA based methods. The result indicates that our approach is more robust to language change.

5 Conclusion

We have presented a new approach to the creation of specialized bilingual lexicons, one of the central

building blocks of machine translation systems. The proposed approach directly tackles two of the major challenges identified in the Introduction. The scarcity of resources is addressed by an adequate exploitation of Wikipedia, a resource that is available in hundreds of languages. The quality of automatic translations was improved by appropriate domain delimitation and linking across languages, as well as by an adequate statistical processing of concepts similar to a word in a given context.

The main advantages of our approach compared to state of the art methods come from: the increased number of languages that can be processed, from the smaller sensitivity to structured resources and the appropriate domain delimitation. Experimental validation is obtained through evaluation with four different domains and two pairs of languages which shows consistent performance improvement. For French-English, two languages that have rich associated Wikipedia representations, performances are very interesting and are starting to approach those of manual translations for three domains out of four (F-Measure@20 around 0.8). For Romanian-English, a pair involving a language with a sparser Wikipedia representation, the performances of our method drop compared to French-English . However, they do not decrease to the same extent as those of the best state of the art method tested. This finding indicates that our approach is more general and, given its low language dependence, it can be easily extended to a large number of language pairs.

The results presented here are very encouraging and we will to pursue work in several directions. First, we will pursue the integration of our method, notably through comparable corpora creation using the data driven domain delimitation technique described in Subsection 3.5. Equally important, the size of the domain can be adapted so as to find enough context for all the words in domain reference lists. Second, given a word in a context, we currently exploit all similar concepts from the target language. Given that comparability of article versions in the source and the target language varies, we will evaluate algorithms for filtering out concepts from the target language that have low alignment with their source language versions. A final line of work is constituted by the use of distributional properties of texts in order to automatically rank parts of concept descriptions (i.e. articles) by their relatedness to the candidate word. Similar to the second direction, this process involves finding comparable text blocks but rather at a paragraph or sentence level than at the article level.

References

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2013. Context vector disambiguation for bilingual lexicon extraction. In *Proceedings of the 51st Association for Computational Linguistics (ACL-HLT)*, Sofia, Bulgaria, August.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th international conference on Computational linguistics - Volume 2*, COLING '02, pages 1–5. Association for Computational Linguistics.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2003. The effect of a general lexicon in corpus-based identification of french-english medical word translations. In *Proceedings Medical Informatics Europe, volume 95 of Studies in Health Technology and Informatics*, pages 397–402, Amsterdam.

Pascale Fung. 1998. A statistical view on bilingual lexicon extraction: From parallel corpora to non-parallel corpora. In *Parallel Text Processing*, pages 1–17. Springer.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Alexander Halavais and Derek Lackaff. 2008. An Analysis of Topical Coverage of Wikipedia. *Journal of Computer-Mediated Communication*, 13(2):429–440.

Z.S. Harris. 1954. Distributional structure. *Word*.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *AAAI*.

Amir Hazem and Emmanuel Morin. 2012. Adaptive dictionary for bilingual lexicon extraction from comparable corpora. In *Proceedings, 8th international conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey, May.

Azniah Ismail and Suresh Manandhar. 2010. Bilingual lexicon extraction from comparable corpora using in-domain terms. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 481–489. Association for Computational Linguistics.

Audrey Laroche and Philippe Langlais. 2010. Revisiting context-based projection methods for term-translation spotting in comparable corpora. In *23rd International Conference on Computational Linguistics (Coling 2010)*, pages 617–625, Beijing, China, Aug.

Emmanuel Morin and Béatrice Daille. 2006. Comparabilité de corpus et fouille terminologique multilingue. In *Traitement Automatique des Langues (TAL)*.

Emmanuel Morin and Emmanuel Prochasson. 2011. Bilingual lexicon extraction from comparable corpora enhanced with parallel corpora. In *Proceedings, 4th Workshop on Building and Using Comparable Corpora (BUCC)*, page 27–34, Portland, Oregon, USA.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March.

Emmanuel Prochasson, Emmanuel Morin, and Kyo Kageura. 2009. Anchor points for bilingual lexicon extraction from small comparable corpora. In *Proceedings, 12th Conference on Machine Translation Summit (MT Summit XII)*, page 284–291, Ottawa, Ontario, Canada.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, pages 320–322. Association for Computational Linguistics.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 519–526. Association for Computational Linguistics.

P. Sorg and P. Cimiano. 2012. Exploiting wikipedia for cross-lingual and multilingual information retrieval. *Data Knowl. Eng.*, 74:26–45, April.

Document Summarization via Guided Sentence Compression

Chen Li[1], **Fei Liu**[2], **Fuliang Weng**[2], **Yang Liu**[1]
[1] Computer Science Department, The University of Texas at Dallas
Richardson, Texas 75080, USA
[2] Research and Technology Center, Robert Bosch LLC
Palo Alto, California 94304, USA
{chenli,yangl@hlt.utdallas.edu}
{fei.liu, fuliang.weng@us.bosch.com}

Abstract

Joint compression and summarization has been used recently to generate high quality summaries. However, such word-based joint optimization is computationally expensive. In this paper we adopt the 'sentence compression + sentence selection' pipeline approach for compressive summarization, but propose to perform *summary guided compression*, rather than generic sentence-based compression. To create an annotated corpus, the human annotators were asked to compress sentences while explicitly given the important summary words in the sentences. Using this corpus, we train a supervised sentence compression model using a set of word-, syntax-, and document-level features. During summarization, we use multiple compressed sentences in the integer linear programming framework to select salient summary sentences. Our results on the TAC 2008 and 2011 summarization data sets show that by incorporating the guided sentence compression model, our summarization system can yield significant performance gain as compared to the state-of-the-art.

1 Introduction

Automatic summarization can be broadly divided into two categories: extractive and abstractive summarization. Extractive summarization focuses on selecting the salient sentences from the document collection and concatenating them to form a summary; while abstractive summarization is generally considered more difficult, involving sophisticated techniques for meaning representation, content planning, surface realization, etc., and the "true abstractive summarization remains a researcher's dream" (Radev et al., 2002).

There has been a surge of interest in recent years on generating compressed document summaries as a viable step towards abstractive summarization. These compressive summaries often contain more information than sentence-based extractive summaries since they can remove insignificant sentence constituents and make space for more salient information that is otherwise dropped due to the summary length constraint. Two general strategies have been used for compressive summarization. One is a pipeline approach, where sentence-based extractive summarization is followed or proceeded by sentence compression (Knight and Marcu, 2000; Lin, 2003; Zajic et al., 2007; Wang et al., 2013). Another line of work uses joint compression and summarization. They have been shown to achieve promising performance (Daumé, 2006; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Chali and Hasan, 2012; Almeida and Martins, 2013; Qian and Liu, 2013). One popular approach for such joint compression and summarization is via integer linear programming (ILP). However, since words are the units in the optimization framework, solving this ILP problem can be expensive.

In this study, we use the pipeline compression and summarization method because of its computational efficiency. Prior work using such pipeline methods simply uses generic sentence-based compression for each sentence in the documents, no matter whether compression is done before or after summary sentence extraction. We propose to use sum-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 490–500,
Seattle, Washington, USA, 18-21 October 2013.

mary guided compression combined with ILP-based sentence selection for summarization in this paper. We create a compression corpus for this purpose. Using human summaries for a set of documents, we identify salient words in the sentences. During annotation, the human annotators are given these salient words and asked to generate compressed sentences. We expect such "guided" sentence compression is beneficial for the pipeline compression and summarization task. In addition, previous research on joint modeling for compression and summarization suggested that the labeled extraction and compression data sets would be helpful for learning a better joint model (Daumé, 2006; Martins and Smith, 2009). We hope that our work on this guided compression will also be of benefit to the future joint modeling studies.

Using our created compression data, we train a supervised compression model using a variety of word-, sentence-, and document-level features. During summarization, we generate multiple compression candidates for each sentence, and use the ILP framework to select compressed summary sentences. In addition, we also propose to apply a preselection step to select some important sentences, which can both speed up the summarization system and improve performance. We evaluate our proposed summarization approach on the TAC 2008 and 2011 data sets using the standard ROUGE metric (Lin, 2004). Our results show that by incorporating a guided sentence compression model, our summarization system can yield significant performance gain as compared to the state-of-the-art reported results.

2 Related Work

Summarization research has seen great development over the last fifty years (Nenkova and McKeown, 2011). Compared to the abstractive counterpart, extractive summarization has received considerable attention due to its clear problem formulation – to extract a set of salient and non-redundant sentences from the given document set. Both unsupervised and supervised approaches have been explored for sentence selection. The supervised approaches include the Bayesian classifier (Kupiec et al., 1995), maximum entropy (Osborne, 2002), skip-chain conditional random fields (CRF) (Galley, 2006), discriminative reranking (Aker et al., 2010), among others.

The extractive summary sentence selection problem can also be formulated in an optimization framework. Previous approaches include the integer linear programming (ILP) and submodular functions, which are used to solve the optimization problem. In particular, Gillick et al. (2009) proposed a concept-based ILP approach for summarization. Li et al. (2013) improved it by using supervised stragety to estimate concept weight in ILP framework. In (Lin and Bilmes, 2010), the authors model the sentence selection problem as maximizing a submodular function under a budget constraint. A greedy algorithm is proposed to efficiently approximate the solution to this NP-hard problem.

Compressive summarization receives increasing attention in recent years, since it offers a viable step towards abstractive summarization. The compressed summaries can be generated through a joint model of the sentence selection and compression processes, or through a pipeline approach that integrates a generic sentence compression model with a summary sentence pre-selection or post-selection step.

Many studies explore the joint sentence compression and selection setting. Martins and Smith (2009) jointly perform sentence extraction and compression by solving an ILP problem; Berg-Kirkpatrick et al. (2011) propose an approach to score the candidate summaries according to a combined linear model of extractive sentence selection and compression. They train the model using a margin-based objective whose loss captures the final summary quality. Woodsend and Lapata (2012) present a method where the summary's informativeness, succinctness, and grammaticality are learned separately from data but optimized jointly using an ILP setup; Yoshikawa et al. (2012) incorporate semantic role information in the ILP model; Chali and Hasan (2012) investigate three strategies in compressive summarization: compression before extraction, after extraction, or joint compression and extraction in one global optimization framework. These joint models offer a promise for high quality summaries, but they often have high computational cost. Qian and Liu (2013) propose a graph-cut based method that improves the speed of joint compression and summarization.

The pipeline approach, where sentence-based extractive summarization is followed or proceeded by sentence compression, is also popular. Knight and Marcu (2000) utilize the noisy channel and decision tree method to perform sentence compression; Lin (2003) shows that pure syntactic-based compression may not improve the system performance; Zajic et al. (2007) compare two sentence compression approaches for multi-document summarization, including a 'parse-and-trim' and a noisy-channel approach; Galanis and Androutsopoulos (2010) use the maximum entropy model to generate the candidate compressions by removing the branches from the source sentences; Liu and Liu (2013) couple the sentence compression and extraction approaches for summarizing the spoken documents; Wang et al. (2013) design a series of learning-based compression models built on parse trees, and integrate them in query-focused multi-document summarization. Prior studies often rely heavily on the generic sentence compression approaches (McDonald, 2006; Nomoto, 2007; Clarke and Lapata, 2008; Thadani and McKeown, 2013) for compressing the sentences in the documents, yet a generic compression system may not be the best fit for the summarization purpose.

In this paper, we adopt the pipeline-based compressive summarization framework, but propose a novel *guided compression* method that is catered to the summarization task. We expect this approach to take advantage of the efficient pipeline processing while producing satisfying results as the joint models. We train a supervised guided compression model to produce n-best compressions for each sentence, and use an ILP formulation to select the best set of summary sentences. In addition, we propose to apply a sentence pre-selection step to further accelerate the processing and enhance the performance.

3 Guided Compression Corpus

The goal of guided sentence compression is to create compressed sentences that are grammatically correct and contain the important information that we would like to preserve in the final summary. Following the compression literature (Clarke and Lapata, 2008), the compression task is defined as a word deletion problem, that is, the human annotators (and also automatic compression systems) are allowed to only remove words from the original sentence to form a compression. The key difference between our proposed guided compression with generic sentence compression is that, we provide guidance to the human compression process by specifying a set of "important words" that we wish to keep for each sentence. We expect this kind of summary oriented compression would benefit the ultimate summarization task. Take the sentence shown in Table 1 as an example. For generic sentence compression, there may be multiple 'good' human compressions for this sentence, such as those listed in the table. Without guidance, a human annotator (or automatic system) is likely to use option A or B; however, if "18 hours" appears in the summary, then we want to provide this guidance in the compression process, hence option C may be the best compression choice. This guided compression therefore avoids removing the salient words that are important to the final summary.

Original Sentence:
The gas leak was contained Monday afternoon , nearly 18 hours after it was reported , Statoil spokesman Oeivind Reinertsen said .
Compression A:
The gas leak was contained
Compression B:
The gas leak was contained ***Monday afternoon***
Compression C:
The gas leak was contained ***nearly 18 hours after it was reported***

Table 1: Example sentence and three compressions.

To generate the guided compression corpus, we use the TAC 2010 data set[1] that was used for the multi-document summarization task. There are 46 topics. Each has 10 news documents, and also four human-created abstractive reference summaries. Since annotating all the sentences in this data set is time consuming and some sentences are not very important for the summarization task, we choose a set of sentences that are highly related to the human abstracts for annotation. We compare each sentence with the four human abstracts using the ROUGE-2 metric (Lin, 2004), and the sentences

[1] http://www.nist.gov/tac/2010/

Original Sentence: He said ***Vietnam veterans*** are ***presumed to have been exposed to Agent Orange*** and veterans with any of the ***10 diseases*** is presumed to have contracted it from the exposure , without individual proof .
Guided Compression: Vietnam veterans are presumed to have been exposed to Agent Orange.
Original Sentence: The ***province has*** limited the number of trees to be chopped down in the forest area in northwest Yunnan and has ***stopped building sugar factories in the Xishuangbanna region to preserve*** the only ***tropical rain forest*** in the country located there .
Guided Compression: province has stopped building sugar factories in the Xishuangbanna region to preserve tropical rain forest.

Table 2: Example original sentences and their guided compressions. The "guiding words" are italicized and marked in red.

with the highest scores are selected.

In annotation, human annotators are provided with important 'guiding words' (highlighted in the annotation interface) that we want to preserve in the sentences. We calculate the word overlap between a sentence and each of those sentences in the human abstracts, and use a set of heuristic rules to determine the "guiding words" in a sentence: the longest consecutive word overlaps (greater than 2 words) in each sentence pair are first selected; the rest overlaps that contain 2 or more words (excluding the stopwords) are also selected. We suggest the human annotators to use their best judgment to keep the guiding words as many as possible while compressing the sentence.

We use the Amazon Mechanical Turk (AMT) for data annotation[2]. In total, we select 1,150 sentences from the TAC news documents. They are grouped into about 230 human intelligence tasks (HITs) with 5 sentences in each HIT. A sentence was compressed by 3 human annotatorsand we select the shortest candidate as the goldstandard compression for each sentence. In Table 2, we show two example sentences, their guiding words (bold), and the human compressions. The first example shows that giving up some guiding words is acceptable, since more unnecessary words will be included in order to accommodate all the guiding words; the second example shows that the guided compression can lead to more aggressive word deletions since the constituents that are not important to the summary will be deleted even though they contain salient information by themselves.

For our compression corpus, which contains 1,150 sentences and their guided compressions, the average compression rate, as measured by the percentage of dropped words, is about 50%. This compression ratio is higher compared to other generic sentence compression corpora, in which the word deletion rate ranges from 24% to 34% depending on different text genres and annotation guidelines (Clarke and Lapata, 2008; Liu and Liu, 2009). This suggests that the annotators can remove words more aggressively when they are provided with a limited set of guiding words.

4 Summarization System

Our summarization system consists of three key components: we train a supervised guided compression model using our created compression data, with a variety of features.then we use this model to generate n-best compressions for each sentence; we feed the multiple compressed sentences to the ILP framework to select the best summary sentences. In addition, we propose a sentence pre-selection step that can both speed up the summarization system and improve the performance.

4.1 Guided Sentence Compression

Sentence compression has been explored in previous studies using both supervised and unsupervised approaches, including the noisy-channel and decision tree model (Knight and Marcu, 2000; Turner and Charniak, 2005), discriminative learning (McDonald, 2006), integer linear programming (Clarke and Lapata, 2008; Thadani and McKeown, 2013), conditional random fields (CRF) (Nomoto, 2007; Liu and Liu, 2013), etc. In this paper, we employ the CRF-based compression approach due to its proved performance and its flexibility to integrate different levels of discriminative features. Under this framework, sentence compression is formulated as a sequence labeling problem, where each word is

[2]http://www.mturk.com

labeled as either "0" (retained) or "1" (removed). We develop different levels of features to capture word-specific characteristics, sentence related information, and document level importance. Most of the features are extracted based only on the sentence to be compressed. However, we introduce a few document level features. These are designed to capture the word and sentence significance within the given document collection and are thus expected to be more summary related.

Word and sentence features:

- **Word n-grams**: identity of the current word and two words before and after, as well as all the bigrams and trigrams that can be formed by the adjacent words and the current word.
- **POS n-grams**: same as the word n-grams, but use the part-of-speech tags instead.
- **Named entity tags**: binary features representing whether the current word is a person, location, or temporal expression. We use the Stanford CoreNLP tools[3] for named entity tagging.
- **Stopwords**: whether the current word is a stopword or not.
- **Conjunction features**: (1) conjunction of the current word with its relative position in the sentence; (2) conjunction of the NER tag with its relative position.
- **Syntactic features**: We obtain the syntactic parsing tree using the Berkeley Parser (Petrov and Klein, 2007), then obtain the following features: (1) the last sentence constituent tag in the path from the root to the word; (2) depth: length of the path starting from the root node to the word; (3) normalized depth: depth divided by the longest path in the parsing tree; (4) whether the word is under an SBAR node; (5) depth and normalized depth of the SBAR node if the word is under an SBAR node;
- **Dependency features:** We employ the Penn2Malt toolkit [4] to convert the parse result from the Berkeley parser to the dependency parsing tree, and use these dependency features: (1) dependency relations such as 'AMOD' (adjective modifier), 'NMOD' (noun modifier), etc. (2) whether the word has a child, left child, or right child in the dependency tree.

Document-level features:

- **Sentence salience score:** We use a simple regression model to estimate a salience score for each sentence (more details in Section 4.3), which represents the importance of the sentence in the document. This score is discretized into four binary features according to the average sentence salience.
- **Unigram document frequency:** this is the current word's document frequency based on the 10 documents associated with each topic.
- **Bigram document frequency:** document frequency for the two bigrams, the current word and its previous or next word.

Some of the above features were employed in related sentence compression studies (Nomoto, 2007; Liu and Liu, 2013). In addition to these features, we explored other related features, including the absolute position of the current word, whether the word appears in the corresponding topic title and descriptions, conjunction of the syntactic tag with the tree depth, etc.; however, these features did not lead to improved performance. We train the CRF model with the Pocket CRF toolkit[5] using the guided compression corpus collected in Section 3. During summarization, we apply the model to a given sentence to generate its n-best guided compressions and use them in the following summarization step.

4.2 Summary Sentence Selection

The sentence selection process is similar to the standard sentence-based extractive summarization, except that the input to the selection module is a list of compressed sentences in our work. Many extractive summarization approaches can be applied for this purpose. In this work, we choose the integer linear programming (ILP) method, specifically, the concept-based ILP framework introduced in (Gillick

[3] http://nlp.stanford.edu/software/corenlp.shtml

[4] http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html

[5] http://sourceforge.net/projects/pocket-crf-1/

et al., 2009), mainly because it yields best performance in the TAC evaluation tasks. This ILP approach aims to extract sentences that can cover as many important concepts as possible, while ensuring the summary length is within a given constraint. We follow the study in (Gillick et al., 2009) to use word bi-grams as concepts, and assign a weight to each bi-gram using its document frequency in the given document collection for a test topic. Two differences are between our ILP setup and that in (Gillick et al., 2009). First, since we use multiple compressions for one sentence, we need to introduce an additional constraint: for each sentence, only one of the n-best compressions may be included in the summary. Second, we optimize a joint score of the concept coverage and the sentence salience. The formal ILP formulation is shown below:

$$\max \quad \sum_i w_i c_i + \sum_j v_j \sum_k s_{jk} \tag{1}$$

$$s.t. \quad \sum_k s_{jk} \leq 1 \forall j \tag{2}$$

$$s_{jk} Occ_{i\_jk} \leq c_i \tag{3}$$

$$\sum_{jk} s_{jk} Occ_{i\_jk} \geq c_i \tag{4}$$

$$\sum_{jk} l_{jk} s_{jk} \leq L \tag{5}$$

$$c_i \in \{0,1\} \ \forall i \tag{6}$$

$$s_{jk} \in \{0,1\} \ \forall j,k \tag{7}$$

where c_i and s_{jk} are binary variables indicating the presence of a concept and a sentence respectively; s_{jk} denotes the k^{th} candidate compression of the j^{th} sentence; w_i represents the weight of the concept; v_j is the sentence salience score of the j^{th} sentence, predicted using a regression model (Section 4.3), and all of its compressed candidates share this value. (1) is the new objective function we use that combines the coverage of the concepts and the sentence salience scores. (2) represents our additional constraint, which requires that for each sentence j, only one candidate compression will be chosen. Occ_{i_jk} represents the occurrence of concept i in the sentence s_{jk}. Inequalities (3) and (4) associate the sentences and the concepts. Constraint (5) controls the summary length, as measured by the total number of words in the summary. We use an open source ILP solver[6].

4.3 Sentence Pre-selection

The above ILP method can offer an exact solution to the defined objective function. However, ILP is computationally expensive when the formulation involves large quantities of variables, i.e, when we have many sentences and a large number of candidate compressions for each sentence. We therefore propose to apply a sentence pre-selection step before the compression. This kind of selection step has been used in previous ILP-based summarization systems (Berg-Kirkpatrick et al., 2011; Gillick et al., 2009). In this work, we propose to use a simple supervised support vector regression (SVR) model (Ng et al., 2012) to predict a salience score for each sentence and select the top ranked sentences for further processing (compression and summarization).

To train the SVR model, the target value for each sentence is the ROUGE-2 score between the sentence and the four human abstracts (this same value is used for sentence selection in corpus annotation (Section 3)). We employ three commonly used features: (1) sentence position in the document; (2) sentence length as indicated by a binary feature: it takes the value of 0 if the number of words in the sentence is greater than 50 or less than 10, otherwise the feature value is 1; (3) interpolated n-gram document frequency as introduced in (Ng et al., 2012), which is a weighted linear combination of the document frequency of the unigrams and bigrams contained in the sentence:

$$f(s) = \frac{\alpha \sum_{w_u \in S} DF(w_u) + (1-\alpha) \sum_{w_b \in S} DF(w_b)}{|S|}$$

where w_u and w_b represent the unigrams and bigrams contained in the sentence S; α is a balancing factor; $|S|$ denotes the number of words in the sentence.

The SVR model was trained using the SVMlight toolkit[7]. Using this model, we can predict a salience score (V_j in Eq 1) for each sentence and only select the top n sentences and supply them to the compression and summarization steps. In practice, using a fixed n may not be a good choice since the number

[6] http://www.gnu.org/software/glpk/

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

of sentences varies greatly for different topics. We therefore set n heuristically based on the total number of sentences m for each topic: n=15 if $m > 150$; n=10 if $m < 100$; n=$0.1 * m$ otherwise.

5 Experimental Results

5.1 Experimental Setup

For our experiments, we use the standard TAC data sets[8], which have been used in the NIST competitions and in other summarization studies. In particular, we used the TAC 2010 data set for creating the guided compression corpus and training the SVR pre-selection model, the TAC 2009 data set as development set for parameter tuning, and the TAC 2008 and 2011 data sets as the test set for reporting the final summarization results.

We compare our pipeline summarization system against three recent studies, which have reported some of the highest published results on this task. Berg-Kirkpatrick et al. (2011) introduce a joint model for sentence extraction and compression. The model is trained using a margin-based objective whose loss captures the end summary quality; Woodsend and Lapata (2012) learn individual summary aspects from data, e.g., informativeness, succinctness, grammaticality, stylistic writing conventions, and jointly optimize the outcome in an integer linear programming framework. Ng et al. (2012) exploit category-specific information for multi-document summarization. In addition to the three previous studies, we also report the best achieved results in the TAC competitions.

5.2 Summarization Results

In Table 3 and Table 4, we present the results of our system and the aforementioned summarization studies. We use the ROUGE evaluation metrics (Lin, 2004), with R-2 measuring the bigram overlap between the system and reference summaries and R-SU4 measuring the skip-bigram with the maximum gap length of 4. "Our System" uses the pipeline setting including the three components described in Section 4. We use the SVR-based approach to preselect a set of sentences from the document set; these sentences are further fed to the guided compression module that produces n-best compressions for each sentence; the ILP-based framework is then used to select the summary sentences from these compressions.

[8] http://www.nist.gov/tac/data/index.html

System	R-2	R-SU4	CompR
TAC'08 Best System	11.03	13.96	n/a
(Berg-Kirkpatrick et al., 2011)	11.70	14.38	n/a
(Woodsend et al., 2012)	11.37	14.47	n/a
Our System	**12.35**†	**15.27**†	43.06%
Our System w/o Pre-selection	12.02	14.98	55.69%
Our System w/ Generic Comp	10.88	13.79	30.90%

Table 3: Results on the TAC 2008 data set. "Our System" uses the SVR-based sentence pre-selection + guided compression + ILP-based summary sentence selection. "Our System w/ Generic Comp" uses the pre-selection + generic compression + ILP summary sentence selection setting. "CompR" represents the compression ratio, i.e., percentage of dropped words. † represents our system outperforms the best previous result at the 95% significance level.

System	R-2	R-SU4	CompR
TAC'11 Best System	13.44	16.51	n/a
(Ng et al., 2012)	13.93	16.83	n/a
Our System	**14.40**	**16.89**	39.90%
Our System w/o Pre-selection	13.74	16.5	53.81%
Our System w/ Generic Comp	13.08	16.23	30.10%

Table 4: Results on the TAC 2011 data set. The systems use the same settings as for the TAC 2008 data set.

We can see from the table that in general, our system achieves considerably better results compared to the state-of-the-art on both the TAC 2008 and 2011 data sets. On the TAC 2008 data set, our system outperforms the best reported result at the 95% significance level; on the TAC 2011 data set, our system also yields considerable performance gain though not exceed the 95% significance level. In the following, we show more detailed analysis to study the effect of different system parameters.

With or without sentence pre-selection. First we evaluate the impact of sentence pre-selection step. In Table 3 and Table 4, we include the results when this step is not used ("Our System w/o Pre-selection"). That is, all of the sentences in the documents (excluding those containing less than 5 words) are compressed and used in the ILP-

based summary sentence selection module. We can see that although sentence pre-selection removes some sentences from consideration in the later summarization step, it actually significantly improves system performance. In the TAC 2008 data set, each topic contains averagely 210 sentences; while the pre-selection step chooses 13 sentences among them. These numbers are 185 and 12 for the TAC 2011 data set. Table 5 shows the average running time of each topic in TAC 2011 data for the two systems, with or without the pre-selection step. Here we fix the number of compressions to 100 in both cases for fair comparison. We can see the selection step greatly accelerates the system processing. When applying the pre-selection step, fewer sentences are used in the compression and summarization, this means we are able to use more compression candidates for each sentence (considering the complexity of ILP module). Using the TAC 2009 as development set, we tuned the number of candidate compressions generated for each sentence. Without pre-selection, we used the 100-best candidates generated from the compression model; with pre-selection, we are able to increase the number to 200-best candidate compressions and still maintain reasonable computational cost. These are the numbers used in the results in Table 3 and 4. Using more compressions helps improve summarization performance. We also notice that the compression ratios are quite different when using sentence pre-selection vs. not. This suggests that in the important sentences (those are kept after pre-selection), there is more summary related information and thus the compression model keeps more words in them (lower compression ratio).

System	Compressed Sentences	Number of Compressions	Running Time (sec)
w/o Pre-selection	185	100	3.9
w/ Pre-selection	12	100	0.85

Table 5: Average running time of our system, w/ or w/o the sentence pre-selection step. Experiments conducted on the TAC 2011 data set. Running time refers only to the execution time of the ILP module for each topic.

Number of compression candidates. This parameter (denoted as n) also impacts system performance. Figure 1 shows the R-2 scores of the two systems (with and without the sentence pre-selection step) when using different number of compressions for each sentence. In general, we find that the R-2 scores do not change much when n is large enough. For example, the 'with pre-selection' system can achieve relatively stable R-2 scores on the TAC 2008 data set (ranging from 12.2 to 12.4) when m is greater than 140; similarly, the R-2 scores on the TAC 2011 data is over 14.2 when m is greater than 100. Without the pre-selection step, the scores are less stable in regard to the changing of the m value, since the large amount of sentences plus a high volume of the compression candidates may incur huge computational cost to the ILP solver. This is also the reason that in Figure 1, for the system without pre-selection, we only vary n from 1 to 100. In general, we also notice that given more compression candidates, the R-2 score is still improving, as indicated by Figure 1. The improved performance of 'with pre-selection' over 'without pre-selection' is partly because fewer sentences are used and thus we are able to increase the number of compression candidates for these sentences in the ILP sentence extraction module.

Quality of sentence compression training data. In order to illustrate the contribution of our summary-guided sentence compression component, we train a generic sentence compression model and use this in our compression and summarization pipeline. The generic compression model was trained using the Edinburgh sentence compression corpus (Clarke and Lapata, 2008), which contains 1370 sentences collected from news articles. This data set has been widely used in other summarization studies (Martins and Smith, 2009). Each sentence has 3 compressions and we choose the shortest compression as the reference. The average compression rate of this corpus is about 28%, lower than that in our summary guided compression data. Note that in generic sentence compression, we only use those word and sentence features described in Section 4.1, not the document-level features since they are not available for the Edinburgh data set. Results of our system using the generic compression model (with sentence pre-selection) are shown in the last row of Table 3 and Table 4. We can see that the system with this generic compression model performs

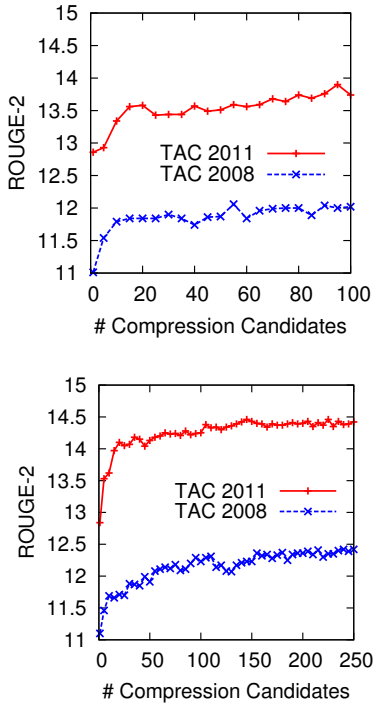


Figure 1: R-2 scores of the two systems (without and with the sentence pre-selection step) when using different number of compressions for each sentence.

worse than ours, and is also inferior to the TAC best performing system on both data sets, which signifies the importance of our proposed summary guided sentence compression approach. We can also see there is a difference in the compression ratio in the system generated compressions when using different compression corpora to train the compression models. The resulting compression ratio patterns are consistent with those in the training data, that is, using our guided compression corpus our system compressed sentences more aggressively.

Learning curve of guided compression. Since we use a supervised compression model, we further consider the relationship between the summarization performance and the number of sentence pairs used for training the guided compression model. In total, there are 1150 training sentence pairs in our corpus. We incrementally add 100 sentence pairs each time and plot the learning curve in Figure 2. In the compression step, we generate only the 1-best compression candidate in order to remove the impact caused by the downstream summary sentence selection module. As seen from Figure 2, increasing the compression training data generally improves summarization performance, although there are also fluctuations. When adding more training sentence pairs, the system performance is likely to further increase.

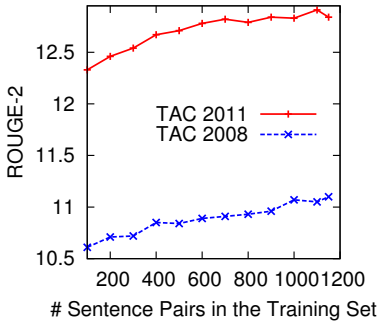


Figure 2: ROUGE-2 scores when using different number of sentences to train the guided compression model.

6 Conclusion and Future Work

In this paper, we propose a pipeline summarization approach that combines a novel *guided compression* model with ILP-based summary sentence selection. We create a guided compression corpus, where the human annotators were explicitly informed about the important summary words during the compression annotation. We then train a supervised compression model to capture the guided compression process using a set of word-, sentence-, and document-level features. We conduct experiments on the TAC 2008 and 2011 summarization data sets and show that by incorporating the guided sentence compression model, our summarization system can yield significant performance gain as compared to the state-of-the-art. In future, we would like to further explore the reinforcement relationship between keywords and summaries (Wan et al., 2007), improve the readability of the sentences generated from the guided compression system, and report results using multiple evaluation metrics (Nenkova et al., 2007; Louis and Nenkova, 2012) as well as performing human evaluations.

Acknowledgments

Part of this work was done during the first author's internship in Bosch Research and Technology Center. The work is also partially supported by NSF award IIS-0845484 and DARPA Contract No. FA8750-13-2-0041. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the funding agencies.

References

Ahmet Aker, Trevor Cohn, and Robert Gaizauskas. 2010. Multi-document summarization using a* search and discriminative training. In *Proceedings of EMNLP*.

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL*.

Yllias Chali and Sadid A. Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. In *Proceedings of COLING*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *Journal of Artificial Intelligence Research*.

Hal Daumé. 2006. Practical structured learning techniques for natural language processing. *Ph.D. thesis, University of Southern California*.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP*.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The icsi/utd summarization system at tac 2009. In *Proceedings of TAC*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of SIGIR*.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ilp for extractive summarization. In *Proceedings of ACL*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL*.

Chin-Yew Lin. 2003. Improving summarization performance by sentence compression - A pilot study. In *Proceeding of the Sixth International Workshop on Information Retrieval with Asian Language*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of ACL*.

Fei Liu and Yang Liu. 2009. From extractive to abstractive meeting summaries: Can it be done by sentence compression? In *Proceedings of ACL-IJCNLP*.

Fei Liu and Yang Liu. 2013. Towards abstractive speech summarization: Exploring unsupervised and supervised approaches for spoken utterance compression. *IEEE Transactions on Audio, Speech, and Language Processing*.

Annie Louis and Ani Nenkova. 2012. Automatically assessing machine summary content with a gold-standard. *Computational Linguistics*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing*.

Jun-Ping Ng, Praveen Bysani, Ziheng Lin, Min-Yen Kan, and Chew-Lim Tan. 2012. Exploiting category-specific information for multi-document summarization. In *Proceedings of COLING*.

Tadashi Nomoto. 2007. Discriminative sentence compression with conditional random fields. *Information Processing and Management*.

Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of EMNLP*.

Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. In *Computational Linguistics*.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of ACL*.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP-CoNLL*.

Katsumasa Yoshikawa, Tsutomu Hirao, Ryu Iida, and Manabu Okumura. 2012. Sentence compression with semantic role constraints. In *Proceedings of ACL*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. In *Information Processing and Management*.

Anchor Graph: Global Reordering Contexts for Statistical Machine Translation

Hendra Setiawan *
IBM Research
1101 Kitchawan Road
NY 10598, USA

Bowen Zhou
IBM Research
1101 Kitchawan Road
NY 10598, USA

Bing Xiang *
Thomson Reuters
3 Times Square
NY 10036, USA

Abstract

Reordering poses one of the greatest challenges in Statistical Machine Translation research as the key contextual information may well be beyond the confine of translation units. We present the "Anchor Graph" (AG) model where we use a graph structure to model global contextual information that is crucial for reordering. The key ingredient of our AG model is the edges that capture the relationship between the reordering around a set of selected translation units, which we refer to as *anchors*. As the edges link anchors that may span multiple translation units at decoding time, our AG model effectively encodes global contextual information that is previously absent. We integrate our proposed model into a state-of-the-art translation system and demonstrate the efficacy of our proposal in a large-scale Chinese-to-English translation task.

1 Introduction

Reordering remains one of the greatest challenges in Statistical Machine Translation (SMT) research as the key contextual information may span across multiple translation units.[1] Unfortunately, previous approaches fall short in capturing such cross-unit contextual information that could be critical in reordering. For example, state-of-the-art translation models, such as Hiero (Chiang, 2005) or Moses (Koehn et al., 2007), are good at capturing local reordering within the confine of a translation unit, but their formulation is approximately a simple unigram model over derivation (a sequence of the application of translation units) with some aid from target language models. Moving to a higher order formulation (say to a bigram model) is highly impractical for several reasons: 1) it has to deal with a severe *sparsity issue* as the size of the unigram model is already huge; and 2) it has to deal with a *spurious ambiguity issue* which allows multiple derivations of a sentence pair to have radically different model scores.

In this paper, we develop "*Anchor Graph*" (AG) where we use a graph structure to capture global contexts that are crucial for translation. To circumvent the sparsity issue, we design our model to rely only on contexts from a set of selected translation units, particularly those that appear frequently with important reordering patterns. We refer to the units in this special set as *anchors* where they act as vertices in the graph. To address the spurious ambiguity issue, we insist on computing the model score for every anchors in the derivation, including those that appear inside larger translation units, as such our AG model gives the same score to the derivations that share the same reordering pattern.

In AG model, the actual reordering is modeled by the edges, or more specifically, by the edges' labels where different reordering around the anchors would correspond to a different label. As detailed later, we consider two distinct set of labels, namely *dominance* and *precedence*, reflecting the two dominant views about reordering in literature, i.e. the first one that views reordering as a linear operation over a sequence and the second one that views reordering as a recursive operation over nodes in a tree structure The former is prevalent in phrase-based context, while the latter in hierarchical phrase-based and

* This work was done when the authors were with IBM.

[1] We define translation units as phrases in phrase-based SMT or as translation rules in syntax-based SMT.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 501–512,
Seattle, Washington, USA, 18-21 October 2013.

syntax-based context. More concretely, the dominance looks at the anchors' relative positions in the translated sentence, while the precedence looks at the anchors' relative positions in a latent structure, induced via a novel synchronous grammar: *Anchor-centric, Lexicalized Synchronous Grammar*.

From these two sets of labels, we develop two probabilistic models, namely the *dominance* and the *orientation* models. As the edges of AG link pairs of anchors that may appear in multiple translation units, our AG models are able to capture high order contextual information that is previously absent. Furthermore, the parameters of these models are estimated in an unsupervised manner without linguistic supervision. More importantly, our experimental results demonstrate the efficacy of our proposed AG-based models, which we integrate into a state-of-the-art syntax-based translation system, in a large scale Chinese-to-English translation task. We would like to emphasize that although we use a syntax-based translation system in our experiments, in principle, our approach is applicable to other translation models as it is agnostic to the translation units.

2 Anchor Graph Model

Formally, an AG consists of $\{\mathcal{A}, \mathcal{L}\}$ where $\mathcal{A}$ is a set of vertices that correspond to anchors, while $\mathcal{L}$ is a set of labeled edges that link a pair of anchors. In principle, our AG model is part of a translation model that focuses on the reordering within the source sentence F and its translation E. Thus, we start by first introducing $\mathcal{A}$ into a translation model (either word-based, phrase-based or syntax-based model) followed by $\mathcal{L}$. Given an F, $\mathcal{A}$ is essentially a subset of non-overlapping (word or phrase) units that make up F. As the information related to $\mathcal{A}$ is not observed, we introduce $\mathcal{A}$ as a latent variable.

Let $P(E, \sim |F)$ be a translation model where $\sim$ corresponds to the alignments between units in F and E. [2] We introduce $\mathcal{A}$ into a translation model, as follow:

$$P(E, \sim |F) = \sum_{\forall \mathcal{A}'} P(E, \sim, \mathcal{A}'|F) \quad (1)$$

$$P(E, \sim, \mathcal{A}'|F) = P(E, \sim |\mathcal{A}', F)P(\mathcal{A}') \quad (2)$$

As there can be many possible subsets of F and summing over all possible $\mathcal{A}$ is intractable, we make the following approximation for $P(\mathcal{A}')$ such that we only need to consider one particular $\mathcal{A}^*$: $P(\mathcal{A}') = \delta(\mathcal{A}' = \mathcal{A}^*)$ which returns 1 only for $\mathcal{A}^*$, otherwise 0. The exact definition of the heuristic will be described in Section 7, but in short, we equate $\mathcal{A}^*$ with units that appear frequently with important reordering patterns in training data.

Given an $\mathcal{A}^*$, we then introduce the edges of AG ($\mathcal{L}$) into the equation as follow:

$$P(E, \sim |\mathcal{A}^*, F) = P(E, \sim, \mathcal{L}|\mathcal{A}^*, F) \quad (3)$$

Note that $\mathcal{L}$ is also a latent variable but its values are derived deterministically from $(F, E, \sim)$ and $\mathcal{A}^*$, thus no extra summation is present in Eq. 3.

Then, we further simplify Eq. 3 by factorizing it with respect to each individual edges, as follow:

$$P(E, \sim, \mathcal{L}|\mathcal{A}^*, F) \approx \prod_{\substack{\forall a_m, a_n \in \mathcal{A}^* \\ m<n}} P(L_{m,n}|a_m, a_n) \quad (4)$$

where $L_{m,n} \in \mathcal{L}$ corresponds to the label of an edge that links a_m and a_n.

In principle, $L_{m,n}$ can take any arbitrary value. For addressing the reordering challenge, it should ideally correspond to some aspect of the reordering around a_m and a_n, for example, how the reordering around a_m affects the reordering around a_n. As mentioned earlier, we choose to associate $L_{m,n}$ with the dominance and the precedence relations between a_m and a_n, where the former looks at the relative positions of the two anchors when they are projected into a latent tree structure, while the latter looks at their relative positions when they are projected into the target sentence. We illustrate the two in Fig. 1.

Furthermore, we assume that dominance and precedence are independent and develop one model for each, resulting in the dominance and the orientation models, which we describe in Section 3 and 4 respectively. To make the model more compact, we

[2] Alignment ($\sim$) represents an existing latent variable. Depending on the translation units, it can be defined at different level, i.e. word, phrase or hierarchical phrase. As during translation, we are interested in the anchors that appear inside larger translation units, we set $\sim$ at word level, which information can be induced for (hierarchical) phrase units by either keeping the word alignment from the training data inside the units or inferring it via lexical translation probability. We use the former.

introduce an additional parameter O that restricts the maximum order of AG as follows:

$$\approx \prod_{o=1}^{O} \prod_{i=0}^{|\mathcal{A}^*|+o-1} P_o(L_{i-o,i}|a_{i-o}, a_i) \quad (5)$$

Thus, we only consider edges that link two anchors that are at most $O-1$ anchors apart. For $O=1$, the AG model only considers relations between neighboring anchors. Following the standard practice in the n-gram language modeling, we append O number of pseudo anchors at the beginning and at the end of F, which represent the sentence delimiter markers. We do so in a monotone order.

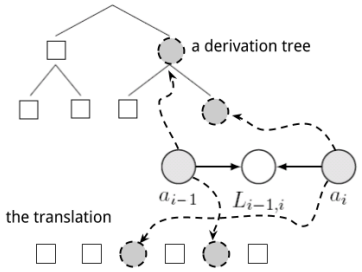


Figure 1: The illustration of the dominance and the precedence relations. The former looks at the anchors' projection on a derivation structure. The latter looks at the anchors' projection on the translated sentence.

3 Dominance Model

This section describes our dominance model where we equate $L_{m,n}$ in Eq. 4 with $dom(a_m, a_n)$ that expresses to the dominance relation between a_m and a_n in a latent tree structure. Due to reordering, anchors can only appear in specific nodes. We first describe a novel formalism of *Anchor-centric, Lexicalized Synchronous Grammar* (AL-SG), used to induce the tree structure and then discuss the probabilistic formulation of the model. Just to be clear, we introduce AL-SG mainly to facilitate the computation of $dom(a_m, a_n)$. The actual translation model at decoding time remains either phrase-based, hierarchical phrase-based or syntax-based model.

3.1 Anchor-centric, Lexicalized Synchronous Grammar

Given $(F, E, \sim)$ and $\mathcal{A}$, Anchor-centric, Lexicalized Synchronous Grammar (AL-SG) produces a tree structure where the nodes are decorated with anchors-related information. As the name alludes, the core of AL-SG is *anchor-centric constituents* (*ACC*), which corresponds to nodes, composed from merging anchors with by either their left, their right neighboring constituents or both.

More concretely, first of all, we consider a span on the source sentence F to be a *constituent* if it is consistent with the alignment ($\sim$). Second of all, we can construct a larger constituent by merging smaller constituents given that the larger constituent is also consistent with the alignment. These two constraints are similar to the heuristic applied to extract hierarchical phrases (Chiang, 2005).

Then, specific to AL-SG, we consider an anchor a to lexicalize a constituent c, if: a) we can compose c from at most three smaller constituents: c_L, a and c_R where a is the anchor while c_L,c_R are the (possibly empty) constituents immediately to the left and to the right of a; and b) we can create smaller anchors-centric constituents from concatenating a with c_L and a with c_R. If a can lexicalize c, then the node associated with c would be marked with a. In computing $dom(a_m, a_n)$, we look at the constituents that cover both anchors and check whether the anchors can lexicalized any of such constituents.

Now, we will describe AL-SG in a formal way. For simplicity, we use a simple grammar, called Inversion Transduction Grammar (ITG) (Wu, 1997), although in practice, we handle a more powerful synchronous grammar. Hence, we proceed to describe Anchor-centric, Lexicalized ITG (AL-ITG).

An AL-ITG is a quadruple $\{\Sigma, \mathcal{A}, \mathcal{V}, \mathcal{R}\}$ where:

- $\Sigma = \{(f/e)\}$ is a set of terminal symbols, which represents all possible units defined over $(F, E, \sim)$ where each pair corresponds to a link in $\sim$. We define $\sim$ at the most fine-grained level (i.e. word-level), as we insist on computing model score for each anchors even if they appear inside larger units.
- $\mathcal{A} \in \Sigma$ is a set of anchors, which is a subset of the terminal symbols.
- $\mathcal{V} = \{\{P, X, Y\} \times \{\mathcal{A}, \emptyset\}\}$ is a set of (possibly lexicalized) nonterminal symbols. P represents the terminal symbols (Σ); while X and Y correspond to the spans that are created from merging two adjacent constituents. On the tar-

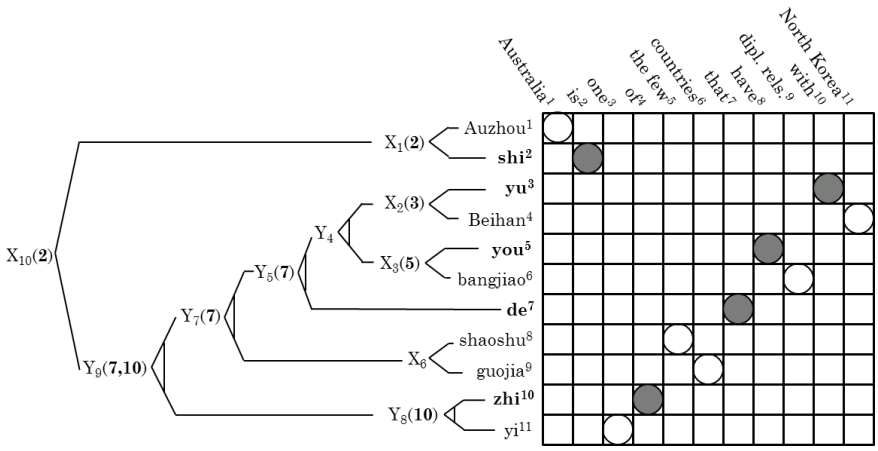


Figure 2: An illustration of an aligned Chinese-English sentence pair with one possible AL-ITG derivation obtained by applying the grammar in a left-to-right fashion. Circles represent alignment points. Black circle represents the anchor; boxes represent the anchor's neighbors. In the derivation tree, the anchors are represented by their position and in bold. For succinctness, we omit the preterminal rules in the tree.

get side, for X, the order of the two children follows the source order, while for Y, the order follows the inverse. Nonterminal symbols can be lexicalized with zero or more than one anchor. We represent a lexicalized constituent as a nonterminal symbol followed by a bracket which contains the lexicalizing anchors, e.g. $P(H)$ where H is the anchors lexicalizing P.

- $\mathcal{R}$ is a set of production rules which can be classified into the following categories:
  - Preterminal rules. We propagate the symbol if it corresponds to an anchor.

$$P(H = f/e) \rightarrow f/e, \text{ if } f/e \in \mathcal{A}^*$$
$$P(H = \emptyset) \rightarrow f/e, \text{ otherwise}$$

  - Monotone production rules, which reorder the children in monotone order, denoted by square brackets ("[","]").

$$\begin{aligned} X(H_1 \cup H_2) &\rightarrow [P(H_1)P(H_2)] \\ X(H_1 \cup H_2) &\rightarrow [X(H_1)P(H_2)] \\ X(H_1 \cup H_2) &\rightarrow [X(H_1)X(H_2)] \\ X(H_1) &\rightarrow [X(H_1)Y(H_2)] \\ X(H_2) &\rightarrow [Y(H_1)P(H_2)] \\ X(H_2) &\rightarrow [Y(H_1)X(H_2)] \\ X(\emptyset) &\rightarrow [Y(H_1)Y(H_2)] \end{aligned}$$

  - Inverse production rules, which reorder the children in the inverse order, denoted by angle brackets ("⟨","⟩").

$$\begin{aligned} Y(H_1 \cup H_2) &\rightarrow \langle P(H_1)P(H_2)\rangle \\ Y(H_1 \cup H_2) &\rightarrow \langle Y(H_1)P(H_2)\rangle \\ Y(H_1 \cup H_2) &\rightarrow \langle Y(H_1)Y(H_2)\rangle \\ Y(H_1) &\rightarrow \langle Y(H_1)X(H_2)\rangle \\ Y(H_2) &\rightarrow \langle X(H_1)P(H_2)\rangle \\ Y(H_2) &\rightarrow \langle X(H_1)Y(H_2)\rangle \\ Y(\emptyset) &\rightarrow \langle X(H_1)X(H_2)\rangle \end{aligned}$$

Like ITG, AL-ITG only permits two kind of reordering operations, namely monotone and inverse. To accommodate the lexicalization, we first assign a unique nonterminal symbol for each, i.e. X for monotone reordering and Y for inverse reordering. Then, we lexicalize Xs and Ys with anchors as long as they satisfy the constraint that the child shares the same label as the parent. This constraint guarantees that the constituents are valid ACCs. It also enables the anchors to lexicalize long constituents, although the terminal symbols are defined at word-level.

Fig. 2 illustrates an example Chinese-to-English translation with a AL-ITG derivation when the grammar is applied in a left-to-right fashion. Admittedly, AL-ITG (or more generally AL-SG) is susceptible to spurious ambiguity as it produces multiple derivation trees for a given $(F, E, \sim)$. Fortunately, the value of $dom(a_m, a_n)$ is identical for all derivations, since the computation of $dom(a_m, a_n)$ relies

only on whether a_m and a_n can lexicalize *at least one* constituent that covers both anchors. Hence, we only need to look at one derivation to compute $dom(a_m, a_n)$. Generalizing AL-ITG to a more powerful formalism is trivial; we just need to forbid the propagation for non-binarizeable production rules.

3.2 Probabilistic Model

We read-off the dominance relations $dom(a_m, a_n)$ from D obtained from the application of AL-SG to $(F, E, \sim)$. As lexicalization is a bottom-up process, for reading-off $dom(a_m, a_n)$, it is sufficient to look at the lowest common ancestor (LCA) of both anchors; if the anchors cannot lexicalize the LCA, they won't be able to lexicalize the constituents larger than LCA. To be more concrete, let's consider the D in Fig. 2. In that D, the LCA of a_m = yu^3/with^{10} and a_n = de^7/that^7 is Y_5(**7**). Then, we check the anchors that can lexicalize the LCA. Let $V(H)$ be the LCA, then $dom(a_m, a_n) \in$

$$
\begin{aligned}
(\textit{LH}) \quad & \text{, if } a_m \in H \wedge a_n \notin H \\
(\textit{RH}) \quad & \text{, if } a_m \notin H \wedge a_n \in H \\
(\textit{BL}) \quad & \text{, if } a_m \in H \wedge a_n \in H \\
(\textit{BD}) \quad & \text{, if } a_m \notin H \wedge a_n \notin H
\end{aligned}
$$

The value refers to cases where a_m and a_n can lexicalize $V(H)$ and it is useful to model spans that share a simple, uniform reordering, i.e. all-monotone or all-inverse, while the value refers to the cases where a_m and a_n cannot lexicalize $V(H)$ and it is useful to model spans that involve in a complex reordering. Meanwhile, the and refer to cases where only one anchor can lexicalize $V(H)$, i.e. a_m and a_n respectively. These values are useful for modeling cases where the surroundings of the two anchors exhibit different kind of reordering pattern.

With such definition, the edge labels $\mathcal{L}$ in Fig. 2 are indicated in Table 1. Note that in Table 1, we don't specify the relations involving pseudo anchors, although they are crucial.

The final probabilistic formulation of the dominance model is as follows:

$$
\approx \prod_{o=1}^{O} \prod_{i=0}^{|\mathcal{A}|+o-1} P_{dom_o}(dom(a_{i-o}, a_i) | a_{i-o}, a_i) \quad (6)
$$

As shown, we allocate a separate model P_{dom_o} for each separate order (o) where each P_{dom_o} will contribute as one additional feature in the log-linear model of the translation model. In allocating a separate model for each o, we conjecture that different pair of anchors contributes differently depending on how far the two anchors are.

m / n	1	2	3	4	5
1 = (shi^2/is^2)	-	-	-	-	-
2 = (yu^3/with^{10})	*LH*	-	-	-	-
3 = (you^5/have^8)	*LH*	*BD*	-	-	-
4 = (de^7/that^7)	*LH*	*RH*	*RH*	-	-
5 = (zhi^{10}/of^4)	*LH*	*RH*	*RH*	*BL*	-

Table 1: The dominance relations between pairs of anchors according to the derivation in Fig. 2.

4 Orientation Model

In this section, we introduce the orientation model (*ori*) where we equate $L_{m,n}$ with the precedence relations between a pair of anchors. Instead of directly modeling the precedence between the two anchors, we approximate it by modeling the precedence of each anchor with its neighboring constituents. Formally, we approximate $P(L_{m,n}|a_m, a_n)$ as

$$
\begin{aligned}
& P_{ori_R}(ori(a_m, M_R(a_m))|a_m) \times \\
& \quad P_{ori_L}(ori(a_n, M_L(a_n))|a_n) \qquad (7)
\end{aligned}
$$

where $M_R(a_m)$ is the largest constituent to the right of the first anchor a_m, $M_L(a_n)$ the largest constituent to the left of the second anchor a_n, and $ori()$ a function that maps the anchor and the neighboring constituent to a particular orientation.

Plugging Eq. 7 into Eq. 5 results in the following approximation of $P(\Theta|\mathcal{A})$:

$$
\begin{aligned}
C. \prod_{i=0}^{|\mathcal{A}|-1} & \{P_{ori_L}(ori(a_i, M_L(a_i))|a_i) \times \\
& P_{ori_R}(ori(a_i, M_R(a_i))|a_i)\}^O \qquad (8)
\end{aligned}
$$

where C is a constant term related to the pseudo anchors and O is the maximum order of the AG. In practice, we can safely ignore both C and O as they are constant for a given AG. As shown, the orientation model is simplified into a model that looks at the reordering of the anchors' neighboring constituents.

The exact definition of M_L and M_R will be discussed in Section 5. Their orientation, i.e.

$ori_L(C_L, a)$ and $ori_R(C_R, a)$ respectively, may take one of the following four values: (*MA*), (*RA*), (*MG*) and (*RG*). The first clause (monotone, reverse) indicates whether the target order follows the source order; the second (adjacent, gap) indicates whether the anchor and its neighboring constituent are adjacent or separated by an intervening when projected.

5 Parameter Estimation

For each $(F, E, \sim)$, the training starts with the identification of the regions in the source sentences as anchors ($\mathcal{A}$). For our Chinese-English experiments, we use a simple heuristic that equates anchors ($\mathcal{A}^*$) with constituents whose corresponding word class belongs to function words-related classes, bearing a close resemblance to (Setiawan et al., 2007). In total, we consider 21 part-of-speech tags; some of which are as follows: VC (copula), DEG, DEG, DER, DEV (*de*-related), PU (punctuation), AD (adjectives) and P (prepositions).

5.1 Extracting Events from $(F, E, \sim)$

The parameter estimation first involves extracting two statistics from $(F, E, \sim)$, namely $dom(a_m, a_n)$ for the dominance model as well as $ori(a, M_L(a))$ and $ori(a, M_R(a))$ for the orientation model. Instead of developing a separate algorithm for each, we describe a unified way to extract these statistics via the largest neighboring constituents of the anchors, i.e. $M_L(a)$ and $M_R(a)$. This approach enables the dominance model to share the same residual state information as the orientation model.[3]

Let a_m be an anchor and $M_R(a_m)$ be its largest neighboring constituent to the right. Let a_n be an anchor to the left of a_m and $M_L(a_n)$ be a_n's largest neighboring constituent to the left. According to AL-SG, we say that a_m dominates a_n if $ori(a_m, M_R(a_m)) \in \{MA, RA\}$ and $a_n \in M_R(a_m)$. By the same token, we say that a_n dominates a_m if $ori(a_n, M_L(a_n)) \in \{MA, RA\}$ and $a_m \in M_L(a_n)$. The constraints on the orientation reflect the fact that in AL-SG, anchors can only be propagated through monotone or inverse production rules, which correspond to the MA and RA respectively. The fact that we are looking at the largest neighboring constituents guarantees that if the other anchor is outside that constituent, then that other anchor is never dominated.

More formally, given an aligned sentence pair $\Theta = (F, E, \sim)$, let $\Delta(\Theta)$ be all possible constituents that can be extracted from Θ:[4]

$$\{(f_{j_1}^{j_2}/e_{i_1}^{i_2}) : \forall (j,i) \in \sim : ((j_1 \leq j \leq j_2) \wedge (i_i \leq i \leq i_2)) \vee (\neg(j_1 \leq j \leq j_2) \wedge \neg(i_i \leq i \leq i_2))$$

Then, let the anchors $\mathcal{A}$ be a subset of $\Delta(\Theta)$. Given $\mathcal{A} \subset \Delta(\Theta)$, let $a = (f_{j_1}^{j_2}/e_{i_1}^{i_2}) \in \mathcal{A}$ be a particular anchor. And, let $\mathcal{C}_L(a) \subset \Delta(\Theta)$ be a's left neighbors and let $\mathcal{C}_R(a) \subset \Delta(\Theta)$ be a's right neighbors, iff:

$$\forall C_L = (f_{j_3}^{j_4}/e_{i_3}^{i_4}) \in \mathcal{C}_L(a) : j_4 + 1 = j_1$$
$$\forall C_R = (f_{j_5}^{j_6}/e_{i_5}^{i_6}) \in \mathcal{C}_R(a) : j_2 + 1 = j_5$$

Then, let $M_L(a) \in \mathcal{C}_L(a)$ and $M_R(a) \in \mathcal{C}_R(a)$ be the largest left and right neighbors according to:

$$M_L(a) = \underset{(f_{j_3}^{j_4}/e_{i_3}^{i_4}) \in \mathcal{C}_L(a)}{\arg\max} (j_4 - j_3)$$
$$M_R(a) = \underset{(f_{j_5}^{j_6}/e_{i_5}^{i_6}) \in \mathcal{C}_R(a)}{\arg\max} (j_6 - j_5)$$

Let $M_L = (f_{j_3}^{j_4}/e_{i_3}^{i_4})$ and $M_R = (f_{j_5}^{j_6}/e_{i_5}^{i_6})$. We then proceed to extract $ori_L(a, M_L(a))$ and $ori_R(a, M_R(a))$ respectively as follows:

- *MA*, if $(i_4 + 1) = i_1$ for ori_L or if $(i_2 + 1) = i_5$ for ori_R
- *RA*, if $(i_2 + 1) = i_3$ for ori_L or if $(i_6 + 1) = i_1$ for ori_R
- *MG*, if $(i_4 + 1) < i_1$ for ori_L or if $(i_2 + 1) < i_5$ for ori_R
- *RG*, if $(i_2 + 1) < i_3$ for ori_L or if $(i_6 + 1) < i_1$ for ori_R.

Then, we proceed to extract $dom(a_m, a_n)$. Given two anchors a_m, a_n where $m < n$, we define the

[3] The analogy in an n-gram language model is the first $n-1$ words of the hypothesis that have incomplete history.

[4] We represent a constituent as a source and target phrase pair $(f_{j_1}^{j_2}/e_{i_1}^{i_2})$ where the subscript and the superscript indicate the starting and the ending indices as such $f_{j_1}^{j_2}$ denotes a source phrase that spans from j_1 to j_2.

dominance relation between a_m and a_n via $M_R(a_m)$ and $M_L(a_n)$. Let $a_m = (f_{j_1}^{j_2}/e_{i_1}^{i_2})$, $M_R(a_m) = (f_{j_3}^{j_4}/e_{i_3}^{i_4})$, $a_n = (f_{j_5}^{j_6}/e_{i_5}^{i_6})$ and $M_L(a_n) = (f_{j_7}^{j_8}/e_{i_7}^{i_8})$. Then, $ldom(a_m, a_n)$ is true only if $(j_4 \geq j_6)$ and $ori_R(a_m, M_R(a_m)) \in \{MA, RA\}$. Similarly, $rdom(a_m, a_n)$ is true only if $(j_7 \leq j_1)$ and $ori_L(a_n, M_L(a_n)) \in \{MA, RA\}$.

Hence, $dom(a_m, a_n)$ is as follows:

- *LH*, if $ldom(a_m, a_n) \wedge \neg rdom(a_m, a_n)$
- *RH*, if $\neg ldom(a_m, a_n) \wedge rdom(a_m, a_n)$
- *BL*, if $ldom(a_m, a_n) \wedge rdom(a_m, a_n)$
- *BD*, if $\neg ldom(a_m, a_n) \wedge \neg rdom(a_m, a_n)$

5.2 Parameterization and Training

After extracting events, we are now ready to train the models. To estimate them, we train a discriminative classifier for each model and use the normalized posteriors at decoding time as additional feature scores in SMT's log-linear framework.

At a high level, we use a rich set of *binary* features ranging from lexical to part-of-speech (POS) and to syntactic features. Additionally, we augment the feature set with compound features, e.g. a conjunction of the source word of the left anchor and the source word of the right anchor. Although they increase the number of features significantly, we found that they are empirically beneficial.

Suppose $a = (f_{j_1}^{j_2}/e_{i_1}^{i_2})$, $M_L(a) = (f_{j_3}^{j_4}/e_{i_3}^{i_4})$ and $M_R(a) = (f_{j_5}^{j_6}/e_{i_5}^{i_6})$, then based on the context's location, the elementary features employed in our classifiers can be categorized into:

- *anchor-related*: (the actual word of $f_{j_1}^{j_2}$), (part-of-speech (POS) tag of), ('s parent in the parse tree), ($e_{i_1}^{i_2}$'s actual target word).
- *surrounding*: (the previous word / $f_{j_1-1}^{j_1-1}$), (the next word / $f_{j_2+1}^{j_2+1}$), ('s POS tag), ('s POS tag), ('s parent), ('s parent).
- *non-local*: (the previous anchor's source word) , (the next anchor's source word), ('s POS tag), ('s POS tag).

There is a separate set of elementary features for a_m and a_n and we come up with manual combination to construct compound features.

In training the models, we manually come up with around 30-50 types of features, which consists of a combination of elementary and compound features. Due to space constraints, we will describe the actual features that we use and the classification performance of our models elsewhere. In total, we generate around one hundred millions binary features from our training data that contains six million sentence pairs. To reduce the number of features, we employ the L1-regularization in training to enforce sparse solutions, using the off-the-shelf LIBLINEAR toolkit (Fan et al., 2008). After training, the number of features in our classifiers decreases to below 1 million features for each classifier.

6 Decoding

As mentioned earlier, we wish to avoid the spurious ambiguity issue where different derivations have radically different scores although they lead to the same reordering. This section describes our decoding algorithm that avoids spurious ambiguity issue by incrementally constructing M_Ls and M_Rs thus allowing the computation of the models over partial hypotheses.

In our experiments, we integrate our dominance model as well as our orientation model into a syntax-based SMT system that uses SCFG formalism. Integrating the models into syntax-based SMT systems is non-trivial, especially since the anchors often reside within translation rules and the model doesn't always decompose naturally with the hypothesis structure. To facilitate that, we need to first induce the necessary alignment for all translation units in the hypothesis.

To describe the algorithm, let us consider a cheating exercise where we have to translate the Chinese sentence in Fig. 2 with the following set of hierarchical phrases:

$X_a \rightarrow \langle$*Aozhou*[1] *shi*[2] X_1, Australia[1] is[2] $X_1\rangle$
$X_b \rightarrow \langle$*yu*[3] *Beihan*[4] X_1, $X_1$with[3] North[4] Korea$\rangle$
$X_c \rightarrow \langle$*you*[5] *bangjiao*[6], have[5]dipl.[6] rels.$\rangle$
$X_d \rightarrow \langle X_1$ ***de***[7] *shaoshu*[8] *guojia*[9] *zhi*[10] *yi*[11],
one[11]of[10]the few[8] countries[9] **that**[7] $X_1\rangle$

As a case in point, let us consider $D = X_a \prec X_b \prec X_d \prec X_c$, which will lead to the correct English

		Target string (w/ source index)	Symbol(s) read	Op.	Stack(s)
(1)	X_c	have[5] dipl.[6] rels.	[5][6]	S,S,R	X_c:[5-6]
(2)	X_d	one[11] of[10] few[8] countries[9] **that**[7] X_c	[11][10]	S,S,R	[10-11]
(3)			[8][9]	S,S,R,R	[8-11]
(4)			[7]	S	[8-11]**[7]**
(5)			X_c:[5,6]	S	X_d:[8-11]**[7]**[5,6]
(6)	X_b	X_d with[3] North[4] Korea	X_d:[8-11]**[7]**[5,6]	S	[8-11]**[7]**[5,6]
(7)			[3][4]	S,S,R,R	X_b:[8-11]**[7]**[3-6]
(8)	X_a	Australia[1] is[2] X_b	[1][2]	S,S,R	[1-2]
(9)			X_b:[8-11]**[7]**[3,6]	S,A	X_a:[1-2][8-11]**[7]**[3,6]

Table 2: The application of the shift-reduce parsing algorithm, which corresponds to the following derivation $D = X_a \prec X_b \prec X_d \prec X_c$. Anchor is in **bold**. In column Op., S, R and A refer to shift, reduce and accept operation respectively.

translation as in Fig. 2. Note that the translation rules contain internal word alignment, which we assume to have been previously inferred.

The algorithm bears a close resemblance to the shift-reduce algorithm found in phrase-based decoding (Galley and Manning, 2008; Feng et al., 2010; Cherry et al., 2012). A stack is used to accumulate (partial) information about a, M_L and M_R for each $a \in \mathcal{A}$ in the derivation. This algorithm takes an input stream and applies either the *shift* or the *reduce* operations starting from the beginning until the end of the stream. The *shift* operation advances the input stream by one symbol and push the symbol into the stack; while the *reduce* operation applies some rule to the top-most elements of the stack. The algorithm terminates at the end of the input stream where the resulting stack will be propagated to the parent for the later stage of decoding. In our case, the input stream is the target string of the rule and the symbol is the corresponding source index of the elements of the target string. The reduction rule looks at two indices and merge them if they are adjacent (i.e. has no intervening phrase). We forbid the application of the reduction rule to anchors. Table 2 shows the execution trace of the algorithm for the derivation described earlier. For conciseness, we assume that there is only one anchor and that is de[7]/that[7].

As shown, the algorithm starts with an empty stack. It then projects the source index to the corresponding target word and then enumerates the target string in a left to right fashion. If it finds a target word with a source index, it applies the shift operation, pushing the index to the stack. Unless the symbol corresponds to an anchor, it tries to apply the reduce operation. Line (4) indicates the special treatment to the anchor. If the symbol being read is a nonterminal, then we push the entire stack that corresponds to that nonterminal. For example, when the algorithm reads X_d at line (6), it pushes the entire stack from line (5).

As M_Ls and M_Rs are being incrementally constructed, we can immediately compute $P_{dom_o}(dom(a_m, a_n)|a_m, a_n)$ as soon as a partial derivation covers both a_m and a_n. For example, we can compute $P_{dom_1}(dom(\text{you}_5/\text{have}_8, \text{de}_7/\text{that}_7) =)$, $P_{dom_1}(dom(\text{de}_7/\text{that}_7, \text{zhi}_{10}/\text{of}_4) =)$ and $P_{dom_2}(dom(\text{you}_5/\text{have}_8, \text{zhi}_{10}/\text{of}_4) =)$ at partial hypothesis $X_d \prec X_c$ which corresponds to a constituent spanning from 5-11.

7 Experiments

Our baseline systems is a state-of-the-art string-to-dependency system (Shen et al., 2008). The system is trained on 10 million parallel sentences that are available to the Phase 1 of the DARPA BOLT Chinese-English MT task. The training corpora include a mixed genre of newswire, weblog, broadcast news, broadcast conversation, discussion forums and comes from various sources such as LDC, HK Law, HK Hansard and UN data.

In total, our baseline model employs more than 50 features, including from our proposed dominance and orientation models. In addition to the standard

Model		newswire			weblog			newswire+weblog		
		BLEU (a)	TER (b)	Comb (c)	BLEU (d)	TER (e)	Comb (f)	BLEU (g)	TER (h)	Comb (i)
(1)	S2D	37.63	53.17	7.77	27.60	57.19	14.77	33.39	54.97	10.79
(2)	+dom_1	38.12	52.31	7.10	27.56	56.58	14.51	33.64	54.24	10.30
(3)	+dom_2	38.31	52.28	6.99	27.66	56.57	14.45	33.78	54.20	10.21
(4)	+dom_3	38.31	52.52	7.10	28.24	56.56	14.16	34.02	54.33	10.15
(5)	+dom_4	*38.54*	*52.22*	*6.84*	28.38	56.55	14.08	*34.20*	*54.16*	*9.98*
(6)	+dom_5	38.17	52.57	7.20	*28.67*	56.27	13.80	34.16	54.27	10.05
(7)	+dom_6	38.17	52.52	7.18	28.64	*56.22*	*13.79*	34.10	54.18	10.04
(8)	+*ori*	38.52	52.43	6.96	28.26	56.54	14.14	34.15	54.27	10.06
(9)	+*ori*+dom_1	38.87	52.05	6.59	28.01	56.48	14.23	34.26	54.03	9.89
(10)	+*ori*+dom_2	38.96	51.87	6.45	27.98	56.23	14.12	34.29	53.82	9.77
(11)	+*ori*+dom_3	39.19	51.77	6.29	28.19	56.15	13.98	34.52	53.73	9.61
(12)	+*ori*+dom_4	**39.34**	51.77	6.21	28.41	56.17	13.88	34.60	53.69	9.54
(13)	+*ori*+dom_5	39.31	**51.67**	**6.18**	**28.62**	**56.09**	**13.74**	**34.76**	**53.65**	**9.45**

Table 3: The NIST MT08 results on newswire (nw), weblog (wb) and combined genres. S2D is the baseline string-to-dependency system (line 1). Lines 2-7 shows the results of the dominance model with $O = 1 - 6$. Line 8 shows result on adding ori to the baseline. Lines 9-13 shows the results of the orientation complemented with the dominance model with varying O. The best BLEU, TER and Comb on each genre of the first set are in *italic* while those of the second set are in **bold**. For BLEU, higher scores are better, while for TER and Comb, lower scores are better.

features such as translation probabilities, we incorporate features that are found useful for developing a state-of-the-art baseline, such as the provenance features (Chiang et al., 2011). We use a 6-gram language model, which was trained on 10 billion English words from multiple corpora, including the English side of our parallel corpus plus other corpora such as Gigaword (LDC2011T07) and Google News. We also train a class-based language model (Chen, 2009) on two million English sentences selected from the parallel corpus. As for our string-to-dependency system, we train 3-gram models for left and right dependencies and unigram for head using the target side of the parallel corpus. To train our models, we select a set of 5 million sentence pairs.

For the tuning and development sets, we set aside 1275 and 1239 sentences selected from LDC2010E30 corpus. We tune the feature weights with PRO (Hopkins and May, 2011) to minimize (TER-BLEU)/2 metric. As for the blind test set, we report the performance on the NIST MT08 evaluation set, which consists of 691 sentences from newswire and 666 sentences from weblog. We pick the weights that produce the highest development set scores to decode the test set.

We perform two sets of experiments. The first set looks at the contribution of the dominance model with varying values of o. The second one looks at the combination of the dominance model and the orientation model. Table 3 summarizes the experimental results on NIST MT08 sets, categorized by genres. We report the results on newswire genre in columns a-c, those on weblog genre in column d-f, and those on mixed genre in column g-i. The performance of our baseline string-to-dependency syntax-based SMT is shown in the first line.

Lines 2-7 in Table 3 show the results of our first set of experiments, starting from the result of dom_1, which looks at only at pairs of adjacent anchors, to the result of dom_6, which looks at pairs of anchors that are at most 5 anchors away. As shown in line 2, our dominance model provides a nice improvement of around 0.5 point over the baseline even if it only looks at restricted context. Increasing the order of our dominance model provides an additional gain. However, the gain is more pronounced in the weblog genre (up to around 1 BLEU point) than in the newswire genre. We conjecture that this may be the artifact of our tune set, which comes from the weblog genre. We stop at dom_6 because we observe

that the weight of the feature score that corresponds to the maximum order ($o = 6$) has a negative sign, which often indicates a high correlation between the new features and existing ones.

Lines 8-13 in Table 3 shows the results of our second set of experiments. Line 8 shows the result of adding the orientation model (ori) to the baseline system. As shown, integrating ori shows a significant gain. On top of which, we then integrate dom_1 to dom_5. We see a very encouraging result as adding the dominance model increases the performance further, consistently over different value of o. This suggests that the dominance model is complementary to the orientation model. Our best result provides more than 1 BP improvement and 1 TER reduction consistently over different genres. We see this result as confirming our intuition that the global contextual information provided by our AG model can significantly improve the performance of SMT even in a state-of-the-art system.

8 Related Work

Our work intersects with existing work in many different respects. In this section, we mainly focus on work related to introducing higher-order contextual information to reordering model.

In providing global contextual information, our work is related to a large amount of literature. To name a few, Zens and Ney (2006) improves the lexicalized reordering model of Tillman (2004) by incorporating part-of-speech information. Chang et al. (2009) incorporates contexts from syntactic parse tree. Bach et al. (2009) exploits the dependency information and Xiong et al. (2012) uses the predicate-argument structure.

Vaswani et al. (2011) introduces rule markov models for a forest-to-string model in which the number of possible derivations is restricted. More recently, Durrani et al. (2013) and Zhang et al. (2013) cast reordering process as a Markov process. Similar to these models, our proposed model also provide context dependencies to the application of translation rules, however, as they focus on minimal translation units (MTU) where we focus on a selected set of translation units. (Banchs et al., 2005) introduces a bigram model for monotone phrase-based system, but their definition of translation units is suitable only for language pairs with limited reordering, such as translating Spanish to English.

In equating anchors with the function word class, our work is closely related to the function word-centered model of Setiawan et al. (2007), especially the orientation model. Our dominance model is closely related to the reordering model of Setiawan et al. (2009), except that they only look at pair of adjacent anchors, forming a chain structure instead of a graph like in our dominance model. Furthermore, we provide a discriminative treatment to the model to include a richer set of features including syntactic features. This work can be seen as modeling the identity of the neighboring of the anchors, similar to (Setiawan et al., 2013). However, instead of looking at the words at the borders, we look at whether the neighboring constituents contain other anchors.

9 Conclusion

We propose the "Anchor Graph" (AG) model to encode global contextual information. A selected set of translation units, which we call anchors, serves as the vertices of AG. And as the edges, we model two types of relations, namely the dominance and the precedence relations, where the former looks at the positions of the anchors in the derivation structure, while the latter looks at the positions of the anchors in the surface structure, resulting into two probabilistic models over edge labels. As the models look at the pairs of anchors that go beyond multiple translation units, our AG model provides global contextual information.

Our AG model embodies (admittedly crudely) some basic principles of sentence organization, namely categorization (in categorizing units into anchors and non-anchors), linear order (in modeling the precedence of anchors) and constituency structure (in modeling the dominance between anchors). We are encouraged by the facts that we learn these principles in an unsupervised way and that we can achieve a significant improvement over a strong baseline in a large-scale Chinese-to-English translation task. In the future, we hope to continue this line of research, perhaps by learning to identify anchors automatically from training data or by using our models to induce derivations directly from unaligned sentence pair.

Acknowledgements

We would like to acknowledge the support of DARPA under Grant HR0011-12-C-0015 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the DARPA.

References

Nguyen Bach, Qin Gao, and Stephan Vogel. 2009. Source-side dependency tree reordering models with subtree movements and constraints. In *Proceedings of the Twelfth Machine Translation Summit (MTSummit-XII)*, Ottawa, Canada, August. International Association for Machine Translation.

Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, and José B. Mariño. 2005. Statistical machine translation of Euparl data by using bilingual n-grams. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 133–136, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 51–59, Boulder, Colorado, June. Association for Computational Linguistics.

Stanley Chen. 2009. Shrinking exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 468–476, Boulder, Colorado, June. Association for Computational Linguistics.

Colin Cherry, Robert C. Moore, and Chris Quirk. 2012. On hierarchical re-ordering and permutation parsing for phrase-based decoding. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 200–209, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang, Steve DeNeefe, and Michael Pust. 2011. Two easy improvements to lexical weighting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 455–460, Portland, Oregon, USA, June. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013. Model with minimal translation units, but decode with phrases. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Atlanta, Georgia, June. Association for Computational Linguistics.

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

Yang Feng, Haitao Mi, Yang Liu, and Qun Liu. 2010. An efficient shift-reduce decoding algorithm for phrased-based machine translation. In *Coling 2010: Posters*, pages 285–293, Beijing, China, August. Coling 2010 Organizing Committee.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation, June.

Hendra Setiawan, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 712–719, Prague, Czech Republic, June. Association for Computational Linguistics.

Hendra Setiawan, Min Yen Kan, Haizhou Li, and Philip Resnik. 2009. Topological ordering of function words in hierarchical phrase-based translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 324–332, Suntec, Singapore, August. Association for Computational Linguistics.

Hendra Setiawan, Bowen Zhou, Bing Xiang, and Libin Shen. 2013. Two-neighbor orientation model with cross-boundary global contexts. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

1264–1274, Sofia, Bulgaria, August. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. 2011. Rule markov models for fast tree-to-string translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 856–864, Portland, Oregon, USA, June. Association for Computational Linguistics.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, Sep.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 902–911, Jeju Island, Korea, July. Association for Computational Linguistics.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL): Proceedings of the Workshop on Statistical Machine Translation*, pages 55–63, New York City, NY, June. Association for Computational Linguistics.

Hui Zhang, Kristina Toutanova, Chris Quirk, and Jianfeng Gao. 2013. Beyond left-to-right: Multiple decomposition structures for smt. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 12–21, Atlanta, Georgia, June. Association for Computational Linguistics.

Source-Side Classifier Preordering for Machine Translation

Uri Lerner
Google Inc.
Mountain View, CA, USA
uri@google.com

Slav Petrov
Google Inc.
New York, NY, USA
slav@google.com

Abstract

We present a simple and novel classifier-based preordering approach. Unlike existing preordering models, we train feature-rich discriminative classifiers that directly predict the target-side word order. Our approach combines the strengths of lexical reordering and syntactic preordering models by performing long-distance reorderings using the structure of the parse tree, while utilizing a discriminative model with a rich set of features, including lexical features. We present extensive experiments on 22 language pairs, including preordering into English from 7 other languages. We obtain improvements of up to 1.4 BLEU on language pairs in the WMT 2010 shared task. For languages from different families the improvements often exceed 2 BLEU. Many of these gains are also significant in human evaluations.

1 Introduction

Generating the appropriate word order for the target language has been one of the fundamental problems in machine translation since the ground setting work of Brown et al. (1990). Lexical reordering approaches (Tillmann, 2004; Zens and Ney, 2006) add a reordering component to standard phrase-based translation systems (Och and Ney, 2004). Because the reordering model is trained discriminatively, it can use a rich set of lexical features. However, it only has access to the local context which often times is insufficient to make the long-distance reordering decisions that are necessary for language pairs with significantly different word order.

Preordering (sometimes called pre-reordering or simply reordering) approaches (Xia and McCord, 2004; Collins et al., 2005) preprocess the input in such a way that the words on the source side appear closer to their final positions on the target side. Because preordering is performed prior to word alignment, it can improve the alignment process and can then be combined with any subsequent translation model. Most preordering models use a source-side syntactic parser and perform a series of tree transformations. Approaches that do not use a parser exist as well and typically induce a hierarchical representation that also allows them to perform long-distance changes (Tromble and Eisner, 2009; DeNero and Uszkoreit, 2011; Neubig et al., 2012).

Models that use a source-side parser differ on two main dimensions: the way tree transformations are expressed, and whether they are built manually or learned from data. One common type of tree transformation are rewrite rules. These typically involve some condition under which the transformation can be applied (e.g., a noun and an adjective found in the same clause) and the transformation itself (e.g., move the adjective after the noun). These rules can be designed manually (Collins et al., 2005; Wang et al., 2007) or learned from data (Xia and McCord, 2004; Habash, 2007; Genzel, 2010; Wu et al., 2011).

Another type of tree transformations uses ranking functions to implement precedence-based reordering. Here, a function assigns a numerical value to every word in a clause, intended to express the precedence of the word in the target language. The reordering operation is then to sort the words according to their assigned values. The ranking function

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 513–523,
Seattle, Washington, USA, 18-21 October 2013.

can be designed manually (Xu et al., 2009) or trained from data (Yang et al., 2012). This approach is particularly effective for Subject-Object-Verb (SOV) languages.

In this work we present a simple classifier-based preordering model. Our model operates over dependency parse trees and is therefore able to perform long-distance reordering decisions, as is typical for preordering models. But instead of deterministic rules or ranking functions, we use discriminative classifiers to directly predict the final word order, using rich (bi-)lexical and syntactic features.

We present two models. The first model uses a classifier to directly predict the permutation order in which a family of words (a head word and all its children) will appear on the target side. This approach is similar in spirit to the work of Li et al. (2007), except that they use constituency parse trees and consider only nodes with 2 or 3 children. We instead work with dependency trees and consider much larger head-children sets. Our second model is designed to decompose the exponential search space of all possible permutations. The prediction task is broken into two separate steps. In the first step, for each child word a binary classifier decides whether it appears before or after its parent in the target language. In the second step, we predict the best order of the words on each side of the parent. We show that the second approach is never worse than the first one and sometimes significantly better.

We present experiments on 22 language pairs from different language families using our preordering approach in a phrase-based system (Och and Ney, 2004), as well as a forest-to-string system (Zhang et al., 2011). In a first set of experiments, we use the WMT 2010 shared task data (Callison-Burch et al., 2010) and show significant improvements of up to 1.4 BLEU (Papineni et al., 2002) on three out of eight language pairs. In a second set of experiments, we use automatically mined parallel data from the web and build translation systems for languages from various language families. We obtain especially big improvements in translation quality (2-7 BLEU) when the language pairs have divergent word order (for example English to Indonesian, Japanese, Korean or Malay). In our experiments on English to and from Hungarian, Dutch, and Portuguese translation, we find that we can obtain consistent improvements in both translation directions. To additionally verify our improvements we use human raters, who confirm the significance of the BLEU score improvements.

Finally, we compare training the preordering classifiers on small amounts of manually aligned data to training on large quantities of automatically aligned data for English to Arabic, Hebrew, and Japanese. When evaluated on a pure reordering task, the models trained on manually aligned data perform slightly better, but similar BLEU scores are obtained in both scenarios on an end-to-end translation task.

2 Classifier Reordering

Our goal is to learn a model that can transform the word order of an input sentence to an order that is natural in the target language. For example, when translating the English sentence:

The black cat climbed to the tree top.

to Spanish, we would like to reorder it as:

The cat black climbed to the top tree.

When translating to Japanese, we would like to get:

The black cat the tree top to climbed.

Such a model can then be used in combination with any translation model.

In our approach we first part-of-speech (POS) tag and parse the input sentence, producing the POS tags and head-modifier dependencies shown in Figure 1. Reordering is then done by traversing the dependency tree starting at the root. For each head word we determine the order of the head and its children (independently of other decisions) and continue the traversal recursively in that order. In the example, we first need to decide on the order of the head *"climbed"* and the children *"cat"*, *"to"*, and *"."*.

2.1 Classification Model & Features

The reordering decisions are made by multi-class classifiers where class labels correspond to permutation sequences. We train a separate classifier for each number of possible children. Crucially, we do not learn explicit tree transformations rules, but let the classifiers learn to trade off between a rich set of overlapping features.

Obviously, it is possible to use any classification model and learning algorithm. We use maximum entropy classifiers with l_1/l_∞ regularization trained with the GradBoost algorithm (Duchi and Singer, 2009). We chose this setup since it naturally supports multi-class prediction and can therefore be used to select one out of many possible permutations. Additionally, the learning algorithm produces a sparse set of features. In our experiments the final models have typically only a few 100K non-zero feature weights per language pair. Given this relatively small number of features, it is possible to manually inspect the feature weights and gain insights into the behavior of the model. We show an example analysis in Section 5.

Our features encode information about the context in which a word occurs in the sentence. We model context as "informative" words:

- The head itself.
- The children. We indicate whether each child is before, immediately before, immediately after, or after the head.
- For every child, if there is a gap between it and the head, then the first and last word of that gap.
- For every pair of consecutive children, if there is a gap between them, then the first and last word of that gap.
- The head's immediate sibling to the left/right or an indication that none exists.

When extracting the features, every word can be represented by its word identity, its fine-grained POS tag from the treebank, and a coarse-grained POS category, similar to the universal categories described in Petrov et al. (2012). We also include pairs of these features, resulting in potentially bilexical features.

2.2 Training Data

The training data for the classifiers is generated from the word aligned parallel text. Since parallel data is plentiful, we can afford to be selective. We first construct the intersection of high-confidence source-to-target and target-to-source alignments. For every family in the source dependency tree we generate a training instance if and only if the intersection defines a full order on the source words:

- Every source word must be aligned to at least one target word.

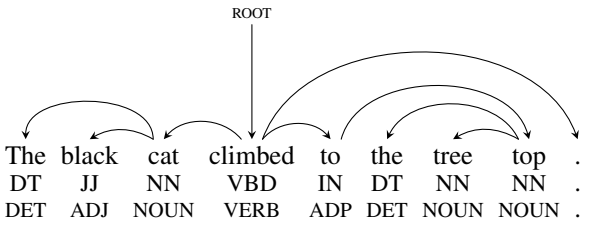


Figure 1: A sentence, its dependency parse and its fine-grained and coarse-grained POS tags.

- No two source words can be aligned to the same target word.
- If a source word is aligned to multiple target words, then no target word in this range can be aligned to a different source word.

While this might sound restrictive, we can usually generate at least some training instances from every sentence and discard the remaining families in the tree. In particular, we do not need to extract training instances for all words in a given sentence since the reordering decisions are made independently for every head word.

A potential concern might be that our method for selecting training data can exclude all instances of certain words. Consider the English phrase *"the boy"*. For languages without articles (e.g. Russian or Japanese) the determiner *"the"* may either not be aligned to any word or get aligned to the foreign word for *"boy"*. In both cases the family will be discarded according to either the first or the second condition above. The concern is therefore that we would have no training data with the English word *"the"*. In practice, however, this does not seem to be a problem. First, there are instances where the English word *"the"* gets aligned to something (perhaps a preposition), and second, since the word *"the"* is omitted in the target language its location in the reordered sentence is not very important.

Naturally we learn better classifier models from better alignments. The other direction is also true – if we run preordering on the source side then the alignment task becomes easier and tends to produce better results. Therefore it can be useful to iterate between generating the alignment and learning a preordering model. Empirically, the gains from this bootstrapping approach are not dramatic and are realized after just one iteration, i.e., create the align-

ment, train a preordering model, use the preordering model to learn a new alignment, and then train the final preordering model.

2.3 1-Step Classifier

As a first approach we use a single classifier to directly predict the correct permutation of a given family. Consider the family headed by *"climbed"* in Figure 1. There are three children and the original order of the words is *"cat"*, *"climbed"*, *"to"*, and *"."*. A possible outcome of the classifier can be the permutation 0-2-1-3, representing the order *"cat"*, *"to"*, *"climbed"*, and *"."*.

The number of permutations for the head and n children is of course $(n + 1)!$, which becomes large very quickly and causes some problems. In practice we therefore limit ourselves to the K most common permutations. Unfortunately, this means that whenever there are many children, the correct permutation order might not be available as an option. Even when the correct permutation is available, classification accuracy typically deteriorates as the number of possible classes increases.

An additional subtle issue is that the 1-step classifier cannot share useful information across different numbers of children. For example, in Spanish adjectives usually appear after the noun, but sometimes they appear before the noun. The decision depends on the adjective itself and sometimes the head noun, but does not depend on other children. Ideally, if for some adjective we have enough examples with 1 or 2 children we would like to make the same decision for a larger number of children, but these classifiers may not have enough relevant examples.

2.4 2-Step Classifier

Our 2-step approach addresses the exponential blowup of the number of children by decomposing the prediction into two steps:

1. For every child, decide whether it should appear before or after the head.
2. Determine the order of the children that appear before the head and the order of the children after the head.

The two steps make the reordering of the modifiers before and after the head independent of each other, which is reminiscent of the lexicalized parse tree generation approach of Collins (1997). In the running example, for the head *"climbed"* we might first make the following three binary decisions: the word *"cat"* should appear before the head and the words *"to"* and *"."* should appear after the head. In the second step there is only one word before the head so there is nothing to do. There are two words after the head, so we use another classifier to determine their order. The first step is implemented using a binary classifier, called the *pivot* classifier (since the head functions like the pivot in quicksort). The second step classifiers directly predict the correct permutation of the children before / after the head.

To illustrate the effectiveness of the 2-step approach, consider a head word with 4 children. The 1-step approach must predict 1 of 5! = 120 outcomes. In the 2-step approach, in the worst case the second step must predict 1 of 4! = 24 outcomes (if all the children are on one side of the head); if we are lucky and the children split evenly, then we only need two binary decisions in the second step (for the two pairs before and after the head). If we define hard cases as cases involving 5 or more words, 5.54% of the non-leaves are hard cases with the 1-step approach, but only 1.07% are hard cases with the 2-step approach.

3 Experimental Setup

To provide a through evaluation of our approach, we conduct experiments on two sets of data and with two translation systems. The first translation system is a phrase-based system (Och and Ney, 2004). In addition to the regular distance distortion model, we incorporate a maximum entropy based lexicalized phrase reordering model (Zens and Ney, 2006). Our second system is a forest-to-string system (Zhang et al., 2011). The forest-to-string system uses a one-best parse tree but factorizes it into a packed forest of binary elementary trees – hence the name forest-to-string rather than tree-to-string.

The systems are configured and tuned for each language pair to produce the best results. We then add our 1-step and 2-step preordering classifiers as preprocessing steps at training and test time. We train the reordering classifiers on up to 15M training instances. We train separate classifiers for every number of involved words, and restrict each one to the $K = 20$ most frequent outcomes.

In our implementation, in the 1-step approach we did not do any reordering for nodes with 7 or more children. In the 2-step approach we did not reorder the children on either side of the head if there were 7 or more of them. Even though there was no technical reason that prevented us from raising the thresholds, there was no good reason to do so. There were very few cases where children were not reordered because of these thresholds, many of them corresponded to bad parses, and they had very little impact on the final scores. Thus, for the 1-step approach we had 6 classifiers: 1 binary classifier for a head and a single child and 5 multi-class classifiers for 3–7 words. For the 2-step approach we had 11 classifiers: 1 pivot classifier, 5 classifiers for words before the head, and 5 for words after the head.

For a direct comparison to a strong preordering system, we compare to the system of Genzel (2010), which learns a set of unlexicalized reordering rules from automatically aligned data by minimizing the number of crossing alignments. We used a sliding window of size 3 and tried all three of their variants. There were about 40-50 rules per language pair. While conceptually possible, it is not practical to learn more rules (including lexicalized rules) with this system, because of the computational complexity of the learning algorithm and the incremental nature in which the rules are learned and applied.

3.1 WMT Setup

In our first set of experiments, we use the data provided for the WMT 2010 shared task (Callison-Burch et al., 2010). We build systems for all language pairs: English to and from Czech, French, German, and Spanish. Since this is a publicly available dataset, it is easy to compare our results to other submissions to the shared task.

During word alignment, we filter out sentences exceeding 60 words in the parallel texts and perform 6 iterations of IBM Model-1 training (Brown et al., 1993), followed by 6 iterations of HMM training (Vogel et al., 1996). We do not use Model-4 because it is slow and did not add much value to our systems in a pilot study. Standard phrase extraction heuristics (Koehn et al., 2003) are applied to extract phrase pairs with a length limit of 6 from alignments symmetrized with the "union" heuristic. Maximum jump width is set to 8. Rule extraction for the forest-to-string system is limited to 16 rules per tree node. There are no length-based reordering constraints in the forest-to-string system. We train two 5-gram language models with Kneser-Ney smoothing for each of the target languages. One is trained on the target side of the parallel text, the other on a news corpus provided by the shared task. We tune the feature weights for every configuration with 10 rounds of hypergraph-based Minimum Error Rate Training (MERT) (Kumar et al., 2009).

3.2 Additional Languages

In our second set of experiments, we explore the impact of classifier preordering for a number of languages with different word orders. Some of the languages included in our study are verb-subject-object (VSO) languages (Arabic, Irish, Welsh), subject-object-verb (SOV) languages (Japanese, Korean), and fairly free word order languages (Dutch, Hungarian). Where a parser is available, we also conduct experiments on translating into English.

Since there are no standard training sets for many of these language pairs, we use parallel data automatically mined from the web. The amount of parallel text for each language pair is between 120M and 160M words. For evaluation, we use a set of 9K English sentences collected from the web and translated by humans into each of the target languages. Each sentence has one reference translation. We use 5K sentences for evaluation and the rest for tuning.

The systems and training configurations are similar to the WMT setup. The word alignment step includes 3 iterations of IBM Model-1 training and 2 iterations of HMM training. Lexical reordering is included where it helps, but typically makes only a small difference. We again use a 5-gram language model trained on a large amount of monolingual text. Overall, we use between 20 and 30 features, whose weights are optimized using hypergraph-based MERT. All experiments for a given language pair use the same set of MERT weights. This potentially underestimates the improvements that can be obtained, but also eliminates MERT as a possible source of improvement, allowing us to trace back improvements in translation quality directly to changes in preordering of the input data.

3.3 Evaluation

We use case-sensitive BLEU (Papineni et al., 2002) to assess translation quality. For Japanese and Korean we use character-level BLEU. We use bootstrap resampling to compute confidence intervals.

Additionally, we also conduct a side-by-side human evaluation on 750 sentences for each language pair (sampled from the same sentences used for computing BLEU). For each sentence, we ask bilingual annotators to compare the translations from two different systems and say whether one is better, leading to three possible scores of -1, 0, and +1. We focus on this relative comparison since absolute scores are difficult to calibrate across languages and raters.

3.4 Syntactic Parsers

Table 1 shows our treebank sources and parsing accuracies. For English, we use the updated WSJ with OntoNotes-style annotations converted to Stanford dependencies (de Marneffe et al., 2006). The remaining treebanks are all available in dependency format. In all cases, we apply a set of heuristics to the treebank data to make the tokenization as similar as possible to the one of the bitext. Our heuristics can split treebank tokens but do not merge treebank tokens. We found that adjusting the treebank tokenization is crucial for obtaining good results. However, this makes the reported parsing accuracies not comparable to other numbers in the literature. When necessary, we projectivize the treebanks by raising arcs until the tree becomes projective, as described in Nivre and Nilsson (2005); we do not reconstruct non-projective arcs at parsing time, since our subsequent systems expect projective trees.

Our part-of-speech tagger is a conditional random field model (Lafferty et al., 2001) with simple word-identity and affix features. The parsing model is a shift-reduce dependency parser, using the higher-order features from Zhang and Nivre (2011). Additionally, we include 256 word-cluster features (Koo et al., 2008) trained on a large amount of unlabeled monolingual text (Uszkoreit and Brants, 2008).

4 Experiments

Due to the large number of experiments and language pairs we divide the experiments into groups and discuss each in turn. We only include the results from the forest-to-string system when they are better than the phrase-based results. We use * to denote results from the forest-to-string system.

	UAS	LAS	POS
en: English[1]	92.28	90.28	97.05
cs: Czech[2]	84.66	72.01	98.97
de: German[3]	89.30	86.98	97.69
es: Spanish[4]	86.24	82.32	96.62
fr: French[5]	88.57	86.40	97.48
hu: Hungarian[2]	87.66	82.51	94.47
nl: Dutch[3]	86.09	82.31	97.38
pt: Portuguese[4]	90.22	87.26	98.10

Table 1: Parsing accuracies on the retokenized treebanks. UAS is unlabeled attachment score, LAS is labeled attachment score, and POS is part-of-speech tagging accuracy. The treebank sources are (1): Marcus et al. (1993) + Judge et al. (2006) + Petrov and McDonald (2012), (2): Nivre et al. (2007), (3): Buchholz and Marsi (2006), (4): McDonald et al. (2013), (5): Abeillé et al. (2003).

4.1 WMT Experiments

Table 2 presents detailed results on the WMT setup. Lexical reordering (Zens and Ney, 2006) never hurts and is thus included in all systems. Overall, our results are a little better than the best results of the WMT 2010 shared task for two language pairs and within reach of the best results in most other cases.

The 2-step classifier preordering approach provides statistically significant improvements over the lexical reordering baseline on three out of the eight language pairs: English-Spanish (en-es: 1.4 BLEU), German-English (de-en: 1.2 BLEU), and English-French (en-fr: 1.0 BLEU). These improvements are significant in our human side-by-side evaluation. We also observe gains when combining our preordering approach with the forest-to-string system for English-Spanish and German-English. While the forest-to-string system is capable of performing long distance reordering in the decoder, it appears that an explicitly trained lexicalized preordering model can provide complementary benefits. These benefits are especially pronounced for German-English where long distance verb movement is essential. For the romance languages (Spanish and French), word ordering depends highly on lexical choice which is captured by the lexical features in our classifiers.

	base	lexical	rule	1-step	2-step	wmt best
en-cs	14.9	**15.1**	**15.2**	**15.2**	**15.2**	15.4
en-de	15.3	15.6	**15.9**	**15.9**	**15.7**	16.3
en-es	27.4	27.8♢	28.4♧	**29.0**	**28.8♠♣**	28.6
en-es*	28.9	-	28.7	29.0	29.2	28.6
en-fr	26.3	26.5♢	26.8♧	**27.2**	**27.3♠♧**	27.6
cs-en	21.6	21.6	21.5	21.6	21.7	21.9
de-en	20.6	21.1♢	**21.9**	**21.9**	**21.8♠**	22.8
de-en*	22.1	-	**22.5**	**22.5**	**22.7**	22.8
es-en	28.3	**28.7**	**28.7**	**28.8**	**28.9**	28.8
fr-en	26.8	27.0	26.9	26.9	27.0	28.3

Table 2: BLEU scores on the WMT 2010 setup. Results from the forest-to-string system are marked with * and are only included when better than the phrase-based results. The *base* system includes a distance distortion model; the *lexical* system adds lexical reordering; *rule* is the rule preordering system of Genzel (2010) plus lexical reordering; *1-step* and *2-step* are our classifier-based systems plus lexical reordering. Bolded results are statistically significantly better than non-bolded results as measured by a bootstrap sample test with a 99% confidence interval. Human evals are conducted only where indicated; we use ♠ and ♣ to indicate a significantly better result than ♢ and ♧ in the human eval at 95%. Also included are the best results from the WMT 2010 task.

Compared to a state-of-the-art preordering system, the automatic rule extraction system of Genzel (2010), we observe significant gains in several cases and no losses at all. The improvements on English-Spanish are significant also in the human evaluation, while the English-French improvements are positive, but not statistically significant.

Comparing the different languages, Czech (cs) appears the most immune to improvements from preordering (and lexical reordering). One possible explanation is that Czech has a relatively free word order with a default SVO structure. It is therefore difficult to learn reordering changes from English to Czech. Additionally, the accuracy (LAS) of our Czech parser is by far the lowest of all parsers that we used, potentially limiting the benefits that can be obtained when translating from Czech into English.

On this setup there is fairly little difference in performance between the 1-step and 2-step approaches. The main benefit of the 2-step approach is compactness: the set of 2-step classifiers has about half the number of non-zero features as the 1-step classifiers.

4.2 Additional Languages Experiments

Table 3 shows our first set of results on the additional languages, including some languages with a wide disparity in word order relative to English. The SOV languages Korean (ko) and Japanese (ja) benefit the most from preordering and gain more than 7 BLEU relative to the phrase-based baseline and still more than 3 BLEU for the forest-to-string system. Similar improvements were reported by Xu et al. (2009) with manual reordering rules. Indonesian (id) and Malay (ms) are next with gains of 2.5 BLEU. Malay does not have a grammatical subject in the sense that English does, but instead uses a concept of an agent and an object, whose order is determined by the voice of the verb. It appears that our classifiers have learned to model some of these highly lexical, but systematic ordering preferences. Welsh (cy) and Irish (ga) as VSO languages also exhibit large gains of 2.1 BLEU. For Arabic (ar) and Hebrew (iw), the gains are smaller, but still significant and exceed 1 BLEU relative to the baseline.

The benefits of our 2-step approach over the 1-step approach become apparent on this set of languages where reordering is most important. By predicting the target word order in two steps, we reduce sparsity and make two easier decisions in place of a single difficult high entropy decision. Indeed, the 2-step approach produces improvements over the 1-step approach on five out of nine language pairs. The improvements are as large as 0.9 BLEU for Korean and 0.5 BLEU for Japanese and Welsh. We performed human evaluation for all language pairs with a noticeable BLEU gain for the 2-step system over

	base	rule	1-step	2-step
en-ar	11.4	12.3	**12.5**	**12.6**
en-cy	29.3	31.1	31.9♧	**32.4**♣
en-ga	17.0	18.5	18.8♧	**19.1**♣
en-iw	18.8	19.7	**20.2**	**20.2**
en-id	31.0	33.4	**34.0**♧	**34.3**♧
en-ja	10.4	16.4	17.5♧	**18.0**♣
en-ja*	14.9	18.0	18.2♧	**18.6**♣
en-ko	24.1	31.8	31.8♧	**32.7**♣
en-ms	20.4	22.5	**22.9**	**22.9**

Table 3: BLEU scores for language from various language families: Arabic (ar), Welsh (cy), Irish (ga), Indonesian (id), Hebrew (iw), Japanese (ja), Korean (ko), and Malay (ms). Lexical reordering is not included in any of the systems. Bolded results are significant at 99%. ♣ is significantly better than ♧ in a human eval at 95%.

the 1-step system. The human judgments exactly agree with the results of the BLEU significance tests. The gains relative to the rule reordering system of Genzel (2010) and the no-preordering baseline are even larger and therefore clearly also significant.

In Table 4 we show results for Hungarian (hu), Dutch (nl), and Portuguese (pt). In all cases but English-Hungarian we observe significant improvements over the no preordering baseline. It should be noted that the gains are not symmetric – sometimes there are larger gains for translating out of English, while for Hungarian the gains are higher for translating into English. Hungarian has a free word order which is difficult to predict which might partially explain why there are no improvements for translating into Hungarian. For Dutch-English, the forest-to-string system yields the best results, which was also the case for German-English, further supporting the observation that combining different types of syntactic reordering approaches can be beneficial.

4.3 Manually Aligned Data

For Arabic (ar), Hebrew (iw), and Japanese (ja) we conducted some additional experiments with manually aligned data. We asked bilingual speakers to translate about 20K English sentences into the respective target language and to mark the alignment between the words. We reserved 20% of this data for evaluation and used the rest for training. For evaluation we used the fuzzy metric defined by Talbot et al. (2011), which counts the fraction of words that are reordered into the correct position.

	base	rule	1-step	2-step
en-hu	12.7	12.6	12.8	12.7
en-nl	25.3	26.1	**26.4**	**26.4**
en-pt	30.2	31.9	**32.6**	**32.8**
hu-en	22.0	22.2	**22.7**	**22.7**
nl-en	34.9	**35.7**	35.2	35.1
nl-en*	36.3	**36.5**	**36.6**	**36.7**
pt-en	39.8	**40.1**	**40.1**	**40.1**

Table 4: BLEU scores for translating to and from English for: Hungarian (hu), Dutch (nl), and Portuguese (pt). Lexical reordering is not used for any language pair. Bolded results are significant at 99%.

The BLEU scores in Table 5 show that training from small amounts of manually aligned data or large amounts of automatically aligned data results in models of similar quality. In terms of the fuzzy metric, the models trained from manually aligned data were better. A possible explanation is that these models were trained on data which was much more similar to the evaluation data (both were subsets of the manually aligned data), biasing the metric in their favor. In absolute terms, the reordering accuracy is around 80% for Arabic and Japanese and close to 90% for Hebrew. Most impressively, more than 60% of the Hebrew sentences are exactly in the correct word order, implying that monotonic translation may suffice.

We also examined the accuracy of the individual classifiers and found that the pivot classifier has an accuracy around 95%. It is therefore unlikely that a word is reordered to the wrong side of its head in the 2-step reordering approach. The classifiers that predict the final word order have an accuracy above 90% when there are only two words and drop to still respectable 70%-80% when there are 4 or more children or 20 possible options.

5 Analysis

In this section, we analyze an example whose translation is significantly improved by our preordering approach, demonstrating the usefulness of our lexicalized features. Consider the English sentence:

It was a real whirlwind.

	no reordering			manual			automatic		
	fuzzy	exact	BLEU	fuzzy	exact	BLEU	fuzzy	exact	BLEU
en-ar	63.2	19.8	11.4	83.5	47.6	12.4	79.0	38.9	12.6
en-iw	67.9	22.2	18.8	89.8	62.4	20.3	89.2	61.2	20.2
en-ja*	44.1	0.0	14.9	80.9	41.5	18.4	78.5	36.8	18.6

Table 5: Preordering accuracy for the 2-step classifiers using manual alignments vs. automatic alignments. Fuzzy refers to the metric defined in Talbot et al. (2011) and exact is the percentage of sentences with a perfect preordering.

taken from the WMT test set. The dependency parse tree is shown in Figure 2. In our experiments the rule-based approach of (Genzel, 2010) reordered the source sentence into:

It was a whirlwind real.

and produced the translation:

Es un torbellino real.

In comparison, our 2-step system kept the English sentence unchanged and produced the translation:

Fue un auténtico torbellino.

The second translation is better than the first because of the correct tense (which is not related directly to the preordering) and because the noun phrase *"real whirlwind"* is ordered correctly.

The main reason for the difference in the ordering is that the rule-based system can only use the unlexicalized information from the parse tree. The head *"whirlwind"* is a noun and the child *"real"* is an adjective; since adjectives typically appear after nouns in Spanish, their order is reversed.

To understand why the classifier-based system keeps *"real"* before *"whirlwind"* we can examine the features used by the classifier to make this decision. In Table 6 we consider the 3 strongest features in favor of the child *"real"* appearing after the head *"whirlwind"* and the three strongest features in favor of the child appearing before the head. Recall that the pivot is a binary classifier: positive features support one decision (in our case: the child should be after the head) and the negative features support the other decision (the child should be before the head).

The three features that have the highest positive weight encode the fact that the child is an adjective, since in general, adjectives in Spanish appear after the noun. On the other hand, the three features with the most negative weights all encode the fact that the child is the word *"real"* which unlike most adjectives tends to appear before the noun. It is interesting to note that for this particular ordering decision the child word is much more informative than the head word and indeed, all the important features contain information about the child and none of them contains any information about the head.

6 Conclusions & Future Work

We presented a simple and novel preordering approach that produces substantial improvements in translation accuracy on a large number of languages. We use a source-side syntactic parser and train discriminative classifiers to predict the order of a parent and its children in the target language, using features from the dependency tree as well as (bi-)lexical features. To decompose the exponential space of all possible permutations, we introduce the 2-step approach. We show empirically that this approach is significantly better than directly predicting the full permutation for some languages, and never significantly worse.

We obtain strong results on the WMT 2010 shared task data, observing gains of up to 1.4 BLEU over a state-of-the-art system. We also show gains of up to 0.5 BLEU over a strong directly comparable preordering system that is based on learning unlexicalized reordering rules. We obtain improvements of more than 2 BLEU in experiments on additional languages. The gains are especially large for languages where the sentence structure is very different from English. These positive results are confirmed in human side-by-side evaluations.

When comparing our approach to syntax-based translation systems (Yamada and Knight, 2001; Galley et al., 2004; Huang et al., 2006; Dyer and Resnik, 2010) we note that both approaches use syntactic information for reordering decisions. Our preordering approach has several advantages. First, be-

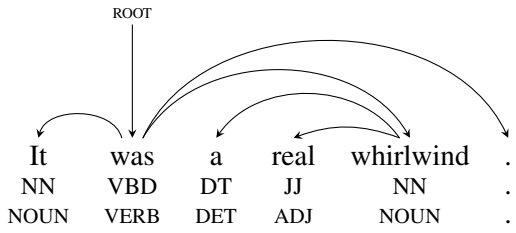


Figure 2: An example where lexical information is necessary for choosing the correct word order.

cause preordering is performed before learning word alignments, it has the potential to improve the word alignments. Second, by using discriminative classifiers we can take advantage of lexical features. Finally, preordering can be combined with syntax-based translation models and our results confirm the complementary benefits that can be obtained.

Compared to other preordering models, our approach has the obvious problem of having to make predictions over an exponential set of permutations. We show that this is not an insurmountable difficulty: our 2-step approach decomposes the exponential space, often leading to much easier prediction tasks. Even when the number of possible permutations is large we can limit ourselves to the K most popular permutations.

On the other hand, our approach provides important advantages. Compared to systems that use rewrite rules, it is much easier to encode useful knowledge that by itself is not enough to determine a full rewrite rule, such as "a determiner is unlikely to be the last word in a clause." Perhaps more importantly, our model provides an elegant answer to the question of what to do when multiple rewrite rules can be applied. Previous work has employed different heuristics: use the most specific rule (Xia and McCord, 2004), use all applicable rules (Genzel, 2010), or use the most frequent rule (Wu et al., 2011). In our model there is no need for such heuristics – all the "rules" are treated as features to a discriminative classifier, and the task of analyzing their interactions is handled by the learning algorithm.

Compared to preordering systems that use ranking functions, our model has the advantage that it can encode information about the complete permutation. For example, for three source words A, B, and C, we can naturally express the useful prior that A-B-C and C-B-A are likely orders but C-A-B is not.

Feature	Weight
`PrevChild:tag=JJ,PrevSibling:a`	`0.448`
`PrevChild:cat=ADJ,PrevSibling:a`	`0.292`
`PrevChild:cat=ADJ,NoNextSibling`	`0.212`
`...`	
`PrevChild:real,NoNextHeadSibling`	`-0.310`
`PrevChild:real,PrevSibling:cat=DET`	`-0.516`
`PrevChild:real,PrevSibling:a`	`-0.979`

Table 6: The three features with the highest and lowest weights for choosing the position of *"real"* relative to *"whirlwind."* *PrevChild* means that the child is the immediate word before the head. *PrevSibling* refers to the child's sibling immediately to the left (the determiner *"a"*). *NoNextSibling* and *NoNextHeadSibling* mean that the child and head do not have a sibling to the right.

Promising directions for future work are joint parsing and reordering models, and measuring the influence of parsing accuracy on preordering and final translation quality.

References

A. Abeillé, L. Clément, and F. Toussenel. 2003. Building a Treebank for French. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 10. Kluwer.

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2).

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL '06*.

C. Callison-Burch, P. Koehn, C. Monz, K. Peterson, M. Przybocki, and O. Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proc. of ACL'05 WMT*.

M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL '05*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL '97*.

M.-C. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC '06*.

J. DeNero and J. Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proc. of EMNLP '11*.

J. Duchi and Y. Singer. 2009. Boosting with structural sparsity. In *Proc. of ICML '09*.

C. Dyer and P. Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. of NAACL-HLT '10*.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *Proc. of NAACL-HLT '04*.

D. Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proc. of COLING '10*.

N. Habash. 2007. Syntactic preprocessing for statistical machine translation. In *Proc. of MTS '07*.

L. Huang, K. Knight, and A. Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA '06*.

J. Judge, A. Cahill, and J. v. Genabith. 2006. QuestionBank: creating a corpus of parse-annotated questions. In *Proc. of ACL '06*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase based translation. In *Proc. of NAACL-HLT '03*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL-HLT '08*.

S. Kumar, W. Macherey, C. Dyer, and F. Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of ACL '09*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML '01*.

C. H. Li, M. Li, D. Zhang, M. Li, M. Zhou, and Y. Guan. 2007. A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation. In *Proc. of ACL '07*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

R. McDonald, J. Nivre, Y. Quirmbach-Brundagez, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Täckström, C. Bedini, N. Bertomeu Castelló, and J. Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL '13*.

G. Neubig, T. Watanabe, and S. Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proc. of EMNLP-CoNLL '12*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL '05*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. EMNLP-CoNLL '07*.

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

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL '02*.

S. Petrov and R. McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Proc. of NAACL '12 SANCL*.

S. Petrov, D. Das, and R. McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC '12*.

D. Talbot, H. Kazawa, H. Ichikawa, J. Katz-Brown, M. Seno, and F. Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proc. of EMNLP '11 WMT*.

C. Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proc. of NAACL-HLT '04*.

R. Tromble and J. Eisner. 2009. Learning linear ordering problems for better translation. In *Proc. of EMNLP '09*.

J. Uszkoreit and T. Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proc. of ACL-HLT '08*.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *In Proc. of COLING '96*.

C. Wang, M. Collins, and P. Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proc. of EMNLP-CoNLL '07*.

X. Wu, K. Sudoh, K. Duh, H. Tsukada, and M. Nagata. 2011. Extracting pre-ordering rules from predicate-argument structures. In *Proc. of IJCNLP '11*.

F. Xia and M. McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proc. of COLING '04*.

P. Xu, J. Kang, M. Ringgaard, and F. Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proc. of NAACL-HLT '09*.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL '01*.

N. Yang, M. Li, D. Zhang, and N. Yu. 2012. A ranking-based approach to word reordering for statistical machine translation. In *Proc. of ACL '12*.

R. Zens and H. Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proc. of NAACL '06 WMT*.

Y. Zhang and J. Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proc. of ACL-HLT '11*.

H. Zhang, L. Fang, P. Xu, and X. Wu. 2011. Binarized forest to string translation. In *Proc. of ACL-HLT '11*.

Improving Pivot-Based Statistical Machine Translation Using Random Walk

Xiaoning Zhu[1*], Zhongjun He[2], Hua Wu[2], Haifeng Wang[2], Conghui Zhu[1], and Tiejun Zhao[1]
Harbin Institute of Technology, Harbin, China[1]
Baidu Inc., Beijing, China[2]
{xnzhu, chzhu, tjzhao}@mtlab.hit.edu.cn
{hezhongjun,wu_hua,wanghaifeng}@baidu.com

Abstract

This paper proposes a novel approach that utilizes a machine learning method to improve pivot-based statistical machine translation (SMT). For language pairs with few bilingual data, a possible solution in pivot-based SMT using another language as a "bridge" to generate source-target translation. However, one of the weaknesses is that some useful source-target translations cannot be generated if the corresponding source phrase and target phrase connect to different pivot phrases. To alleviate the problem, we utilize Markov random walks to connect possible translation phrases between source and target language. Experimental results on European Parliament data, spoken language data and web data show that our method leads to significant improvements on all the tasks over the baseline system.

1 Introduction

Statistical machine translation (SMT) uses bilingual corpora to build translation models. The amount and the quality of the bilingual data strongly affect the performance of SMT systems. For resource-rich language pairs, such as Chinese-English, it is easy to collect large amounts of bilingual corpus. However, for resource-poor language pairs, such as Chinese-Spanish, it is difficult to build a high-performance SMT system with the small scale bilingual data available.

The pivot language approach, which performs translation through a third language, provides a possible solution to the problem. The triangulation method (Wu and Wang, 2007; Cohn and Lapata, 2007) is a representative work for pivot-based machine translation. With a triangulation pivot approach, a source-target phrase table can be obtained by combining the source-pivot phrase table and the pivot-target phrase table. However, one of the weaknesses is that some corresponding source and target phrase pairs cannot be generated, because they are connected to different pivot phrases (Cui et al., 2013). As illustrated in Figure 1, since there is no direct translation between "很可口 henkekou" and "really delicious", the triangulation method is unable to establish a relation between "很可口 henkekou" and the two Spanish phrases.

To solve this problem, we apply a Markov random walk method to pivot-based SMT system. Random walk has been widely used. For example, Brin and Page (1998) used random walk to discover potential relations between queries and documents for link analysis in information retrieval. Analogous to link analysis, the aim of pivot-based translation is to discover potential translations between source and target language via the pivot language.

[*] This work was done when the first author was visiting Baidu.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 524–534,
Seattle, Washington, USA, 18-21 October 2013.

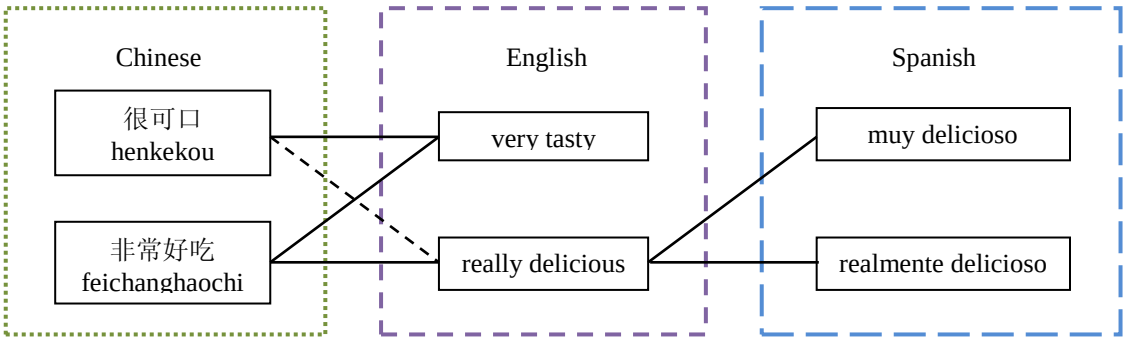


Figure 1: An example of random walk on phrase table. The dashed line indicates an implicit relation in the phrase table.

The goal of this paper is to extend the previous triangulation approach by exploring implicit translation relations using random walk method. We evaluated our approach in several translation tasks, including translations between European languages; Chinese-Spanish spoken language translation and Chinese-Japanese translation with English as the pivot language. Experimental results show that our approach achieves significant improvements over the conventional pivot-based method, triangulation method.

The remainder of this paper is organized as follows. In section 2, we describe the related work. We review the triangulation method for pivot-based machine translation in section 3. Section 4 describes the random walk models. In section 5 and section 6, we describe the experiments and analyze the performance, respectively. Section 7 gives a conclusion of the paper.

2 Related Work

Several methods have been proposed for pivot-based translation. Typically, they can be classified into 3 kinds of methods:

Transfer Method: Within the transfer framework (Utiyama and Isahara, 2007; Wang et al., 2008; Costa-jussà et al., 2011), a source sentence is first translated to *n* pivot sentences via a source-pivot translation system, and then each pivot sentence is translated to *m* target sentences via a pivot-target translation system. At each step (source to pivot and pivot to target), multiple translation outputs will be generated, thus a minimum Bayes-risk system combination method is often used to select the optimal sentence (González-Rubio et al., 2011; Duh et al., 2011). A problem with the transfer method is that it needs to decode twice. On one hand, the time cost is doubled; on the other hand, the translation error of the source-pivot translation system will be transferred to the pivot-target translation.

Synthetic Method: A synthetic method creates a synthetic source-target corpus using source-pivot translation model or pivot-target translation model (Utiyama et al., 2008; Wu and Wang, 2009). For example, we can translate each pivot sentence in the pivot-target corpus to source language with a pivot-source model, and then combine the translated source sentence with the target sentence to obtain a synthetic source-target corpus, and vice versa. However, it is difficult to build a high quality translation system with a corpus created by a machine translation system.

Triangulation Method: The triangulation method obtains source-target model by combining source-pivot and pivot-target translation models (Wu and Wang, 2007; Cohn and Lapata 2007), which has been shown to work better than the other pivot approaches (Utiyama and Isahara, 2007). As we mentioned earlier, the weakness of triangulation is that the corresponding source and target phrase pairs cannot be connected in the case that they connect to different pivot phrases.

3 The Triangulation Method

In this section, we review the triangulation method for pivot-based translation.

With the two additional bilingual corpora, the source-pivot and pivot-target translation models can be trained. Thus, a pivot model can be obtained by merging these two models. In the translation model, the phrase translation probability and the lexical weight are language dependent, which will be introduced in the next two sub-sections.

3.1 Phrase Translation Probability

The triangulation method assumes that there exist translations between phrases $\bar{s}$ and phrase $\bar{p}$ in source and pivot languages, and between phrase $\bar{p}$ and phrase $\bar{t}$ in pivot and target languages. The phrase translation probability ϕ between source and target languages is determined by the following model:

$$\begin{aligned}\phi(\bar{s} \mid \bar{t}) &= \sum_{\bar{p}} \phi(\bar{s} \mid \bar{p}, \bar{t}) \phi(\bar{p} \mid \bar{t}) \\ &= \sum_{\bar{p}} \phi(\bar{s} \mid \bar{p}) \phi(\bar{p} \mid \bar{t})\end{aligned} \quad (1)$$

3.2 Lexical Weight

Given a phrase pair $(\bar{s}, \bar{t})$ and a word alignment a between the source word positions $i = 1, \ldots, n$ and the target word positions $j = 0, 1, \ldots, m$, the lexical weight of phrase pair $(\bar{s}, \bar{t})$ can be calculated with the following formula (Koehn et al. 2003) :

$$p_{\xi}(\bar{s} \mid \bar{t}, a) = \prod_{i=1}^{n} \frac{1}{\left|\{ j \mid (i, j) \in a \}\right|} \sum_{\forall (i,j) \in a} \omega(s_i \mid t_j) \quad (2)$$

In formula 2, the lexical translation probability distribution $\omega(s \mid t)$ between source word s and target word t can be estimated with formula 3.

$$\omega(s \mid t) = \frac{count(s, t)}{\sum_{s'} count(s', t)} \quad (3)$$

Thus the alignment a between the source phrase $\bar{s}$ and target phrase $\bar{t}$ via pivot phrase $\bar{p}$ is needed for computing the lexical weight. The alignment a can be obtained as follows:

$$a = \{(s, t) \mid \exists p : (s, p) \in a_1 \,\&\, (p, t) \in a_2\} \quad (4)$$

where a_1 and a_2 indicate the word alignment between the phrase pair $(\bar{s}, \bar{p})$ and $(\bar{p}, \bar{t})$, respectively.

The triangulation method requires that both the source and target phrases connect to the same pivot phrase. Otherwise, the source-target phrase pair cannot be discovered. As a result, some useful translation relations will be lost. In order to alleviate this problem, we propose a random walk model, to discover the implicit relations among the source, pivot and target phrases.

4 Random Walks on Translation Graph

For phrase-based SMT, all source-target phrase pairs are stored in a phrase table. In our random walk approach, we first build a translation graph according to the phrase table. A translation graph contains two types of nodes: source phrase and target phrase. A source phrase $\bar{s}$ and a target phrase $\bar{t}$ are connected if exists a phrase pair $(\bar{s}, \bar{t})$ in the phrase table. The edge can be weighted according to translation probabilities or alignments in the phrase table. For the pivot-based translation, the translation graph can be derived from the source-pivot phrase table and pivot-target phrase table.

Our random walk model is inspired by two works (Szummer and Jaakkola, 2002; Craswell and Szummer,2007). The general process of random walk can be described as follows:

Let $G = (V, E)$ be a directed graph with n vertices and m edges. For a vertex $v \in V$, $\Gamma(v)$ denotes the set of neighbors of v in G. A random walk on G follows the following process: start at a vertex v_0, chose and walk along a random neighbor v_1, with $v_1 \in \Gamma(v_0)$. At the second step, start from v_1 and chose a random neighbor v_2, and so on.

Let *S* be the set of source phrases, and *P* be the set of pivot phrases. Then the nodes *V* are the union of *S* and *P*. The edges *E* correspond to the relations between phrase pairs.

Let *R* represent the binary relations between source phrases and pivot phrases. Then the 1-step translation R_{ik} from node *i* to node *k* can be directly obtained in the phrase table.

Define operator $\otimes$ to denote the calculation of relation *R*. Then 2-step translation R_{ij} from node *i* to node *j* can be obtained with the following formula.

$$R_{ij} = R_{ik} \otimes R_{kj} \quad (4)$$

We use $R_{t|0}(k \mid i)$ to denote a *t*-step translation relation from node *i* to node *k*. In order to calculate the translation relations efficiently, we use a matrix *A* to represent the graph. A *t* step translation probability can be denoted with the following formula.

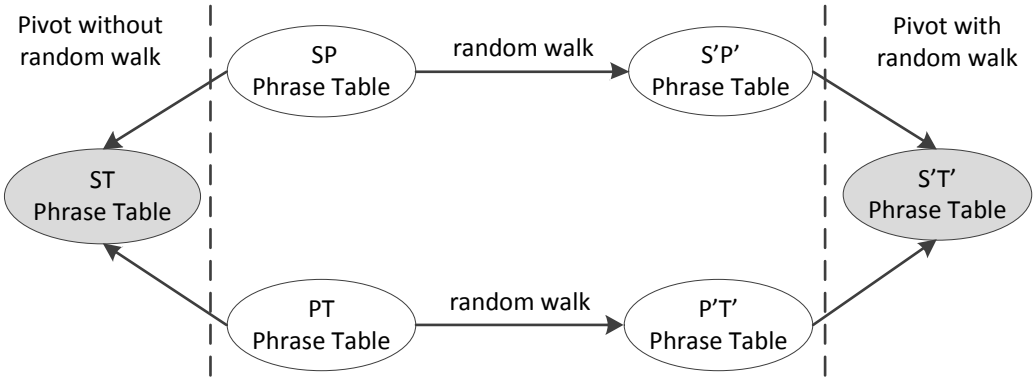
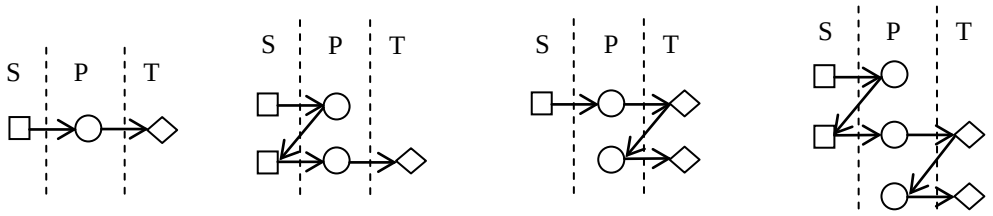


Figure 2: Framework of random walk based pivot translation. The ST phrase table was generated by combining SP and PT phrase table through triangulation method. The phrase table with superscript "'" means that it was enlarged by random walk.



(a) Pivot without random walk (b) Random walk on source-pivot side (c) Random walk on pivot-target side (d) Random walk on both sides

Figure 3: Some possible decoding processes of random walk based pivot approach. The □ stands for the source phrase (S); the ○ represents the pivot phrase (P) and the ◇ stands for the target phrase (T).

$$P_{t|0}(k \mid i) = [A^t]_{ik} \quad (5)$$

where $\boldsymbol{A}$ is a matrix whose i,k-th element is R_{ik} .

4.1 Framework of Random Walk Approach

The overall framework of random walk for pivot-based machine translation is shown in Figure 2. Before using random walk model, we have two phrase tables: source-pivot phrase table (SP phrase table) and pivot-target phrase table (PT phrase table). After applying the random walk approach, we can achieve two extended phrase table: extended source-pivot phrase table (S'P' phrase table) and extended pivot-target phrase table (P'T' phrase table). The goal of pivot-based SMT is to get a source-target phrase table (ST phrase table) via SP phrase table and PT phrase table.

Our random walk was applied on SP phrase table or PT phrase table separately. In next 2 subsections, we will explain how the phrase translation probabilities and lexical weight are obtained with random walk model on the phrase table.

Figure 3 shows some possible decoding processes of random walk based pivot approach. In figure 3-a, the possible source-target phrase pair can be obtained directly via a pivot phrase, so it does not need a random walk model. In figure 3-b and figure 3-c, one candidate source-target phrase pair can be obtained by random walks on source-pivot side or pivot-target side. Figure 3-d shows that the possible source-target can only by obtained by random walks on source-pivot side and pivot-target side.

4.2 Phrase Translation Probabilities

For the translation probabilities, the binary relation R is the translation probabilities in the phrase table. The operator $\otimes$ is multiplication. According to formula 5, the random walk sums up the probabilities of all paths of length t between the node i and k.

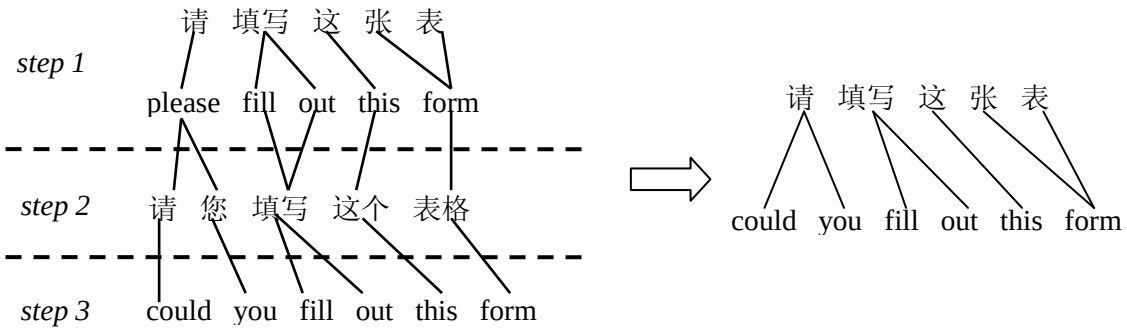


Figure 4: An example of word alignment induction with 3 steps random walks

Take source-to-pivot phrase graph as an example; denote matrix A contains $s+p$ nodes (s source phrases and p pivot phrases) to represent the translation graph.

$$A = \left[g_{ij} \right]_{(s+p)\times(s+p)} \quad (6)$$

where g_{ij} is the i,j-th elements of matrix A.

We can split the matrix A into 4 sub-matrixes:

$$A = \begin{bmatrix} 0_{s\times s} & A_{sp} \\ A_{ps} & 0_{p\times p} \end{bmatrix} \quad (7)$$

where the sub-matrix $A_{sp} = [p_{ik}]_{s\times p}$ represents the translation probabilities from source to pivot language, and A_{ps} represents the similar meaning.

Take 3 steps walks as an example:

Step1:

$$A = \begin{bmatrix} 0_{s\times s} & A_{sp} \\ A_{ps} & 0_{p\times p} \end{bmatrix}$$

Step2:

$$A^2 = \begin{bmatrix} A_{sp}\times A_{ps} & 0_{s\times p} \\ 0_{p\times s} & A_{ps}\times A_{sp} \end{bmatrix}$$

Step3:

$$A^3 = \begin{bmatrix} 0_{s\times s} & A_{sp}\times A_{ps}\times A_{sp} \\ A_{ps}\times A_{sp}\times A_{ps} & 0_{p\times p} \end{bmatrix}$$

For the 3 steps example, each step performs a translation process in the form of matrix's self-multiplication.

1. The first step means the translation from source language to pivot language. The matrix A is derived from the phrase table directly and each element in the graph indicates a translation rule in the phrase table.
2. The second step demonstrates a procedure: S-P-S'. With 2 steps random walks, we can find the synonymous phrases, and this procedure is analogous to paraphrasing (Bannard and Callison-Burch, 2005). For the example shown in figure 1 as an example, the hidden relation between "*很可口 henkekou*" and "*非常好吃 feichanghaochi*" can be found through Step 2.
3. The third step describes the following procedure: S-P-S'-P'. An extended source-pivot phrase table is generated by 3-step random walks. Compared with the initial phrase table in Step1, although the number of phrases is not increased, the relations between phrase pairs are increased and more translation rules can be obtained. Still for the example in Figure 1 , the hidden relation between "*很可口 henkekou*" and "*really delicious*" can be generated in Step 3.

4.3 Lexical Weights

To build a translation graph, the two sets of phrase translation probabilities are represented in the phrase tables. However, the two lexical weights are not presented in the graph directly. To deal with this, we should conduct a word alignment random walk model to obtain a new alignment a after t steps. For the computation of lexical weights, the relation R can be expressed as the word alignment in the phrase table. The operator $\otimes$ can be induced with the following formula.

$$a = \{(x,y) \mid \exists p : (x,z) \in a_1 \,\&\, (z,y) \in a_2\} \quad (8)$$

where a_1 and a_2 represent the word alignment information inside the phrase pairs $(\bar{x},\bar{y})$ and $(\bar{y},\bar{z})$ respectively. An example of word alignment inducing is shown in Figure 4. With a new word alignment, the two lexical weights can be calculated by formula 2 and formula 3.

5 Experiments

5.1 Translation System and Evaluation Metric

In our experiments, the word alignment was obtained by GIZA++ (Och and Ney, 2000) and the heuristics "grow-diag-final" refinement rule. (Koehn et al., 2003). Our translation system is an in-house phrase-based system using a log-linear framework including a phrase translation model, a language model, a lexicalized reordering model, a word penalty model and a phrase penalty model, which is analogous to Moses (Koehn et al., 2007). The baseline system is the triangulation method based pivot approach (Wu and Wang, 2007).

To evaluate the translation quality, we used BLEU (Papineni et al., 2002) as our evaluation metric. The statistical significance using 95% confidence intervals were measured with paired bootstrap resampling (Koehn, 2004).

5.2 Experiments on Europarl

5.2.1. Data sets

We mainly test our approach on Europarl[1] corpus, which is a multi-lingual corpus including 21 European languages. Due to the size of the data, we only select 11 languages which were added to Europarl from 04/1996 or 01/1997, including Danish (da), German (de), Greek (el), English (en), Spanish (es), Finnish (fi), French (fr), Italian (it) Dutch (nl) Portuguese (pt) and Swedish (sv). In order to avoid a trilingual scenario, we split the training corpus into 2 parts by the year of the data: the data released in odd years were used for training source-pivot model and the data released in even years were used for training pivot-target model.

We perform our experiments on different translation directions and via different pivot languages. As a most widely used language in the world (Mydans, 2011), English was used as the pivot language for granted when carrying out experiments on different translation directions. For translating Portuguese to Swedish, we also tried to perform our experiments via different pivot languages. Table 1 and Table 2 summarized the training data.

Language Pairs (src-pvt)	Sentence Pairs #	Language Pairs (pvt-tgt)	Sentence Pairs #
da-en	974,189	en-da	953,002
de-en	983,411	en-de	905,167
el-en	609,315	en-el	596,331
es-en	968,527	en-es	961,782
fi-en	998,429	en-fi	903,689
fr-en	989,652	en-fr	974,637
it-en	934,448	en-it	938,573
nl-en	982,696	en-nl	971,379
pt-en	967,816	en-pt	960,214
sv-en	960,631	en-sv	869,254

Table1. Training data for experiments using English as the pivot language. For source-pivot (src-pvt; xx-en) model training, the data of odd years were used. Instead the data of even years were used for pivot-target (pvt-src; en-xx) model training.

Language Pairs (src-pvt)	Sentence Pairs #	Language Pairs (pvt-tgt)	Sentence Pairs #
pt-da	941,876	da-sv	865,020
pt-de	939,932	de-sv	814,678
pt-el	591,429	el-sv	558,765
pt-es	934,783	es-sv	827,964
pt-fi	950,588	fi-sv	872,182
pt-fr	954,637	fr-sv	860,272
pt-it	900,185	it-sv	813,000
pt-nl	945,997	nl-sv	864,675

Table2. Training data for experiments via different pivot languages. For source-pivot (src-pvt; pt-xx) model training, the data of odd years were used. Instead the data of even years were used for pivot-target (pvt-src; xx-sv) model training.

Test Set	Sentence #	Reference #
WMT06	2,000	1
WMT07	2,000	1
WMT08	2,000	1

Table3. Statistics of test sets.

[1] http://www.statmt.org/europarl/

	TGT / SRC	da	de	el	es	fi	fr	it	nl	pt	sv
Baseline	**da**	-	19.83	20.46	27.59	14.76	24.11	20.49	22.26	24.38	28.33
RW		-	**20.15***	**21.02***	**28.29***	**15.63***	**24.71***	**20.82***	**22.57***	**24.88***	**28.87***
Baseline	**de**	23.35	-	19.83	26.21	12.72	22.43	18.82	23.74	23.05	21.17
RW		**23.69***	-	20.05	**26.70***	**13.57***	**22.78***	**19.32***	**24.11***	**23.35***	21.27
Baseline	**el**	23.24	18.12	-	32.28	13.31	27.35	23.19	20.80	27.62	22.70
RW		**23.82***	**18.49***	-	32.48	**14.08***	**27.67***	**23.63***	**21.26***	27.86	**23.15***
Baseline	**es**	25.34	19.67	27.24	-	13.93	32.91	27.67	22.37	34.73	24.83
RW		**26.07***	**20.17***	27.52	-	**14.61***	33.16	27.92	**22.85***	34.93	**25.50***
Baseline	**fi**	18.29	13.20	14.72	20.17	-	17.52	14.76	15.50	17.30	16.63
RW		**18.63***	13.40	**15.00***	**20.48***	-	**17.84***	15.01	**16.04***	**17.68***	16.79
Baseline	**fr**	25.67	20.02	26.58	37.50	13.90	-	28.51	22.65	33.81	24.64
RW		**26.51***	**20.45***	26.75	**37.80***	**14.75***	-	28.71	**23.33***	33.93	**25.59***
Baseline	**it**	22.63	17.81	24.24	34.36	13.20	30.16	-	21.37	30.84	22.12
RW		**23.27***	**18.40***	**24.66***	**35.42***	**14.11***	**30.48***	-	**21.81***	**30.92***	**22.64***
Baseline	**nl**	22.49	19.86	18.56	24.69	11.96	21.48	18.36	-	21.71	19.83
RW		22.76	**20.45***	**19.10***	**25.19***	**12.63***	**22.05***	**18.67***	-	**22.13***	**22.17***
Baseline	**pt**	24.08	19.11	25.30	36.59	13.33	32.47	28.08	21.52	-	22.90
RW		**25.29***	**19.83***	**26.20***	**37.13***	**14.21***	**32.78***	**28.44***	**22.46***	-	**23.90***
Baseline	**sv**	31.24	20.26	22.06	29.21	15.39	25.63	21.25	22.30	25.60	-
RW		**31.75***	**20.74***	**22.59***	**29.87***	**16.13***	**26.18***	**21.81***	**22.62***	**26.09***	-

Table4. Experimental results on Europarl with different translation directions (BLEU% on WMT08). RW=Random Walk. * indicates the results are significantly better than the baseline ($p<0.05$).

Several test sets have been released for the Europarl corpus. In our experiments, we used WMT2006[2], WMT2007[3] and WMT2008[4] as our test data. The original test data includes 4 languages and extended versions with 11 languages of these test sets are available by the EuroMatrix[5] project. Table 3 shows the test sets.

5.2.2. Experiments on Different Translation Directions

We build 180 pivot translation systems[6] (including 90 baseline systems and 90 random walk based systems) using 10 source/target languages and 1 pivot language (English).

The baseline system was built following the traditional triangulation pivot approach. Table 4 lists the results on Europarl training data. Limited by the length of the paper, we only show the results on WMT08, the tendency of the results on WMT06 and WMT07 is similar to WMT08.

Several observations can be made from the table.

1. In all 90 language pairs, our method achieves general improvements over the baseline system.

2. Among 90 language pairs, random walk based approach is significantly better than the baseline system in 75 language pairs.

3. The improvements of our approach are not equal in different translation directions. The improvement ranges from 0.06 (it-es) to 1.21 (pt-da). One possible reason is that the performance is related with the source and target language. For example, when using Finnish as the target language, the improvement is significant over the baseline. This may be caused by the great divergence between Uralic language (Finnish) and Indo-European language (the other European language in Table4). From the table we can find that the translation between languages in different language family is worse than that in some language family. But our random walk approach can im-

[2] http://www.statmt.org/wmt06/shared-task/
[3] http://www.statmt.org/wmt07/shared-task.html
[4] http://www.statmt.org/wmt08/shared-task.html
[5] http://matrix.statmt.org/test_sets/list
[6] Given N languages, a total of N*(N-1) SMT systems should be build to cover the translation between each language.

prove the performance of translations between different language families.

5.2.3. Experiments via Different Pivot Languages

In addition to using English as the pivot language, we also try some other languages as the pivot language. In this sub-section, experiments were carried out from translating Portuguese to Swedish via different pivot languages.

Table 5 summarizes the BLEU% scores of different pivot language when translating from Portuguese to Swedish. Similar to Table 4, our approach still achieves general improvements over the baseline system even if the pivot language has been changed. From the table we can see that for most of the pivot language, the random walk based approach gains more than 1 BLEU score over the baseline. But when using Finnish as the pivot language, the improvement is only 0.02 BLEU scores on WMT08. This phenomenon shows that the pivot language can also influence the performance of random walk approach. One possible reason for the poor performance of using Finnish as the pivot language is that Finnish belongs to Uralic language family, and the other languages belong to Indo-European family. The divergence between different language families led to a poor performance. Thus how to select a best pivot language is our future work.

The problem with random walk is that it will lead to a larger phrase table with noises. In this sub-section, a pre-pruning (before random walk) and a post-pruning (after random walk) method were introduced to deal with this problem.

We used a naive pruning method which selects the top N phrase pairs in the phrase table. In our experiments, we set N to 20. For pre-pruning, we prune the SP phrase table and PT phrase table before applying random walks. Post-pruning means that we prune the ST phrase table after random walks. For the baseline system, we also apply a pruning method before combine the SP and PT phrase table. We test our pruning method on pt-en-sv translation task. Table 6 shows the results.

With a pre- and post-pruning method, the random walk approach is able to achieve further improvements. Our approach achieved BLEU scores of 25.11, 24.69 and 24.34 on WMT06, WMT07 and WMT08 respectively, which is much better than the baseline and the random walk approach with pruning. Moreover, the size of the phrase table is about half of the no-pruning method. When adopting a post-pruning method, the performance of translation did not improved significantly over the pre-pruning, but the scale of the phrase table dropped to 69M, which is only about 2 times larger than the triangulation method.

Phrase table pruning is a key work to improve the performance of random walk. We plan to explore more approaches for phrase table pruning. E.g. using significance test (Johnson et al., 2007) or monolingual key phrases (He et al., 2009) to filter the phrase table.

	trans language	**WMT 06**	**WMT 07**	**WMT 08**
Baseline	pt-da-sv	23.40	22.80	22.49
RW		**24.47***	**24.21***	**23.75***
Baseline	pt-de-sv	22.72	22.21	21.76
RW		**23.12***	**23.26***	**22.35***
Baseline	pt-el-sv	22.53	22.19	21.37
RW		**23.75***	**23.22***	**22.40***
Baseline	pt-en-sv	23.54	23.24	22.90
RW		**24.66***	**24.22***	**23.90***
Baseline	pt-es-sv	23.58	23.37	22.80
RW		**24.65***	**24.10***	**23.77***
Baseline	pt-fi-sv	21.06	20.06	20.26
RW		21.17	**20.42***	20.28
Baseline	pt-fr-sv	23.55	23.09	22.89
RW		**24.75***	**24.15***	**23.96***
Baseline	pt-it-sv	23.65	22.96	22.79
RW		**24.74***	**24.18***	**24.02***
Baseline	pt-nl-sv	21.87	21.83	21.36
RW		**23.06***	**22.76***	**22.29***

Table5. Experimental results on translating from Portuguese to Swedish via different pivot language. RW=Random Walk. * indicates the results are significantly better than the baseline ($p<0.05$).

	WMT 06	**WMT 07**	**WMT 08**	**Phrase Pairs #**
Baseline	23.54	23.24	22.90	46M
+pruning	**24.05***	**23.70***	**23.59***	**32M**
RW	24.66	24.22	23.90	215M
+pre-pruning	25.11	24.69	24.34	109M
+post-pruning	**25.19***	**24.79***	**24.41***	**69M**

Table6. Results of Phrase Table Filtering

5.3 Experiments on Spoken Language

The European languages show various degrees of similarity to one another. In this sub-section, we consider translation from Chinese to Spanish with English as the pivot language. Chinese belongs to Sino-Tibetan Languages and English/Spanish belongs to Indo-European Languages, the gap between two languages is wide.

A pivot task was included in IWSLT 2008 in which the participants need to translate Chinese to Spanish via English. A Chinese-English and an English-Spanish data were supplied to carry out the experiments. The entire training corpus was tokenized and lowercased. Table 7 and Table 8 summarize the training data and test data.

Table 9 shows the similar tendency with Table 4. The random walk models achieved BLEU% scores 32.09, which achieved an absolute improvement of 2.08 percentages points on BLEU over the baseline.

Corpus	Sentence pair #	Source word #	Target word #
CE	20,000	135,518	182,793
ES	19,972	153,178	147,560

Table 7: Training Data of IWSLT2008

Test Set	Sentence #	Reference #
IWSLT08	507	16

Table8. Test Data of IWSLT2008

System	BLEU%	phrase pairs #
Baseline	30.01	143,790
+pruning	30.25	108,407
RW	31.57	2,760,439
+pre-pruning	31.99	1,845,648
+post-pruning	**32.09***	**1,514,694**

Table9. Results on IWSLT2008

5.4 Experiments on Web Data

The setting with Europarl data is quite artificial as the training data for directly translating between source and target actually exists in the original data sets. The IWSLT data set is too small to represent the real scenario. Thus we try our experiment on a more realistic scenario: translating from Chinese to Japanese via English with web crawled data.

All the training data were crawled on the web. The scale of Chinese-English and English-Japanese is 10 million respectively. The test set was built in house with 1,000 sentences and 4 references.

System	BLEU%	phrase pairs #
Baseline	28.76	4.5G
+pruning	28.90	273M
RW	29.13	46G
+pre-pruning	29.44	11G
+post-pruning	**29.51***	**3.4G**

Table10. Results on Web Data

Table 10 lists the results on web data. From the table we can find that the random walk model can achieve an absolute improvement of 0.75 percentages points on BLEU over the baseline.

In this subsection, the training data contains parallel sentences with different domains. And the two training corpora (Chinese-English and English-Japanese) are typically very different. It means that our random walk approach is robust in the realistic scenario.

6 Discussions

The random walk approach mainly improves the performance of pivot translation in two aspects: reduces the OOVs and provides more hypothesis phrases for decoding.

6.1 OOV

Out-of-vocabulary (OOV[7]) terms cause serious problems for machine translation systems (Zhang et al., 2005). The random walk model can reduce the OOVs. As illustrated in Figure 1, the Chinese phrase "很可口henkekou" cannot be connected to any Spanish phrase, thus it is a OOV term.

We count the OOVs when decoding with triangulation model and random walk model on IWSLT2008 data. The statistics shows that when using triangulation model, there are 11% OOVs when using triangulation model, compared with 9.6% when using random walk model. Less OOV often lead to a better result.

[7] OOV refer to phrases here.

6.2 Hypothesis Phrases

To illustrate how the random walk method helps improve the performance of machine translation, we illustrate an example as follows:

- **Source: 我 想 要 枕头**
  wo xiang yao zhentou
- **Baseline trans: Quiero almohada**
- **Random Walk trans: Quiero una almohada**

For translating a Chinese sentence "我想要枕头 wo xiang yao zhentou" to Spanish, we can get two candidate translations. In this case, the random walk translation is better than the baseline system. The key phrase in this sentence is "枕头 zhentou", figure 5 shows the extension process. In this case, the article "*a*" is hidden in the source-pivot phrase table. The same situation often occurs in articles and prepositions. Random walk is able to discover the hidden relations (hypothesis phrases) among source, pivot and target phrases.

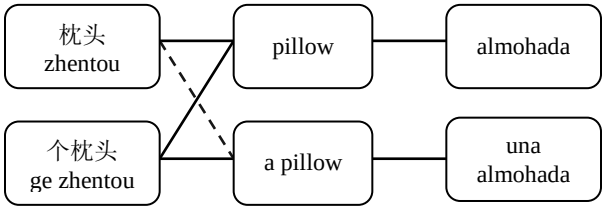


Figure 5: Phrase extension process. The dotted line indicates an implicit relation in the phrase table.

7 Conclusion and Future Work

In this paper, we proposed a random walk method to improve pivot-based statistical machine translation. The random walk method can find implicit relations between phrases in the source and target languages. Therefore, more source-target phrase pairs can be obtained than conventional pivot-based method. Experimental results show that our method achieves significant improvements over the baseline on Europarl corpus, spoken language data and the web data.

A critical problem in the approach is the noise that may bring in. In this paper, we used a simple filtering to reduce the noise. Although the filtering method is effective, other method may work better. In the future, we plan to explore more approaches for phrase table pruning.

Acknowledgments

We would like to thank Jianyun Nie, Muyun Yang and Lemao Liu for insightful discussions, and three anonymous reviewers for many invaluable comments and suggestions to improve our paper. This work is supported by National Natural Science Foundation of China (61100093), and the Key Project of the National High Technology Research and Development Program of China (2011AA01A207).

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 597-604

Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the Seventh International World Wide Web Conference*

Trevor Cohn and Mirella Lapata. 2007. Machine Translation by Triangulation: Make Effective Use of Multi-Parallel Corpora. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, pages 828-735.

Marta R. Costa-jussà, Carlos Henríquez, and Rafael E. Banchs. 2011. Enhancing Scarce-Resource Language Translation through Pivot Combinations. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1361-1365

Nick Craswell and Martin Szummer. 2007. Random Walks on the Click Graph. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239-246

Yiming Cui, Conghui Zhu, Xiaoning Zhu, Tiejun Zhao and Dequan Zheng. 2013. Phrase Table Combination Deficiency Analyses in Pivot-based SMT. In *Proceedings of 18th International Conference on Application of Natural Language to Information Systems*, pages 355-358.

Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada and Masaaki Nagata. 2011. Generalized Minimum Bayes Risk System Combination. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1356–1360

Jesús González-Rubio, Alfons Juan and Francisco Casacuberta. 2011. Minimum Bayes-risk System

Combination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1268–1277

Zhongjun He, Yao Meng, Yajuan Lü, Hao Yu and Qun Liu. 2009. Reducing SMT Rule Table with Monolingual Key Phrase. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 121-124

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving translation quality by discarding most of the phrase table. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 967–975.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *HLT-NAACL: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127-133

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In P*roceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of MT Summit X*, pages 79-86.

Philipp Koehn, Hieu Hoang, Alexanda Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, demonstration session*, pages 177–180.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 1086–1090

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computation Linguistics*, pages 311-319

Karl Pearson. 1905. The Problem of the Random Walk. *Nature*, 27(1865):294

Mydans, Seth. 2011. Across cultures, English is the word. *New York Times*.

Martin Szummer and Tommi Jaakkola. 2002. Partially Labeled Classification with Markov Random Walks. In *Advances in Neural Information Processing Systems*, pages 945-952

Kristina Toutanova, Christopher D. Manning and Andrew Y. Ng. 2004. Learning Random Walk Models for Inducting Word Dependency Distributions. In *Proceedings of the 21st International Conference on Machine Learning*.

Masao Utiyama and Hitoshi Isahara. 2007. A Comparison of Pivot Methods for Phrase-Based Statistical Machine Translation. In *Proceedings of Human Language Technology: the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 484-491

Masao Utiyama, Andrew Finch, Hideo Okuma, Michael Paul, Hailong Cao, Hirofumi Yamamoto, Keiji Yasuda, and Eiichiro Sumita. 2008. The NICT/ATR speech Translation System for IWSLT 2008. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 77-84

Haifeng Wang, Hua Wu, Xiaoguang Hu, Zhanyi Liu, Jianfeng Li, Dengjun Ren, and Zhengyu Niu. 2008. The TCH Machine Translation System for IWSLT 2008. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 124-131

Hua Wu and Haifeng Wang. 2007. Pivot Language Approach for Phrase-Based Statistical Machine Translation. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, pages 856-863.

Hua Wu and Haifeng Wang. 2009. Revisiting Pivot Language Approach for Machine Translation. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th IJCNLP of the AFNLP*, pages 154-162

Ying Zhang, Fei Huang, Stephan Vogel. 2005. Mining translations of OOV terms from the web through cross-lingual query expansion. In *Proceedings of the 27th ACM SIGIR*. pages 524-525

Improving Alignment of System Combination by Using Multi-objective Optimization

Tian Xia[+], **Zongcheng Ji**[*], **Shaodan Zhai**[+], **Yidong Chen**[++], **Qun Liu**[*], **Shaojun Wang**[+]
[++] Xiamen University, Xiamen 361005, P.R. China
[+] Wright State University, 3640 Colonel Glenn Hwy, Dayton, OH 45435, USA
[*] Institute of Computing Technology, Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{jizongcheng, liuqun}@ict.ac.cn **and** ydchen@xmu.edu.cn
{xia.7, zhai.6, shaojun.wang}@wright.edu

Abstract

This paper proposes a multi-objective optimization framework which supports heterogeneous information sources to improve alignment in machine translation system combination techniques. In this area, most of techniques usually utilize confusion networks (CN) as their central data structure to compact an exponential number of an potential hypotheses, and because better hypothesis alignment may benefit constructing better quality confusion networks, it is natural to add more useful information to improve alignment results. However, these information may be heterogeneous, so the widely-used Viterbi algorithm for searching the best alignment may not apply here. In the multi-objective optimization framework, each information source is viewed as an independent objective, and a new goal of improving all objectives can be searched by mature algorithms. The solutions from this framework, termed Pareto optimal solutions, are then combined to construct confusion networks. Experiments on two Chinese-to-English translation datasets show significant improvements, 0.97 and 1.06 BLEU points over a strong Indirected Hidden Markov Model-based (IHMM) system, and 4.75 and 3.53 points over the best single machine translation systems.

1 Introduction

System combination (SC) techniques have the power of boosting translation quality in BLEU by several percent over the best among all input machine translation systems (Bangalore et al., 2001; Matusov et al., 2006; Sim et al., 2007; Rosti et al., 2007b; Rosti et al., 2007a; Huang and Papineni, 2007; He et al., 2008; Rosti et al., 2008; He and Toutanova, 2009; Li et al., 2009; Feng et al., 2009; Pauls et al., 2009). A central data structure in the SC is the confusion network, and its quality greatly affects the final performance. He et al. (2008) proposed a new hypothesis alignment algorithm for constructing high-quality confusion networks called Indirect Hidden Markov Model (IHMM), which does better in synonym matching compared with the classic translation edit rate (TER) based algorithm (Rosti et al., 2007b; Rosti et al., 2008; Sim et al., 2007). Now, current state-of-the-art SC systems have been using IHMM or variants in their alignment algorithms more or less (Li et al., 2009; Feng et al., 2009).

Our motivation derives from an observation that in an ideal alignment of a pair of sentences, many-to-many alignments often exist. For instance, "be about to" has the same meaning with "be on the point of". Because Hidden Markov Model based alignment algorithms, e.g. IHMM for system combination, HMM in GIZA++ software for statistical machine translation (SMT) (Och and Ney, 2000; Koehn et al., 2003), are designed for one-to-many alignment, and running GIZA++ from two directions to gain better performance turns into a standard operation in SMT, therefore we are seeking a way to empower IHMM by introducing bi-directional information.

However, it appears to be intractable in an IHMM model to search the optimal solution by simply defining a new goal as a product of probabilities

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 535–544,
Seattle, Washington, USA, 18-21 October 2013.

from two directions. To bypass this problem, Liang et al. (2006) adopts a simple and effective variational inference algorithm.

Further, different alignment algorithms capture different information and linguistic phenomena for a pair of sentences, hence more information would be expected to benefit the final alignment. Liang's method may not be suitable for this expected outcome.

We propose to adopt multi-objective optimization framework to support heterogeneous information sources which may induce difficulties in a conventional search algorithm. In this framework, there exist a variety of matured multi-objective optimization algorithms, e.g. evolutionary algorithm (Deb et al., 2000; Deb et al., 2002), Tabu search (Hansen, 1997), ants colony (Engelbrecht, 2005), and simulated annealing (Serafini, 1994). In this work, we select the multi-objective evolutionary algorithm because of its public open source software (http://www.iitk.ac.in/kangal/codes.shtml). On the other hand, this framework is also totally unsupervised. It prevents weights of a linearly combined goal from training even if all information is homogeneous and applicable in a Viterbi search (Forney Jr, 1973). This framework views any useful information benefiting alignment as an independent objective, and researchers just need to write short codes for objective definitions. The search algorithm seeks for potentially better solutions which are no worse than the current solution set. The output from multi-objective optimization algorithms includes a set of solutions, called *Pareto optimal solutions*, each one being a many-to-many alignment. We then combine and normalize them into a unique one-to-one alignment to perform confusion network construction (Section 3.3).

Our work is conducted on the classic pipeline which has three modules, pair-wise hypothesis alignment, confusion network construction, and training. Now many work integrates neighboring modules to avoid propagated errors to gain improved performance. For example, Rosti et al. (2008), and Li et al. (2009) combine the first and the second module, and He and Toutanova (2009) combine all modules into one directly. Nevertheless, the classic structure also owns its merits. Because of the independence between modules, a system is relatively simple to maintain, and improvements on each module might contribute to final performance additively. Based on our work, lattice-based minimum error rate training (lattice-MERT) and minimum bayes risk training techniques (Kumar et al., 2009) could be adopted on the third module. And Feng et al. (2009) in the second module adopts a different data structure called lattice which could directly use our better many-to-many alignment for construction.

Experiments on the Chinese-to-English task on two datasets use four objectives, IHMM probability (Section 3.2.1), and alignment probability from GIZA++ (Section 3.2.2) from two directions. Results show multi-objective optimization framework efficiently integrates different information to gain approximately 1 BLEU point improvement over a strong baseline.

2 Background

We briefly give an introduction to confusion networks, and because the IHMM based alignment is an important objective in our multi-objective framework, here we also provide detailed definition of formulas for completeness of content.

2.1 Confusion Network

Table 1 shows hypotheses h_1 and h_2 are aligned to selected backbone h_0. When alignment algorithm obtains good enough results, the expected output *"he prefers apples"* is included in its corresponding confusion network in Figure 1. This suggests developing better alignment algorithm may help creating high-quality confusion networks. This also motivates us to use the BLEU of oracle hypotheses to approximately measure the quality of a set of CNs. We hereafter call it an oracle BLEU of a CN. See more in Section 5.1.

h_0 :he	feels	like	apples
h_1 :he	prefer	ε	apples
h_2 :him	prefers	to	apples

Table 1: A toy example of hypothesis alignment, where h_0 is the backbone hypothesis. $h_1$and h_2 are aligned to the backbone separately. The resulting confusion network is in Figure 1.

A confusion network $G = (V, E)$ is a directed acyclic graph with a unique source and sink vertex,

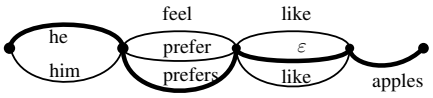


Figure 1: A classic confusion network, and the bold path the expected output.

formally a weighted finite state automation (FSA), where V is the set of nodes and E is the set of edges. Each edge is restricted to attach to a single word as well as an associated probability. A special mark ε is a place-holder denoting no word here.

2.2 IHMM-based Alignment

Indirected Hidden Markov Model (IHMM) was firstly proposed by He et. al (2008). Compared with TER-based alignment performing literal matching, IHMM supports synonym comparison in redefining emission probabilities in an IHMM model.

Let $\boldsymbol{f}^I = (f_1, \dots f_I)$ be a backbone hypothesis, and $\boldsymbol{e}^J = (e_1, \dots e_J)$ be a hypothesis aligned to the backbone, both being English sentences in our experiments. Let $\boldsymbol{a}^J = \{a_1, \dots a_j\}$ be an alignment. Suppose the a_jth word in $\boldsymbol{f}^I$ is aligned to jth word in $\boldsymbol{e}^J$, and the conditional probability that the hypothesis is generated by the backbone, shown in the upper graph of Figure 3, is given by

$$p(\boldsymbol{f}^I, \boldsymbol{e}^J) = \sum_{a^J} \prod_{j=1}^{J} \{p^t(a_j|a_{j-1}, I) p^o(e_j|f_{a_j})\} \tag{1}$$

The *distortion probability* $p^t(a_j|a_{j-1}, I)$ from position a_{j-1} to a_j, relies on jumped distance, which is computed as follows:

$$p^t(i^{'}|i, I) = \frac{c(i^{'} - i)}{\sum_{t=1}^{I} c(t - i)} \tag{2}$$

The *distortion parameters* $c(d)$ are grouped into 11 buckets, $c(\leq -4)$,$c(-3)$,$c(-2) \dots c(5)$,$c(\geq 6)$. Because all the hypotheses in system combination are in the same language, the IHMM model would support more monotonic alignments, and non-monotonic alignments will be penalized.

$$c(d) = (1 + |d - 1|)^{-K}, d = -4 \dots 6 \tag{3}$$

where K is tuned on held-out data.

Let p_0 be the probability of jumping to a $null$ word state, which is also tuned on held-out data, and the accurate transition probability becomes:

$$p^t(i^{'}|i, I) = \begin{cases} p_0 & \text{if } i^{'} = null \\ (1 - p_0) p^t(i^{'}|i, I) & \text{otherwise} \end{cases} \tag{4}$$

The *output probability* $p^o(e|f)$ from the state word f to the observation word e, also called translation probability, is a linear interpolation of semantic similarity $p^{sem}(e|f)$ and surface similarity $p^{sur}(e|f)$, and α is the interpolation factor:

$$p^o(e|f) = \alpha p^{sem}(e|f) + (1 - \alpha) p^{sur}(e|f) \tag{5}$$

When calculating *semantic similarity* $p^{sem}(e|f)$, source sentence src is needed, and a bilingual probabilistic dictionary $p^{dic}(w_1|w_2)$ is necessary.

$$p^{sem}(e|f) \approx \sum_{c \in src} p^{dic}(c|f) \cdot p^{dic}(e|c) \tag{6}$$

Note that $p^{sem}(e|f)$ has been updated with different source sentences.

The *surface similarity* $p^{sur}(e|f)$ is measured by the literal matching rate:

$$p^{sur}(e, f) = \exp\{\rho[\frac{\text{LMP}(f, e)}{\max(|f|, |e|)} - 1]\} \tag{7}$$

where $\text{LMP}(f, e)$ is the length of the longest matched prefix, and ρ is a smoothing parameter.

3 Multi-objective Optimization

Many decision making problems in the real world consider more than one objective. One natural way is to scalarize multiple objectives into one by assigning it with a weight vector. This method allows a simple optimization algorithm in many cases, while in system combination, it would cause problems.

In the first module, in order to train suitable weights of objectives, extra labeled data is needed, besides that, the efficient Viterbi algorithm for searching the optimal alignment would not work for

the alignment objectives in this work. More, the parameter training in the third module relies on the CNs constructed from the output of the first module, which increases the instability of the whole system. Therefore, an unsupervised multi-objective algorithm may be a good choice allowing for more alignment information.

There exist other alternative optimization algorithms in the multi-objective optimization framework, though the evolutionary algorithm is adopted here, we only introduce some general concepts.

3.1 Pareto Optimal Solutions

A general multi-objective optimization problem consists of a number of objectives and is associated with a number of constraints. Mathematically, the problem can be written as follows (Deb, 2001)

$$\begin{array}{lll} \text{Maximize} & f_i(x) & i = 1 \ldots M \\ \text{s.t.} & g_j(x) \leq 0 & j = 1 \ldots N \\ & h_k(x) = 0 & k = 1 \ldots K \end{array}$$

where x denotes a potential solution, its structure relying on different problems, and the number of constraints M, N, K depend on different problems. All the functions f_i, g_j, h_k map a solution x into a scalar. We will explain them in terms of system combination.

In this work, we refer to $x = \{x_{i,j} | x_{i,j} \in \{0, 1\}\}$ as a potential alignment of a pair of hypotheses, where $x_{i,j}$ is a boolean value to denote *whether the ith word in the first hypothesis is aligned to the jth word in the second hypothesis*. Here the definition of $\boldsymbol{x}$ seems different from that of $\boldsymbol{a}$ in Formula 1, and they could convert to each other. Using a line-based access style, a matrix can be unfolded as a vector. We refer to f as IHMM alignment probability (He et al., 2008) and GIZA++ alignment probability (Chen et al., 2009), total four objectives from two directions, and the larger the objectives, the better. The g_js and h_ks serve as the role of checking if $\boldsymbol{x}$ represents a legal alignment. For instance, the subscripts of $x_{i,j}$ are not in bounds.

Definition 1. *Let x, x' be two potential alignments. If $f_i(x) \geq f_i(x')$ holds for all i, we call the alignment x dominates the alignment x'. If there does not exist any alignment x'' to dominate x, we call the alignment x to be non-dominated.*

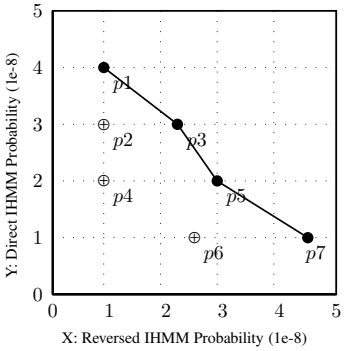


Figure 2: Sample solutions with only two objectives. Pareto Optimal Solutions p_1, p_3, p_5, p_7. Other points p_2, p_4, p_6 are dominated by at least one point in the Pareto optimal solutions.

Definition 2. *A alignment x is said to be Pareto optimal if there is no other alignment x' found to dominate x.*

In Figure 2, p_1 dominates p_2, and p_2 dominates p_4. To summarize, a point is dominated by the ones on its upper and right side with ties. In this example, p_1, p_3, p_5, p_7 are Pareto optimal.

In some cases, Pareto optimal solutions can be used for good candidate solutions. Considering the IHMM model, maximizing Y axis, the top-4 best alignments are p_1, p_2, p_3, p_4. But from the view of Pareto optimal, the top-4 alignments would be p_1, p_3, p_5, p_7 without order, which considers a greater range than a single optimization model. In our method, we just combine these Pareto optimal solutions equally into a unique alignment (Section 3.3).

Our adopted multi-objective optimization searching algorithm is the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2000; Deb et al., 2002) with an open source software (http://www.iitk.ac.in/kangal/codes.shtml). NSGA-II has a complexity of $\mathrm{O}(mn^2)$, where m is the number of objectives and n is the population size in an evolutionary algorithm.

3.2 Objectives in Evolutionary Algorithm

The optimization objectives in our experiments can be categorized as an IHMM alignment probability (He et al., 2008) and GIZA++ alignment probability

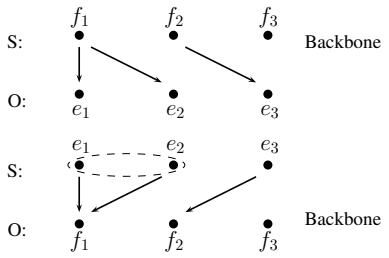


Figure 3: The same alignment $(f_1, e_1)(f_1, e_2)(f_2, e_3)$ in two IHMM models. The upper one is a typical example in IHMM, and in the bottom one, because any word in the observation is required not to correspond to two statuses, it has a minor trouble. S: status sequence, O: observation sequence.

(Chen et al., 2009), total four from two directions.

3.2.1 IHMM Probability

A typical IHMM alignment is demonstrated in the upper graph of Figure 3, where a backbone is acting the role of a status sequence. The unnormalized conditional alignment probability is $[p^t(1|null)] \cdot [p^t(1|1)p^t(2|1)] \cdot [p^o(e_1|f_1)p^o(e_2|f_1)p^o(e_3|f_2)]$. However, the same alignment $(f_1, e_1)(f_1, e_2)(f_2, e_3)$, if we change the alignment direction, the backbone being observations, would be a bit different. We offer a minor modification to Formula 1.

Look at the bottom graph of Figure 3, the observation f_1 has two statuses, e_1 and e_2 at the same time, it becomes ambiguous to compute the transitional probability between $p^t(3|1)$ and $p^t(3|2)$. This is because IHMM algorithm deals with one-to-many alignments, and MOEA permits many-to-many alignments.

We hence empirically modify the IHMM model to support many-to-many alignments. A new status is defined, rather than a single position $p^t(j|i)$, but as a set of positions $p^t(\{j\}|\{i\})$. The positions in one status need not to be adjacent to each other.

The redefined *transitional probability*

$$p^t(\{j\}|\{i\}) = \frac{1}{|\{j\}| \cdot |\{i\}|} \sum_{i,j} p^t(j|i)$$

The redefined *emission probability*

$$p^o(j|\{i\}) = \prod_i p^o(j|i)$$

We need to note that there is no guarantee on the closed property of probabilities, though these approximations prove to be effective in a practical sense. Straightforwardly, when there is only one position in a new status, the expanded IHMM degenerates to the standard IHMM.

Let us return to the second IHMM example. The new probability becomes $[p^t(1|\text{null})p^t(2|\text{null})] \cdot [\frac{1}{2}p^t(3|1)p^t(3|2) \cdot p^t(\text{null}|3)] \cdot [p^o(f_1|e_1)p^o(f_1|e_2)p^o(f_2|e_3)p^o(f_3|\text{null})]$.

3.2.2 Alignment Probability

GIZA++ considers very different and more information in alignment, we attempt to utilize them. All probabilities appearing in below formulas can be looked up in GIZA++.

Given a pair of hypotheses $\boldsymbol{f}^I = (f_1, \ldots f_I)$, $\boldsymbol{e}^J = (e_1, \ldots e_J)$, and their alignment $\boldsymbol{a}$, the alignment probability could be calculated as follows

$$p^{Giza}(\boldsymbol{e}^J|\boldsymbol{f}^I, \boldsymbol{a}) = \prod_{e_i} T(e_i|\boldsymbol{f}^I, \boldsymbol{a})$$

$$T(e_i|\boldsymbol{f}^I, \boldsymbol{a}) = \begin{cases} n(\phi_i|e_i) \sum_{(j,i)\in\boldsymbol{a}} t(e_i|f_j)a(j|i)/\phi_i & \text{if } \phi_i \neq 0 \\ n(0|e_i)t(e_i|\text{null})a(0|i) & \text{otherwise} \end{cases}$$

$$\phi_i = |\{j|(i,j) \in \boldsymbol{a}\}|$$

where ϕ_i is the fertility number, $t(e|c)$ *the translation probability* for the word pair, $z(j|i)$ *alignment probability* to show how likely a target word at position i could be translated into a source word at position j, and $n(\phi|e)$ is the *fertility probability* to show how likely a given target word e is translated into ϕ source words.

In order to increase the coverage of words, we collect all the hypothesis pairs in both the tuning set and the test set and feed them into GIZA++. This is an off-line operation, which makes it not suitable for an online translation system. In some circumstances, users submit a pile of documents in the hope of high-quality translations, thus more useful knowledge sources would be helpful. In our experiments, a pure GIZA++ based system combination does not perform as well as IHMM based, but does benefit the final translation quality if combined in our multi-objective optimization framework.

3.3 Configuration of Evolutionary Algorithm

3.3.1 Encoding

Given a sentence pair $<\boldsymbol{f}^I, \boldsymbol{e}^J>$, we define a two-dimensional matrix $\boldsymbol{x} = \{z_{i,j}|z_{ij} \in \{0,1\}\}$ to encode a set of possible alignments. Using a line-based access style, the matrix could be unfolded as a vector with $|I| \cdot |J|$ bits of length.

3.3.2 Initialization

Because in NSGA-II software the initial population are generated at random. In order to make NSGA-II more consistent and flexible, better initial seeds should be fed with, thus we combine an existing word alignment results as input. Here we use together two N-best lists generated from directional HMM and reversed HMM respectively for initialization.

3.3.3 Normalization of Pareto Optimal Solutions

Multi-objective optimization algorithms do not pose weights on objectives, thus they output a set of so-called Pareto optimal solutions, each of which is a many-to-many alignment. We can understand them as an N-best alignment list without explicit preferences. We also empirically compare it with the idea that directly cuts an N-best list from the IHMM based alignment.

We describe a two-stage strategy for normalization. Firstly, we use a simple and effective voting strategy to combine a set of many-to-many alignments into a single many-to-many alignment, and Secondly we normalize it into a one-to-one alignment for confusion network construction. In the first stage, we count the number of word-to-word alignments on each position pair (i, j). If there is more than a half number of alignments, then we output 1, otherwise 0. In the second stage, if any word relates to more than one word alignment, the one with the highest posterior probability is selected (He et al., 2008; Feng et al., 2009). The posterior probabilities can be computed in a classic forward-backward procedure in IHMM (He et al., 2008).

4 Training and Decoding

Our work does not change the classic pipeline, thus the model and features are nearly identical to the ones in (Rosti et al., 2007b; He et al., 2008), which are modeled in a log-linear fashion in Eq. 8. Translation on a CN is just a concatenation of edges traversed, on which 4 categories of features are defined.

1. word posterior probabilities. In Eq. 8, $p(w|sys, span)$ are word confidence scores. If the word w comes from the kth hypothesis of thesys-th system, the raw score should be $\frac{1}{k+1}$, and then it would be normalized by the same sys and $span$. The same word coming from different systems owns a different score, so there are sys system weights λ_{sys}.

2. logarithm of language model score, $L(h)$.

3. number of $null$ edge, Num_{null}.

4. number of words, Num_w.

$$\begin{aligned} log(h) &= \sum_{span} \log(\sum_{sys} \lambda_{sys} p(w|sys, span)) \\ &+ w_0 L(h) + w_1 Num_{null} + w_2 Num_w \end{aligned} \quad (8)$$

Decoding a confusion network is straightforward, traversing each node from left to right, and the beam search algorithm will retain for each node an N-best list. The final N-best can be acquired following (Huang and Chiang, 2005).

The training process follows minimum error rate training (MERT) described in (Och, 2003; Koehn et al., 2003). In each iteration, the Powell algorithm would attempt to predict the optimal parameters on the cumulative N-best list.

5 Experiments

We evaluate our method in two datasets in the Chinese-to-English task. In the first one, NIST MT 2002 and 2005 are used for tuning and testing respectively, and in the second, the newswire part of MT 2006 and 2008 are for tuning and testing. A 5-gram language model is trained on the Xinhua portion of the Gigaword corpus. We report the case-sensitive NIST-BLEU score.

Four single machine translation systems participating in the system combination consist of a BTG-based system using a Max-Entropy based reordering model, a hierarchical phrase-based system, a Moses decoder and a syntax-based system. 10-best unique hypotheses from a single system on the development

SYSTEM	MT 2005	MT 2008(news)
best single	0.3207	0.3016
IHMM*	0.3585(+3.78%)	0.3263(+2.47%)
IncIHMM	0.3639(+4.32%)	0.3320(+3.04%)
GIZA++	0.3438(+2.31%)	0.3166(+1.50%)
PPBD	0.3619(+4.10%)	0.3306(+2.90%)
N-best IHMM	0.3590(+3.83%)	0.3270(+2.54%)
dH+rH	0.3604	0.3284
dH+dT	0.3610	0.3290
dH+rH+dT	0.3609	0.3289
dH+rH+rT	0.3630*(+4.27%)	0.3320$*$(+3.04%)
dH+rH+dT+rT	**0.3682****(+4.75%)	**0.3369****(+3.53%)

Table 2: PPBD is a posterior probabilistic-based decoding (section 5.3). N-best IHMM simulates the Pareto optimal solutions in our method (section 5.3). The last five systems adopt different objective combinations. The improvement percents in parentheses are compared to the best single. dH: directed IHMM, rH: reversed IHMM, dT: directed translation probability, rT: reversed translation probability. ** significance at 0.01 level, and * significance at 0.05 level over the IHMM model.

and test sets are collected as the input of the system combination.

Our baseline systems are described as follows. Two main baseline systems are IHMM based and incremental IHMM (Li et al., 2009). The first system differs from our method just in hypothesis alignment algorithm, and the second combines the first and second module of the system combination pipeline.

Because our method utilizes bidirectional information, we also provide another two alternative systems for comparison, which are GIZA++ based alignment and the posterior probability based alignment (Liang et al., 2006). Finally, we also provide an N-best alignment IHMM system, which combines an N-best alignment list to simulate the Pareto optimal solutions in our method.

The method that linearly combines all objectives is not listed as our baseline like (Duh et al., 2012) does, because their algorithm finds the best weighted solution in a fixed and small solution set, while in our problem, the solution space is a trellis-style structure consisting of an exponential number of solutions, and no efficient algorithms apply here.

The IHMM based alignment utilizes typical settings (He et al., 2008; Feng et al., 2009). The smoothing factor for the surface similarity model, and $\rho = 3$ the controlling factor for the distortion model, $K = 2$. The bilingual probabilistic dictionary is trained in the FBIS corpus which includes about 230k parallel sentence pairs. GIZA++ based system is to run GIZA++ from two directions to align all the hypotheses, and make the intersection using grow-diag-final heuristics (Koehn et al., 2003). The many-to-many alignments are normalized with the same method with ours. Our system employs NSGA-II software to realize the MOEA algorithm. The main parameters, generation number, cross probability and mutation probability, and population size, are empirically set as 100, 0.9, 0.001 and 40, and we examine the influence of difference populations sizes in the full system combination.

5.1 The Quality of Confusion Networks

This experiment shows the relationship between hypothesis alignment and confusion network. Intuitively, we expect a better hypothesis alignment would reduce the error in constructing confusion networks, and then improve the final translation quality.

We first use the *alignment error rate* (AER) (Och and Ney, 2000), which is widely used to measure the quality of hypothesis alignment. The smaller, the better. For convenience, we only examine exact literal matching. IHMM based alignment reaches around 0.15 in AER, and our method 0.145.

As the AER may not vividly reflect the relations between alignment and the final BLEU of systems, and the quality of confusion network is hard to measure directly, we assume that the quality of confusion networks could be measured by the oracle hypotheses that could be generated from them. We test the BLEU of the oracle hypotheses.

From this angle, we demonstrate several oracle BLEU of CNs generated from some conventional alignment algorithms. The results are shown in Table 3.

We find the confusion network from IHMM based alignment (He et al., 2008) is better than that from TER based alignment (Rosti et al., 2007b) by about 1 point in both two datasets. These quantities agree with the final improvements in the BLEU score in (He et al., 2008). As confusion networks from MOEA based alignment also show superiority over

alignment	MT02	MT05
GIZA++	0.5690	0.5228
TER	0.5720	0.5270
IHMM	0.5883	0.5382
IncIHMM	0.5931	0.5453
MOEA	**0.6017**	**0.5526**

Table 3: Oracle BLEUs of CNs. GIZA++: invoking GIZA++ software. TER: minimum translation edit rate. IHMM: indirect hidden markov model. IncIHMM: incremental indirect hidden markov model. MOEA: multi-objective evolution algorithm.

that from IHMM based in the oracle BLEU, we expect our final translation quality would be improved.

In Table 3, GIZA++ and TER perform similarly, because the former is more capable of tackling many-to-many alignments over the latter, while latter based might obtain relatively more precise alignment information. Both of the two do not consider synonym matching compared to IHMM.

Our method and IncIHMM overpass IHMM on this metric due to different strategies. Obtaining better hypothesis alignment or better construction of confusion networks benefit the quality of CNs.

5.2 Different Objective Combinations

As our framework is convenient to support different alignment information, we test the influence of different objective combinations to the final translation quality. We adopt four objectives to depict the candidate alignment, directed IHMM probability (dH), reversed IHMM probability (rH), directed alignment probability (dT), and reversed alignment probability (rT). Table 2 demonstrates all the results.

We can see that the IHMM based system outperforms the GIZA++ based system by about 1-1.5 points in BLEU, which agrees with the difference of oracle BLEU in Table 1. From (He et al., 2008), the IHMM based system outperforms the TER based by 1 point, which also agrees with our results in Table 1. Our system, using dH + rH + dT + rT, improves BLEU score by about 1 points over the IHMM based system. This comparison verifies our assumption, improving the quality of the confusion network does improve system performance.

The different feature combinations exhibit interesting results. The system with dH + rH + dT is 0.05 point better than the system with dH + rH, and the system dH + rH + rT is 0.3 point better than system with dH + rH, so the contributions of feature dT and rT are 0.05 and 0.3 respectively. While the two features are used together in the fourth system, the contribution is about 0.8 point, rather than 0.35. This phenomenon also proves the correlations between different features.

Our method explores a way to integrate GIZA++ and IHMM, and is supportive of useful features. Compared to the classic and powerful IHMM based system, we obtained an improvement of 0.97 points on MT 05 and 1.06 points on news of MT 2008, and equivalently over the best single system by 4.75 points and 3.53 points respectively. More, compared with the incremental IHMM, our system also shows moderate improvement, though not much. We hope these two ideas could be effectively combined in the future work.

5.3 Comparison with Other Bi-directional Alignment Methods

Our method introduces multiple alignment information into system combination to obtain improvements, thus it would be interesting to explore other alternative methods for utilizing this information. We provide three alternative methods similar to our motivations, and they fall into two categories.

The first category is from the angle of bi-directional alignment. We use GiZA++ alignment and the posterior probability decoding-based alignment for comparison. The basic idea for the latter is setting a word-to-word alignment $x_{i,j}$ as 1, if its approximate posterior marginal probability $q(x_{i,j}, x) = p_d(x_{i,j}|x, \theta_d) \cdot p_r(x_{i,j}|x, \theta_r)$ is greater than a threshold δ, where p_d and p_r are posterior marginal probabilities from directed and reversed IHMM models, which could be conveniently computed with a forward-backward algorithm, and the δ is tuned on a validation-set optimized data. We just list some δ values to examine its best performance shown in Table 4.

The second class is because our method combines the Pareto optimal solutions that consist of several candidate alignments, thus for fairness we also use a 100-best outputs from the directed IHMM model and conduct the same normalization technique.

The general results are shown in Table 2. We can

δ	**MT 2005**	**MT 2008**
IHMM	0.3585	0.3263
0.15	0.3556	**0.3391**
0.2	**0.3619**	**0.3306**
0.25	0.3575	0.3278
0.3	0.3608	0.3259

Table 4: Posterior decoding. When threshold δ are set to suitable values, simple bi-directional alignment could overpass the baseline.

see that, GIZA++ leads to the worst performance, which can be explained as GIZA++ does not support synonym matching like IHMM. The N-best IHMM has a minor improvement over the IHMM method. We found differences in the N-best list are not obvious enough. In comparison, the posterior decoding method brings relatively significant improvements on both datasets. However, the threshold δ must be selected suitably. Table 4 lists the ideal results, which will be hampered when tuning on a validation set.

All of the three candidate methods can not conveniently support extra alignment information, and a linear model poses restrictions on features to get an efficient decoding, the multi-objective optimization may be a good selection as an inference algorithm in many circumstances.

5.4 Population Size

We test the influence of final translation quality and time consumed by different population size.

population size	**BLEU** MT 2005
20	0.3597
40	0.3682
60	0.3655

Table 5: Big population size consumes more CPU time. In our experiments, we use a multi-thread technique to speed up the alignment, and choose 40 as the parameter to leverage the time and BLEU.

We expect enlarging the population size would improve the translation quality, but the BLEU in population size set as 60 does not overpass when set as 40. We conjecture that, in our code, if the N-best size from IHMM (we set as 50-best) does not reach the population size, we would use randomly generated seeds, which may hamper the performance of MOEA. We also tried a larger population in MOEA, but did not receive obvious improvement on performance.

We exerted a hard restriction on the genes in evolutionary algorithm, that is many-to-many discontiguous alignment is forbidden. This trick speeds up running by about 20 times, and does not harm system performance. Now our method runs about 0.9 seconds to align a pair of hypotheses. In practice, we utilize multi-thread to speed up.

6 Conclusion

In this paper, we explore a multi-objective framework to conveniently support more useful alignment objectives to improve the hypothesis alignment. By a minor modification of the first module in the classic pipeline, we successfully combine GIZA++ and IHMM to obtain significant improvement over a powerful and state-of-the-art IHMM based system. In comparison with another genre of improving system combination by combing adjacent modules of the pipeline, more powerful incremental IHMM here, our system also show moderate improvement. Though, our best system may not overpass He and Toutanova (2009) who combine all the modules into a unified training procedure, we believe our method could boost many work on the higher modules of the pipeline to obtain a further improvement to match their work.

7 Acknowledgement

This research is partially supported by Air Force Office of Scientific Research under grant FA9550-10-1-0335, the National Science Foundation under grant IIS RI-small 1218863 and a Google research award. We thank the anonymous reviewers for their insightful comments.

References

B Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Automatic Speech Recognition and Understanding*.

Yidong Chen, Xiaodong Shi, Changle Zhou, and Qingyang Hong. 2009. A word alignment

model based on multiobjective evolutionary algorithms. *Computers and Mathematics with Applications*, 57.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, 1917:849–858.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

Kalyanmoy Deb. 2001. Multi-objective optimization. *Multi-objective optimization using evolutionary algorithms*, pages 13–46.

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *Proc. of NAACL*, pages 975–983.

Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada, and Masaaki Nagata. 2012. Learning to translate with multiple objectives. In *Proc. of ACL*, pages 1–10.

Andries P Engelbrecht. 2005. *Fundamentals of computational swarm intelligence*, volume 1. Wiley Chichester.

Yang Feng, Yang Liu, Haitao Mi, Qun Liu, and Yajuan Lü. 2009. Lattice-based system combination for statistical machine translation. In *Proc. of EMNLP*, EMNLP '09.

G David Forney Jr. 1973. The viterbi algorithm. *Proc. of the IEEE*, 61(3):268–278.

Michael Pilegaard Hansen. 1997. Tabu search for multiobjective optimization: Mots. In *Proc. of Multiple Criteria Decision Making*, pages 574–586.

Xiaodong He and Kristina Toutanova. 2009. Joint optimization for machine translation system combination. In *Proc. of EMNLP*.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-hmm-based hypothesis alignment for combining outputs from machine translation systems. In *Proc. of EMNLP*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. of IWPT*.

Fei Huang and Kishore Papineni. 2007. Hierarchical system combination for machine translation. In *Proc. of EMNLP-CoNLL*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL: Poster*, pages 177–180.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of Joint ACL and AFNLP*.

Zhifei Li and Sanjeev Khudanpur. 2009. Forest reranking for machine translation with the perceptron algorithm. *GALE book chapter on MT From Text*.

Chi-Ho Li, Xiaodong He, Yupeng Liu, and Ning Xi. 2009. Incremental hmm alignment for mt system combination. In *Proc. of Joint ACL and AFNLP*.

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

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. of EACL*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. *Proc. of ACL-08: HLT*, pages 192–199.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. pages 440–447, October.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, pages 160–167.

Adam Pauls, John DeNero, and Dan Klein. 2009. Consensus training for consensus decoding in machine translation. In *Proc. of EMNLP*.

Antti-Veikko I Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007a. Combining outputs from multiple machine translation systems. In *Proc. of NAACL-HLT*.

Antti-Veikko I Rosti, Spyros Matsoukas, and Richard Schwartz. 2007b. Improved word-level system combination for machine translation. In *Proc. of ACL*, volume 45.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2008. Incremental hypothesis alignment for building confusion networks with application to machine translation system combination. In *Proc. of WSMT*.

Paolo Serafini. 1994. Simulated annealing for multi objective optimization problems. In *Proc. of Multiple Criteria Decision Making*, pages 283–292. Springer.

Khe Chai Sim, William J Byrne, Mark JF Gales, Hichem Sahbi, and Phil C Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proc. of ICASSP*, volume 4.

Yong Zhao and Xiaodong He. 2009. Using n-gram based features for machine translation system combination. In *Proc. of NAACL: Short Papers*, pages 205–208.

Flexible and Efficient Hypergraph Interactions for Joint Hierarchical and Forest-to-String Decoding*

Martin Čmejrek†‡
†IBM Prague Research Lab
V Parku 2294/4
Prague, Czech Republic, 148 00
martin.cmejrek@us.ibm.com

Haitao Mi‡ and **Bowen Zhou**‡
‡IBM T. J. Watson Research Center
1101 Kitchawan Rd
Yorktown Heights, NY 10598
{hmi,zhou}@us.ibm.com

Abstract

Machine translation benefits from system combination. We propose *flexible interaction of hypergraphs* as a novel technique combining different translation models within one decoder. We introduce features controlling the interactions between the two systems and explore three interaction schemes of hiero and forest-to-string models—specification, generalization, and interchange. The experiments are carried out on large training data with strong baselines utilizing rich sets of dense and sparse features. All three schemes significantly improve results of any single system on four testsets. We find that specification—a more constrained scheme that almost entirely uses forest-to-string rules, but optionally uses hiero rules for shorter spans—comes out as the strongest, yielding improvement up to 0.9 (T -B)/2 points. We also provide a detailed experimental and qualitative analysis of the results.

1 Introduction

Recent years have witnessed the success of various statistical machine translation (SMT) models using different levels of linguistic knowledge–phrase (Koehn et al., 2003), hiero (Chiang, 2005), and syntax-based (Liu et al., 2006; Galley et al., 2006). System combination became a promising way of building up synergy from different SMT systems and their specific merits.

Numerous efforts that have been proposed in this field recently can be broadly divided into two categories: *Offline system combination* (Rosti et al., 2007; He et al., 2008; Watanabe and Sumita, 2011; Denero et al., 2010) aims at producing consensus translations from the outputs of multiple individual systems. Those outputs usually contain k-best lists of translations, which only explore a small portion of the entire search space of each system. This issue is well addressed in *joint decoding* (Liu et al., 2009), or *online system combination*, showing comparable improvements to the offline combination methods. Rather than finding consensus translations from the outputs of individual systems, joint decoding works with different grammars at the decoding time. Although limited to individual systems sharing the same search paradigm (e.g. *left-to-right* or *bottom-up*), joint decoding offers many potential advatages: search through a larger space, better efficiency, features designed once for all subsystems, potential cross-system features, online sharing of partial hypotheses, and many others.

Different approaches have different strengths in general–hiero rules are believed to provide reliable lexical coverage, while tree-to-string rules are good at non-local reorderings. Different contexts present different challenges–noun phrases usually follow the adjacency principle, while verb phrases require more challenging reorderings. In this work, we study different schemes of interaction between translation models, reflecting their specific strengths at different (syntactic) contexts. We make five new contributions:

> **First**, we propose a framework for joint decoding by means of flexible combination of translation hypergraphs, allowing for detailed con-

*M. Č and H. M. contributed equally to this work.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 545–555,
Seattle, Washington, USA, 18-21 October 2013.

trol of interactions between the different systems using soft constraints (Section 3).

Second, we study three interaction schemes– special cases of joint decoding: *generalization*, *specification*, and *interchange* (Section 3.3).

Third, instead of using a tree-to-string system, we use a much stronger forest-to-string system with fuzzy match of nonterminal categories (Section 2.1).

Fourth, we train strong systems on a large-scale data set, and test all methods on four test sets. Experimental results (Section 6) show that our new approach brings improvement of up to 0.9 points in terms of (T − B)/2 over the best single system.

Fifth, we conduct a comprehensive experimental analysis, and find that joint decoding actually prefers tree-to-string rules in both shorter and longer spans. (Section 6.3).

The paper is organized as follows: We briefly review the individual models in Section 2, describe the method of joint decoding using three alternative interaction schemes in Section 3, describe the features controlling the interactions and fuzzy match in Section 4, review the related work in Section 5, and finally, describe our experiments and give detailed discussion of the results in Section 6.

2 Individual Models

Our individual models are two state-of-the-art systems: a hiero model (Chiang, 2005), and a forest-to-string model (Mi et al., 2008; Mi and Huang, 2008).

We will use the following example from Chinese to English to explain both individual and joint decoding algorithms throughout this paper.

tǎolùn	hùi	zěnmeyàng
discussion/NN	will/VV	how/VV
discuss/VV	meeting/NN	

There are several possible meanings based on the different POS tagging sequences:

1: NN VV VV: *How is the discussion going?*
2: VV NN VV: *Discuss about the meeting.*
3: NN NN VV: *How was the discussion meeting?*
4: VV VV VV: *Discuss what will happen.*

id	rule
r_1	VV(tǎolùn) → discuss
r_2	NP(tǎolùn) → the discussion
r_3	NP(hùi) → the meeting
r_4	VP(zěnmeyàng) → how
r'_4	VP(zěnmeyàng) → about
r_5	IP(x_1:NP x_2:VP) → x_2 x_1
r_6	IP(x_1:VV x_2:IP) → x_1 x_2
r_7	IP(x_1:NP VP(VV(hùi) x_2:VP)) → x_2 is x_1 going
r_{11}	X(x_1:X zěnmeyàng) → how was x_1
r_{12}	X(zěnmeyàng) → what
r_{13}	X(tǎolùn hùi) → the discussion meeting
r_{14}	X(hùi x_1:X) → x_1 will happen
r_{15}	S(x_1:S x_2:X) → x_1 x_2

Table 1: Translation rules. Tree-to-string (r_1–r_7), hiero (r_{11}–r_{14}), vanilla glue (r_{15}).

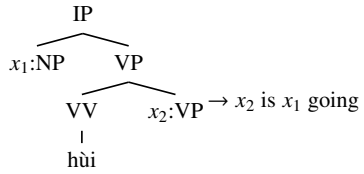


Figure 1: Tree-to-string rule r_7.

Table 1 shows translation rules that can generate all four translations. We will use those rules in the following sections.

2.1 Forest-to-string

Forest-to-string translation (Mi et al., 2008) is a linguistic syntax-based system, which significantly improves the translation quality of the tree-to-string model (Liu et al., 2006; Huang et al., 2006) by using a packed parse forest as the input instead of a single parse tree.

Figure 1 shows a tree-to-string **translation rule** (Huang et al., 2006), which is a tuple $\langle lhs(r), rhs(r), \psi(r)\rangle$, where $lhs(r)$ is the source-side tree fragment, whose internal nodes are labeled by nonterminal symbols (like NP and VP), and whose frontier nodes are labeled by source-language words (like "hùi") or variables from a set $\mathcal{X} = \{x_1, x_2, \ldots\}$; $rhs(r)$ is the target-side string expressed in target-language words (like "going") and variables; and $\psi(r)$ is a mapping from $\mathcal{X}$ to nonterminals. Each

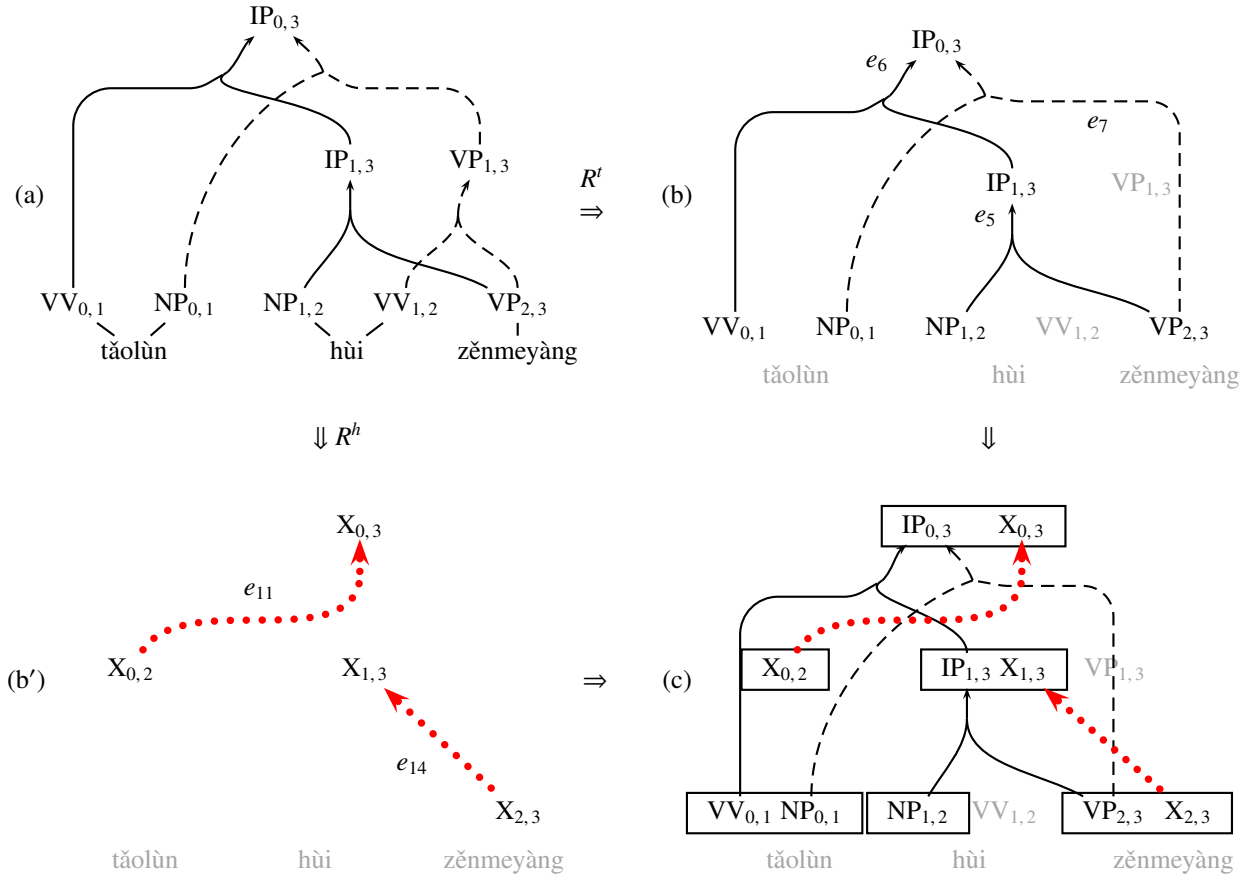


Figure 2: Parse and translation hypergraphs. (a) The parse forest of the example sentence. Solid hyperedges denote the 1-best parse. (b) The corresponding translation forest F^t after applying the tree-to-string translation rule set R^t. Target lexical content is not shown. Each translation hyperedge (e.g. e_7) has the same index as the corresponding rule (r_7). Gray nodes (e.g. $\text{VP}_{1,3}$) became inaccessible due to the insufficient rule coverage. (b′) The translation forest F^h after applying the hierarchical rule set R^h to the input sentence. (c) The combined translation forest H^m obtained by superimposing b and b′. The nodes within each solid box share the same span. See Figure 3 for an example of the internal structure of a box. The forest-to-string system can produce the translation 1 (dashed derivation: r_2, r_4 and r_7) and 2 (solid derivation: r_1, r_3, r'_4, r_5, and r_6). Hierarchical rules generate the translation 3 (r_{11} and r_{13}). The translation 4 is available by using joint decoding at $\text{X}_{1,3} \rightarrow \text{IP}_{1,3}$ with the derivation: r_1, r_6, r_{12}, and r_{14}.

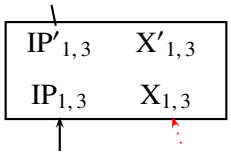
variable $x_i \in \mathcal{X}$ occurs *exactly once* in $lhs(r)$ and *exactly once* in $rhs(r)$. Take the rule r_7 in Figure 1 for example, we have:

$$lhs(r_7) = \text{IP}(x_1\text{:NP VP(VV(hùi) } x_2\text{:VP)}),$$
$$rhs(r_7) = x_2 \text{ is } x_1 \text{ going},$$
$$\psi(r_7) = \{x_1 \mapsto \text{NP}, x_2 \mapsto \text{VP}\}.$$

Typically, a forest-to-string system performs translation in two steps (shown in Figure 2): parsing and decoding. In the parsing step, we convert the source language input into a *parse forest* (a). In the decoding step, we first convert the parse forest into a *translation forest* F^t in (b) by using the fast pattern-matching technique (Zhang et al., 2009). For example, we pattern-match the rule r_7 rooted at $\text{IP}_{0,3}$, in such a way that x_1 spans $\text{NP}_{0,1}$ and x_2 spans $\text{VP}_{2,3}$, and add a translation hyperedge e_7 in (b). Then the decoder searches for the best derivation on the translation forest and outputs the target string.

2.2 Hiero

Hiero (hierarchical phrase-based) model (Chiang, 2005) acquires rules of synchronous context-free grammars (SCFGs) from word-aligned parallel data, and uses plain sequences of words as the input, without any syntactic information.



scheme	interaction edges in supernode
Generalization	IP′1, 3 X′1, 3 / IP1, 3 X1, 3
Specification	IP′1, 3 X′1, 3 / IP1, 3 X1, 3
Interchange	IP′1, 3 X′1, 3 / IP1, 3 X1, 3

Figure 3: Three interaction schemes for joint decoding. Details of the interaction supernode for span $(1, 3)$ shown in Figure 2 (c). Soft constraints control the transitions.

SCFG can be formalized as a set of tuples $\langle lhs(r), rhs(r), \phi(r)\rangle$, where $lhs(r)$ is the source-side one-level CFG, whose root is X or S, and whose frontier nodes are labeled by source-language words (like "hùi") or variables from a set $\mathcal{X} = \{x_1, x_2, \ldots\}$; $rhs(r)$ is the target-side string expressed in target-language words (like "going") and variables; and $\phi(r)$ is a mapping from $\mathcal{X}$ to nonterminals. Table 1 shows examples of hiero rules r_{11}–r_{15}.

Although different on source side, hiero decoding can be formalized equally as forest-to-string decoding: First, pattern-match the input sentence into a *translation forest* F^h. For example, since the rule r_{11} matches "zěnmeyàng" such that x_1 spans the first two words, add a hyperedge e_{11} in Figure 2 (b′). Then search for the best derivation over the translation forest.

3 Joint Decoding

The goal of joint decoding is to let different MT models collaborate within the framework of a single decoder. This can be done by combining translation hypergraphs of the different models at the decoding time, so that online sharing of partial hypotheses overcomes weaknesses and boosts strengths of the systems combined.

As both forest-to-string and hiero produce translation forests that share the same *hypergraph* structure, we first formalize the hypergraph, then we introduce an algorithm to combine different hypergraphs, and finally we describe three joint decoding schemes over the merged hypergraph.

3.1 Hypergraphs

More formally, a hypergraph H is a pair $\langle V, E\rangle$, where V is the set of **nodes**, and E the set of **hyperedges**. For a given sentence $w_{1:l} = w_1 \ldots w_l$, each node $v \in V$ is in the form of $Y_{i,j}$, where Y is a nonterminal in the context-free grammar[1] and i, j, $0 \le i < j \le l$, are string positions in the sentence $w_{1:l}$, which denote the recognition of nonterminal Y spanning the substring from positions i through j (that is, $w_{i+1} \ldots w_j$). Each hyperedge $e \in E$ is a tuple $\langle tails(e), head(e), target(e)\rangle$, where $head(e) \in V$ is the *consequent node* in the deductive step, $tails(e) \in V^*$ is the list of *antecedent nodes*, and $target(e)$ is a list of $rhs(r)$ for rules r such that each rule r has the same $lhs(r)$ pattern-matched at the node $head(e)$. For example, the hyperedge e_7 in Figure 2 (b) is

$$e_7 = \langle (\mathrm{NP}_{0,1}, \mathrm{VP}_{2,3}), \mathrm{IP}_{0,3}, (\mathrm{x}_2 \text{ is } x_1 \text{ going})\rangle,$$

where we can infer the mapping to be

$$\{x_1 \mapsto \mathrm{NP}_{0,1}, x_2 \mapsto \mathrm{VP}_{2,3}\}.$$

We also denote $BS(v)$ to be the set of **incoming hyperedges** of node v, which represent the different ways of deriving v. For example, $BS(\mathrm{IP}_{0,3})$ is a set of e_7 and e_6.

There is also a distinguished **root node** TOP in each hypergraph, denoting the goal item in translation, which is simply $\mathrm{TOP}_{0,l}$.

3.2 Combining Hypergraphs

We enable interaction between translation hypergraphs, such as hiero $F^h = \langle V^h, E^h\rangle$ and forest-to-string $F^t = \langle V^t, E^t\rangle$, on nodes covering the same span (e.g. $\mathrm{IP}_{1,3}$ and $\mathrm{X}_{1,3}$ in Figure 2 (c) grouped in a box). We call such groups *interaction supernodes* and show a detailed example of a supernode for span $(1, 3)$ in Figure 3.

The combination runs in four steps:

[1] In this paper, nonterminal labels X and S denote hiero derivations, other labels are tree-to-string labels.

1. For each node $v = Y_{i,j}, v \in V^h \cup V^t$, we create a new *interaction node* $v' = Y'_{i,j}$ with empty $BS(v')$. For example, we create two nodes, $\mathrm{IP}'_{1,3}$ and $\mathrm{X}'_{1,3}$, at the top of Figure 3.

2. For each hyperedge $e \in BS(v)$, $v \in V^t \cup V^h$, we replace each v in $tails(e)$ with v'. For example, e_7 becomes $\langle(\mathrm{NP}'_{0,1}, \mathrm{VP}'_{2,3}), \mathrm{IP}_{0,3}, (\mathrm{x}_2 \text{ is } x_1 \text{ going})\rangle$.

3. All the nodes and hyperedges form the merged hypergraph F^m, such as in Figure 2 (c).

4. Insert *interaction hyperedges* connecting nodes within each interaction supernode to make F^m connected again.

In the following subsection we present details of interactions and introduce three alternative schemes.

3.3 Three Schemes of Joint Decoding

Interaction hyperedges within each supernode allow the decoder either to stay within the same system (e.g. in hiero using $\mathrm{X}_{1,3} \rightarrow \mathrm{X}'_{1,3}$ in Figure 3), or to switch to the other (e.g. to forest-to-string using $\mathrm{X}_{1,3} \rightarrow \mathrm{IP}'_{1,3}$).

For example, translation 4 can be produced as follows: The source string "zěnmeyàng" is translated by the phrase rule r_{12}. The hiero hyperedge e_{14} combines it with the translation of "hùi", reaching the hiero node $\mathrm{X}_{1,3}$. Using the interaction edge $\mathrm{X}_{1,3} \rightarrow \mathrm{IP}'_{1,3}$ will switch into the tree-to-string model, so that the translation can be completed with the tree-to-string edge e_6 that connects it with a partial tree-to string translation of "tǎolùn" done by r_1.

In order to achieve more precise control over the interaction between tree-to-string and hiero derivations, we propose the following three basic interaction schemes: **generalization**, **specification**, **interchange**. The schemes control the interaction between hiero and tree-to-string models by means of soft constraints. Some schemes may even restrict certain types of transitions. The schemes are depicted in Figure 3 and their details are discussed in the following three subsections.

3.3.1 Specification

The *specification* decoding scheme reflects the intuition of using hiero rules to translate shorter spans and tree-to-string rules to reorder higher-level sentence structures. In other words, the scheme allows one-way switching from the hiero general nonterminal into the more *specific* nonterminal of a tree-to-string rule. Transitions in reverse directions are not allowed. This is achieved by inserting *specification interaction hyperedges e* leading from hiero nodes $\mathrm{X}_{i,j}$ or $\mathrm{S}_{i,j}$ into all tree-to-string interaction nodes $\mathrm{Y}'_{i,j}$ within the same supernode.

3.3.2 Generalization

In some translation domains, hiero outperforms tree-to-string systems, as was shown in experiments in Section 6. While local hiero or tree-to-string reorderings perform well, long distance reorderings proposed by tree-to-string may be too risky (e.g. due to parsing errors), so that monotone concatenation of long sequences[2] is the more reliable strategy. The *generalization* decoding scheme, complementary to the specification, is motivated by the idea of incorporating reliable tree-to-string translations for some sequences into a strong hiero translation system. This is achieved by inserting *generalization interaction hyperedges e* leading from tree-to-string nodes $\mathrm{Y}_{i,j}$ nodes into *general* hiero interaction nodes $\mathrm{X}'_{i,j}$ and $\mathrm{S}'_{i,j}$ within the same supernode.

3.3.3 Interchange

The *interchange* decoding scheme is a union of the two previous approaches. Any derivation can freely combine hiero and tree-to-string productions. Both *specification* and *generalization interaction hyperedges* are inserted leading from all hiero and tree-to-string nodes $\mathrm{X}_{i,j}$, $\mathrm{S}_{i,j}$, and $\mathrm{Y}_{i,j}$ into all interaction nodes $\mathrm{X}'_{i,j}$, $\mathrm{S}'_{i,j}$, and $\mathrm{Y}'_{i,j}$.

3.4 Fuzzy match

The translation rule set cannot usually cover all hyperedges in the parse forest, thus some nodes become inaccessible in the translation forest (e.g. $\mathrm{VP}_{1,3}$ in Figure 2). However, in the parse forest, as opposed to a 1-best tree, we can find other nodes spanning the same sequence $w_{i:j}$ (e.g. node $\mathrm{IP}_{1,3}$). In order to re-enable inaccessible nodes and to increase the variability of the translation forest, we allow reaching them from the other tree-to-string

[2]Monotone glue is the only possibility for very long spans exceeding the hiero *maxParse* treshold.

nodes within the same interaction node. This can be achieved by adding *fuzzy hyperedges* between every tree-to-string state $Y_{i,j}$ and a differently labeled tree-to-string interaction state $Z'_{i,j}$. For example, in the span (0,1), we have a fuzzy hyperedge $\mathrm{VV}_{0,1} \rightarrow \mathrm{NP}'_{0,1}$.

While interaction hyperedges combine different translation models, fuzzy hyperedges combine different derivations within the same (tree-to-string) model.

4 Interaction Features

Our baseline systems use the log-linear framework to estimate the probability $P(D)$ of a derivation D from features ϕ_i and their weights λ_i as $P(D) \propto \exp\left(\sum_i \lambda_i \phi_i\right)$. Similarly as Chiang et al. (2009), our systems use tens of dense (e.g. language models, translation probabilities) and thousands of sparse (e.g. lexical, fertility) features.

The features related to the joint decoding experiments are the costs for *specification*, *generalization*, *interchange*, and the *fuzzy match*. Let L^t be the set of the labels used by the source language parser and $L^h = \{\mathrm{S},\mathrm{X}\}$ be the labels used by hiero.

The **generalization feature**

$$\phi_{Y \rightarrow Z} = |\{e; e \in D, \exists i, j \; tails(e) = \{\mathrm{Y}_{i,j}\} \wedge head(e) = \mathrm{Z}'_{i,j}\}| \quad (1)$$

is the total number of generalization hyperedges in D going from tree-to-string states $Y \in L^t$ to hiero states $Z' \in L^h$.

The **specification feature**

$$\phi_{Z \rightarrow Y} = |\{e; e \in D, \exists i, j \; tails(e) = \{\mathrm{Z}_{i,j}\} \wedge head(e) = \mathrm{Y}'_{i,j}\}| \quad (2)$$

is the total number of specification hyperedges in D going from hiero states $Z \in L^h$ to tree-to-string states $Y' \in L^t$.

The **interchange feature** is implemented by enabling the generalization and specification features at the same time for both tuning and testing.

The **fuzzy match feature**

$$\phi_{U \rightarrow W} = |\{e; e \in D, \exists i, j \; tails(e) = \{\mathrm{U}_{i,j}\} \wedge head(e) = \mathrm{W}'_{i,j}\}| \quad (3)$$

is the total number of fuzzy match hyperedges in D going from tree-to-tree states $U \in L^t$ to tree-to-string states $W' \in L^t$. [3]

We use MIRA to obtain weights for the new features by tuning on the development set. The number of new parameters to tune can be estimated as $|L^h| \times |L^t|$ for generalization and specification, and $2 \times |L^h| \times |L^t|$ for interchange. For the fuzzy match of tree-to-string nonterminals we have $|L^t| \times |L^t|$ parameters organized as a sparse matrix, since we only consider combinations on nonterminal labels that cooccur in the data.[4]

5 Related Work

From the previous explorations of online translation model combination, we see the work of Liu et al. (2009) proposing an unconstrained combination of hiero and tree-to-string models as a special configuration of our framework, and we also replicate it.

Denero et al. (2010) combine translation models even with different search paradigms. Their approach is different, since their component systems do not interact at decoding time, instead, each of them provides its weighted translation forest first, the forests are then combined to infer a new combination model.

6 Experiment

In this section we describe the setup, present results, and analyze the experiments. Finally, we propose future directions of research.

[3] Here we allow $U = W$, which can be viewed in such a way that exact match is a special case of fuzzy match.

[4] We also carried out an alternative experiment with only three fuzzy match features estimated from the training data parse forest by Naïve Bayes by observing all spans in the training data, accumulating counts $C_s(U)$ and $C_s(U, W)$ of nonterminals (or pairs of nonterminals) heading the same span s. The first two features (one for each direction) are based on conditional probabilities:

$$\phi_{(U|W)} = -\log\left(\frac{\sum_{s \in spans} C_s(U, W)}{\sum_{s \in spans} C_s(W)}\right). \quad (4)$$

The third feature is based on joint probability:

$$\phi_{(U,W)} = -\log\left(\frac{\sum_{s \in spans} C_s(U, W)}{\sum_{s \in spans, A, B \in L^t} C_s(A, B)}\right). \quad (5)$$

The average performance drops by 0.1 (T -B)/2 points, compared to the interchange eperiment.

System		GALE-web		P1R6-web		MT08 news		MT08 web		Avg.
		B	(T-B)/2	B	(T-B)/2	B	(T-B)/2	B	(T-B)/2	(T-B)/2
Single	T2S	32.6	11.6	16.9	23.5	37.7	7.8	28.1	14.5	14.4
	Hiero	33.7	**10.2**	17.0	**23.1**	39.2	**6.3**	28.8	13.7	**13.3**
	F2S	34.0	10.3	17.3	23.2	39.6	**6.3**	29.2	**13.6**	13.4
Joint	Liu:09	34.1	9.7	17.0	23.0	38.8	6.7	29.0	13.2	13.2
	Gen.	34.4	9.7	17.8	22.6	40.0	6.1	29.6	13.1	12.9
	Spe.	35.1	**9.4**	18.1	**22.2**	40.2	**5.8**	29.6	**12.9**	**12.6**
	Int.	34.9	**9.4**	17.9	22.3	40.0	6.2	29.6	**12.9**	12.7

Table 2: All results of single and joint decoding systems.

6.1 Setup

The training corpus consists of 16 million sentence pairs available within the DARPA BOLT Chinese-English task. The corpus includes a mix of newswire, broadcast news, webblog and comes from various sources such as LDC, HK Law, HK Hansard and UN data. The Chinese text is segmented with a segmenter trained on CTB data using conditional random fields (CRF). Language models are trained on the English side of the parallel corpus, and on monolingual corpora, such as Gigaword (LDC2011T07) and Google News, altogether comprising around 10 billion words.

We use a modified version of the Berkeley parser (Petrov and Klein, 2007) to obtain a parse forest for each training sentence, then we prune it with the marginal probability-based inside-outside algorithm to contain only $3n$ CFG nodes, where n is the sentence length. Finally, we apply the forest-based GHKM algorithm (Mi and Huang, 2008; Galley et al., 2004) to extract tree-to-string translation rules from forest-string pairs.

In the decoding step, we prune the input hypergraphs to $10n$ nodes before we use fast pattern-matching (Zhang et al., 2009) to convert the parse forest into the translation forest.

We tune on 1275 sentences, each with 4 references, from the LDC2010E30 corpus, initially released under the DARPA GALE program.

All MT experiments are optimized with MIRA (Crammer et al., 2006) to maximize (T -B)/2.

We test on four different test sets: GALE-web test set from LDC2010E30 corpus (1239 sentences, 4 references), P1R6-web test set from LDC2012E124 corpus (1124 sentences, 1 reference), NIST MT08 newswire portion (691 sentences, 4 references), and NIST MT08 web portion (666 sentences, 4 references).

6.2 Results

Table 2 shows all results of single and joint decoding systems. The B score of the single hiero baseline is 39.2 on MT08-news, showing that it is a strong system. The single F2S baseline achieves comparable scores on all four test sets.

Then, for reference, we present results of joint Hiero and T2S decoding, which is, to our knowledge, a strong and competitive reimplementaion of the work described by Liu et al. (2009). Finally, we present results of joint decoding of hiero and F2S in three interaction schemes: generalization, specification, and interchange.

All three combination schemes significantly improve results of any single system on all four testsets. On average and measured in (T -B)/2, our systems improve the best single system by 0.4 (generalization), 0.7 (specification), and 0.6 (interchange).

The specification comes out as the strongest interaction scheme, beating the second interchange on 2 testsets by 0.1 and 0.4 (T -B)/2 points and on 3 testsets by 0.2 B points.

6.3 Discussion of Results

Interpretations of model behavior with thousands of parameters that may possibly overlap and interfere should be always attempted with caution. In this section we highlight some interesting observations, ac-

Specification		Generalization		Interchange			
X → ∗		∗ → X		X → ∗		∗ → X	
VP	0.069	QP	0.057	*VV*	0.062	*NN*	0.048
IP	0.059	PP	0.054	**VP**	0.044	PP	0.041
VV	0.053	*NN*	0.048	*NN*	0.034	CP	0.035
NR	0.032	DP	0.044	QP	0.025	LCP	0.035
ADVP	0.025	*NR*	0.034	ADVP	0.022	*DEG*	0.031
QP	0.023	DNP	0.032	LCP	0.021	DP	0.028
CC	0.017	NP	0.030	NP	0.018	*DEC*	0.027
DVP	0.017	*LC*	0.025	*P*	0.017	QP	0.027
NP	0.017	*DEC*	0.023	**IP**	0.016	*LC*	0.021
P	0.012	*DEG*	0.023	*NR*	0.016	NP	0.019
...	...	...	...	...	...	...	...
CS	-0.005	*VV*	-0.010	VSB	-0.004	FLR	-0.006
CP	-0.007	PRN	-0.011	*PN*	-0.004	DVP	-0.009
AD	-0.011	*PN*	-0.013	*PU*	-0.004	*BA*	-0.010
VRD	-0.012	*BA*	-0.015	*M*	-0.007	*JJ*	-0.011
PU	-0.028	**VP**	-0.015	VRD	-0.014	*AS*	-0.014
ADJP	-0.028	VRD	-0.028	DNP	-0.023	VRD	-0.017
DNP	-0.045	*JJ*	-0.035	ADJP	-0.039	ADVP	-0.021
PP	-0.064	*VC*	-0.037	PP	-0.058	*PN*	-0.033
PRN	-0.069	DFL	-0.054	DP	-0.070	DFL	-0.038
DP	-0.092	*PU*	-0.073	PRN	-0.080	*PU*	-0.103

Table 3: Examples of specification, generalization, and interchange weights. POS tags in italics.

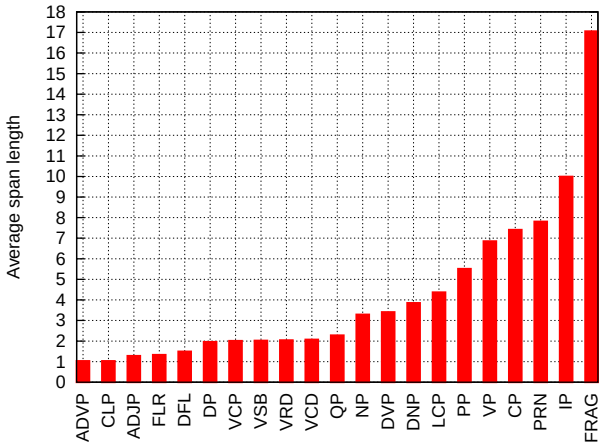


Figure 4: Average span length for selected syntactic labels on GALE-web test set.

#Interactions	Generalization	Inter. gen.
F2S → glue	5557	4202
F2S → hiero	695	1178
total gen.	6252	5380
	Specification	Inter. spec.
phrase → F2S	2763	2235
glue → F2S	946	841
hiero → F2S	683	839
total spec.	4392	3915

Table 5: Rule interactions on GALE-web test set.

companying them with our subjective judgements and speculations.

Table 3 shows the specification and generalization features tuned for the three combination schemes, then sorted by their weights $\lambda_{X\to Y}$ or $\lambda_{Y\to X}$. Features shown at the top of the table are very expensive (the system tries to avoid them), while inexpensive features are at the bottom (the system is encouraged to use them).

The most expensive interactions for the **specification** belong to constituents (IP, VP) that usually occur higher in a syntactic tree (see Figure 4 for average span lengths of selected syntactic labels), and often require non-local reorderings. This indicates that the decoder is discouraged from switching from hiero into F2S derivation at these higher-level spans.

rule type	Generalization		Specification		Interchange	
F2S	18,807	58%	19,399	70%	18,400	61%
Hiero	3,730	12%	2,330	8%	3,133	10%
Glue	7,367	23%	571	2%	4,714	16%
Phrase	2,274	7%	5,484	20%	3,868	13%
total	32,178		27,784		30,115	

Table 4: Rule counts on GALE-web test set.

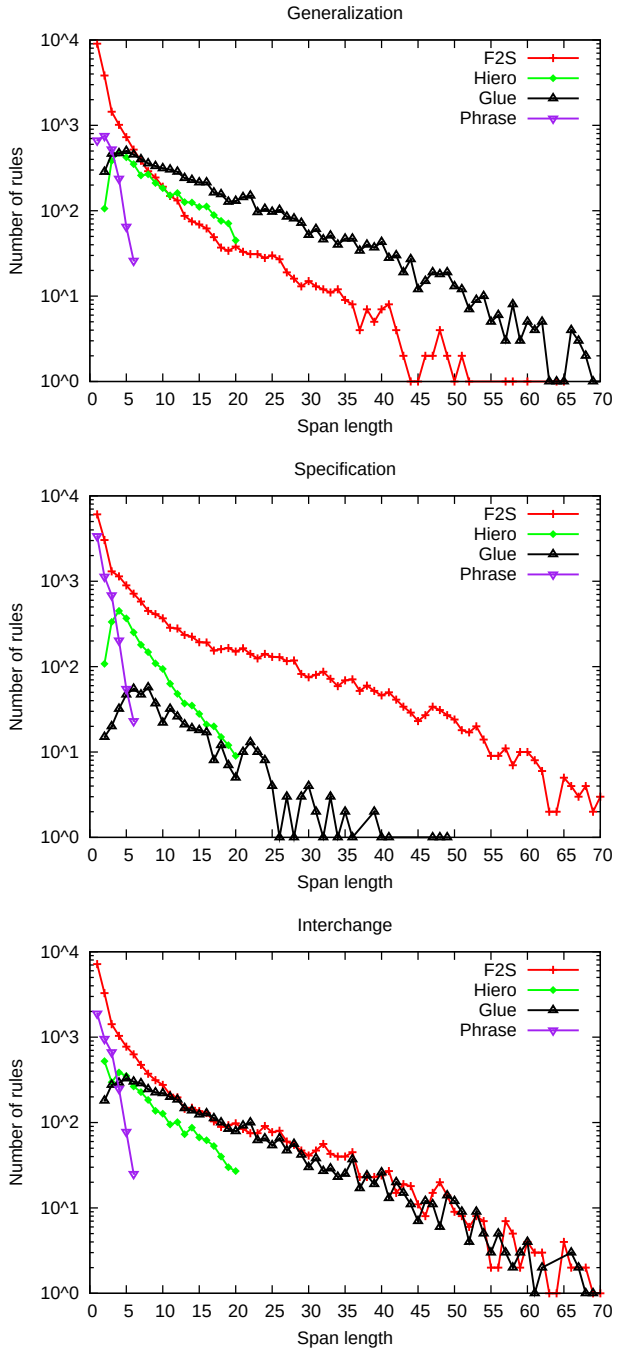


Figure 5: Rule distributions on GALE-web test set.

The third most expensive feature belongs to a part-of-speech tag—the preterminal VV. We may hypothesize that it shows the importance of lexical information for the precision of reordering typically carried out within (parent) VP nodes, and/or the importance of POS information for succesful disambiguation of word senses in translation. Ideally, the system can use a VP rule with a lexicalized VV. Less preferably, the VV part has to be translated by another T2S rule (losing the lexical constraint). In the worst case, the system has to use a hiero hypothesis to translate the VV part (losing the syntactic constraint), risking imprecise translation, since the hiero rule is not constrained to senses corresponding to the source POS VV. Again, the high penalty discourages from using the hiero derivation in this context.

On the other hand, the bottom of the table shows labels that encourage using hiero–DP, PP, DNP, ADJP, etc.–shorter phrases that tend to be monotone and less ambiguous.

Similar interpretations seem plausible when examining the **generalization** experiment. Expensive features related to preterminals (NR, NN, CD) may suggest two alternative principles: First, using F2S rules for thes POS categories and then switching to hiero is discouraged, since these contexts are more reliably handled by hiero due to better lexical coverage and common adjacency in nominal categories. Second, since there is only one attempt to switch from F2S derivation to hiero, letting F2S complete even larger spans (and maybe switching to hiero later) is favorable.

The tail of generalization feature weights is more difficult to interpret. The discount on VP encourages decoder to use F2S for entire verb phrases before switching to hiero, on the other hand, other verb-related preterminals occupy the tail as well, hurrying into early switching from F2S to hiero.

Finally, the feature weights tuned for the **interchange** experiment are divided into two subcolumns. Both generalization and specification weights show similar trends as in the previous two interaction schemes, although blurred (VP and IP descending from the absolute top). Since transitions in both ways are allowed, the search space is bigger and the system may behave differently. It is even possible for a path in the hypergraph to zigzag between F2S and hiero nodes to collect interaction discounts, "diluting" the syntactic homogeneity of the hypothesis.

Figure 5 and Tables 4 and 5 show rule distributions, total rule counts, and numbers of interactions of different types for the three interaction schemes on the GALE-web test set. The scope of phrase rules is limited to 6 words. The scope of hiero rules is limited to 20 words by the commonly used maxParse parameter, leaving longer spans to the glue rule.

The trends of F2S and glue rules show the most obvious difference. In the generalization, F2S rules translate spans of up to 50 words. Glue rules prevail on spans longer then 7 words. The specification is reversed, pushing the longest scope of hiero and glue rules down to 40 words, completing the longest sentences entirely with F2S. The interchange comes out as a mixture of the previous two trends.

All three schemes prefer using F2S rules at shorter spans, to the contrary of our original assumption of phrasal and hiero rules being stronger on local contexts *in general*. Here we may refer again to the specification feature weights for preterminals VV, NR, CC and P in Table 3 and to our previously stated hypothesis about the importance of preserving lexical and syntactic context.

Hiero rules usage on longer spans drops fastest for specification, slowest for generalization, and in between for interchange.

It is also interesting to notice the trends on very short spans (2–4 words) shown by rule distributions and reflected in numbers of interaction types. While specification often transitions from a single phrase rule directly into F2S, the interchange has relatively higher counts of hiero rules, another sign of the hiero and F2S interaction.

Synthesizing from several sources of indications is difficult, however, we arrive at the conclusion that joint decoding of hiero and F2S significantly improves the performance. While the single systems show similar performance, their roles are not balanced in joint decoding. It seems that the role of hiero consists in enabling F2S in most contexts.

We have focused on three special cases of interaction. We see a great potential in further studies of other schemes, allowing more flexible interaction than simple specification, but still more constrained than the interchange. It seems also promising to refine the interaction modeling with features taking into account more information than a single syntactic label, and to explore additional ways of parameter estimation.

7 Conclusion

We have proposed *flexible interaction of hypergraphs* as a novel technique combining hiero and forest-to-string translation models within one decoder. We have explored three basic interaction schemes—*specification*, *generalization*, and *interchange*—and described soft constraints controlling the interactions. We have carried out experiments on large training data and with strong baselines. Of the three schemes, the specification shows the highest gains, achieving improvements from 0.5 to 0.9 (T -B)/2 points over the best single system. We have conducted a detailed analysis of each system output based on different indications of interactions, discussed possible interpretations of results, and finally offered our conclusion and proposed future lines of research.

Acknowledgments

We thank Jiří Havelka for proofreading and helpful suggestions. We would like to acknowledge the support of DARPA under Grant HR0011-12-C-0015 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the DARPA.

References

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

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, Michigan, June.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

John Denero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *In Proceedings NAACL-HLT*, pages 975–983.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*, pages 273–280.

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

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-HMM-based hypothesis alignment for combining outputs from machine translation systems. In *Proceedings of EMNLP*, pages 98–107, October.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, pages 66–73.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 127–133.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*, pages 609–616.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009. Joint decoding with multiple translation models. In *Proceedings of ACL-IJCNLP*, pages 576–584, August.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*, pages 206–214.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*, pages 192–199.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL*, pages 404–411.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007. Improved word-level system combination for machine translation. In *Proceedings of ACL*, pages 312–319, Prague, Czech Republic, June.

Taro Watanabe and Eiichiro Sumita. 2011. Machine translation system combination by confusion forest. In *Proceedings of ACL 2011*, pages 1249–1257.

Hui Zhang, Min Zhang, Haizhou Li, and Chew Lim Tan. 2009. Fast translation rule matching for syntax-based statistical machine translation. In *Proceedings of EMNLP*, pages 1037–1045, Singapore, August.

Factored Soft Source Syntactic Constraints for Hierarchical Machine Translation

Zhongqiang Huang
Raytheon BBN Technologies
50 Moulton St
Cambridge, MA, USA
zhuang@bbn.com

Jacob Devlin
Raytheon BBN Technologies
50 Moulton St
Cambridge, MA, USA
jdevlin@bbn.com

Rabih Zbib
Raytheon BBN Technologies
50 Moulton St
Cambridge, MA, USA
rzbib@bbn.com

Abstract

This paper describes a factored approach to incorporating soft source syntactic constraints into a hierarchical phrase-based translation system. In contrast to traditional approaches that directly introduce syntactic constraints to translation rules by explicitly decorating them with syntactic annotations, which often exacerbate the data sparsity problem and cause other problems, our approach keeps translation rules intact and factorizes the use of syntactic constraints through two separate models: 1) a syntax mismatch model that associates each nonterminal of a translation rule with a distribution of tags that is used to measure the degree of syntactic compatibility of the translation rule on source spans; 2) a syntax-based reordering model that predicts whether a pair of sibling constituents in the constituent parse tree of the source sentence should be reordered or not when translated to the target language. The features produced by both models are used as soft constraints to guide the translation process. Experiments on Chinese-English translation show that the proposed approach significantly improves a strong string-to-dependency translation system on multiple evaluation sets.

1 Introduction

Hierarchical phrase-based translation models (Chiang, 2007) are widely used in machine translation systems due to their ability to achieve local fluency through phrasal translation and handle non-local phrase reordering using synchronous context-free grammars. A large number of previous works have tried to introduce grammaticality to the translation process by incorporating syntactic constraints into hierarchical translation models. Despite some differences in the granularity of syntax units (e.g., tree fragments (Galley et al., 2004; Liu et al., 2006), treebank tags (Shen et al., 2008; Chiang, 2010), and extended tags (Zollmann and Venugopal, 2006)), most previous work incorporates syntax into hierarchical translation models by explicitly decorating translation rules with syntactic annotations. These approaches inevitably exacerbate the data sparsity problem and cause other problems such as increased grammar size, worsened derivational ambiguity, and unavoidable parsing errors (Hanneman and Lavie, 2013).

In this paper, we propose a factored approach that incorporates soft source syntactic constraints into a hierarchical string-to-dependency translation model (Shen et al., 2008). The general ideas are applicable to other hierarchical models as well. Instead of enriching translation rules with explicit syntactic annotations, we keep the original translation rules intact, and factorize the use of source syntactic constraints through two separate models.

The first is a *syntax mismatch model* that introduces source syntax into the nonterminals of translation rules, and measures the degree of syntactic compatibility between a translation rule and the source spans it is applied to during decoding. When a hierarchical translation rule is extracted from a parallel training sentence pair, we determine a tag for each nonterminal based on the dependency parse of the source sentence. Instead of fragmenting rule statistics by directly labeling nonterminals with tags,

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 556–566,
Seattle, Washington, USA, 18-21 October 2013.

we keep the original string-to-dependency translation rules intact and associate each nonterminal with a distribution of tags. That distribution is then used to measure the syntactic compatibility between the syntactic context from which the translation rule is extracted and the syntactic analysis of a test sentence.

The second is a *syntax-based reordering model* that takes advantage of phrasal cohesion in translation (Fox, 2002). The reordering model takes a pair of sibling constituents in the source parse tree as input, and uses source syntactic clues to predict the ordering distribution (straight vs. inverted) of their translations on the target side. The resulting ordering distribution is used in the decoder at the word pair level to guide the translation process. This separate reordering model allows us to utilize source syntax to improve reordering in hierarchical translation models without having to explicitly annotate translation rules with source syntax.

Our results show that both the syntax mismatch model and the syntax-based reordering model are able to achieve significant gains over a strong Chinese-English MT baseline. The rest of the paper is organized as follows. Section 2 discusses related work in the literature. Section 3 provides an overview of our baseline string-to-dependency translation system. Section 4 describes the details of the syntax mismatch and syntax-based reordering models. Experimental results are presented in Section 5. The last section concludes the paper.

2 Related Work

Attempts to use rich syntactic annotations do not always result in improved performance when compared to purely hierarchical models that do not use linguistic guidance. For example, as shown in (Mi and Huang, 2008), tree-to-string translation models (Huang et al., 2006) only start to outperform purely hierarchical models when significant efforts were made to alleviate parsing errors by using forest-based approaches in both rule extraction and decoding. Using only syntactic phrases is too restrictive in phrasal translation as many useful phrase pairs are not syntactic constituents (Koehn et al., 2003). The syntax-augmented translation model of Zollmann and Venugopal (2006) annotates nonterminals in hierarchical rules with thousands of extended syntactic categories in order to capture the syntactic variations of phrase pairs. This results in exacerbated data sparsity problems, partially due to the requirement of exact matches in nonterminal substitutions between translation rules in the derivation. Several solutions were proposed. Shen et al. (2009) and Chiang (2010) used soft match features to explicitly model the substitution of nonterminals with different labels; Venugopal et al. (2009) used a preference grammar to soften the syntactic constraints through the use of a preference distribution of syntactic categories; and recently Hanneman and Lavie (2013) proposed a clustering approach to reduce the number of syntactic categories. Our proposed syntax mismatch model associates nonterminals with a distribution of tags. It is similar to the preference grammar in (Venugopal et al., 2009); however, we use treebank tags and focus on the syntactic compatibility between translation rules and the source sentence. The work of Huang et al. (2010) is most similar to ours, with the main difference being that their syntactic categories are latent and learned automatically in a data driven fashion while we simply use treebank tags based on dependency parsing. Marton and Resnik (2008) also exploited soft source syntax constraints without modifying translation rules. However, they focused on the quality of translation spans based on the syntactic analysis of the source sentence, while our method explicitly models the syntactic compatibility between translation rules and source spans.

Most research on reordering in machine translation focuses on phrase-based translation models as they are inherently weak at non-local reordering. Previous efforts to improve reordering for phrase-based systems can be largely classified into two categories. Approaches in the first category try to reorder words in the source sentence in a preprocessing step to reduce reordering in both word alignment and MT decoding. The reordering decisions are either made using manual or automatically learned rules (Collins et al., 2005; Xia and McCord, 2004; Xia and McCord, 2004; Genzel, 2010) based on the syntactic analysis of the source sentence, or constructed through an optimization procedure that uses feature-based reordering models trained on a word-aligned parallel corpus (Tromble and Eisner, 2009;

Khapra et al., 2013). Approaches in the second category try to explicitly model phrase reordering in the translation process. These approaches range from simple distance based distortion models (Koehn et al., 2003) that globally penalizes reordering based on the distorted distance, to lexicalized reordering models (Koehn et al., 2005; Al-Onaizan and Papineni, 2006) that assign reordering preferences of adjacent phrases for individual phrases, and to hierarchical reordering models (Galley and Manning, 2008; Cherry, 2013) that handle reordering preferences beyond adjacent phrases. Although hierarchical translation models are capable of handling non-local reordering, their accuracy is far from perfect. Xu et al. (2009) showed that the syntax-augmented hierarchical model (Zollmann and Venugopal, 2006) also benefits from reordering source words in a pre-processing step. Explicitly adding syntax to translation rules helps with reordering in general, but it introduces additional complexities, and is still limited by the context-free nature of hierarchical rules. Our work exploits an alternative direction that uses an external reordering model to improve word reordering of hierarchical models. Gao et al. (2011), Xiong et al. (2012), and Li et al. (2013) also studied external reordering models for hierarchical models. However, they focused on specific word pairs such as a word and its dependents or a predicate and its arguments, while our proposed general framework considers all word pairs in a sentence. Our syntax-based reordering model exploits phrasal cohesion in translation (Fox, 2002) by modeling the reordering of sibling constituents in the source parse tree, which is similar to the recent work of Yang et al. (2012). However, the latter focuses on finding the optimal reordering of sibling constituents before MT decoding, while our proposed model generates reordering features that are used together with other MT features to determine the optimal reordering during MT decoding.

3 String-to-Dependency Translation

Our baseline translation system is based on a string-to-dependency translation model similar to the implementation in (Shen et al., 2008). It is an extension of the hierarchical translation model of Chiang et al. (2006) that requires the target side of a phrase pair to have a well-formed dependency structure, defined as either of the two types:

- *fixed structure*: a single rooted dependency sub-tree with each child being a complete constituent. In this case, the phrase has a unique head word inside the phrase, i.e., the root of the dependency sub-tree. Each dependent of the head word, together with all of its descendants, is either completely inside the phrase or completely outside the phrase. For example, the phrase *give him* in Figure 1 (a) has a fixed dependency structure with head word *give*.

- *floating structure*: a sequence of siblings with each being a complete constituent. In this case, the phrase is composed of a sequence of sibling constituents whose common parent is outside the phrase. For example, the phrase *him that brown coat* in Figure 1 is a floating structure whose common parent *give* is not in the phrase.

Requiring the target side to have a well-formed dependency structure is less restrictive than requiring it to be a syntactic constituent, allowing more translation rules to be extracted. However, it still results in fewer rules than pure hierarchical translation models and might hurt MT performance. The well-formed dependency structure on the target side makes it possible to introduce syntax features during decoding. Shen et al. (2008) obtained significant improvements from including a dependency language model score in decoding, outweighing the negative effect of the dependency constraint. Shen et al. (2009) proposed an approach to label each nonterminal, which can be either on the left-hand-side (LHS) or the right-hand-side (RHS) of the rule, with the head POS tag of the underlying target phrase if it has a fixed dependency structure[1], and measure the mismatches between nonterminal labels when a RHS nonterminal of a rule is substantiated with the LHS nonterminal of another rule during decoding. This also resulted in further improvements in MT performance. Figure 1 (c) shows an example string-to-dependency translation rule in our baseline system.

[1]Nonterminals corresponding to floating structures keep their default label "X" as experiments show that it is not beneficial to label them differently.

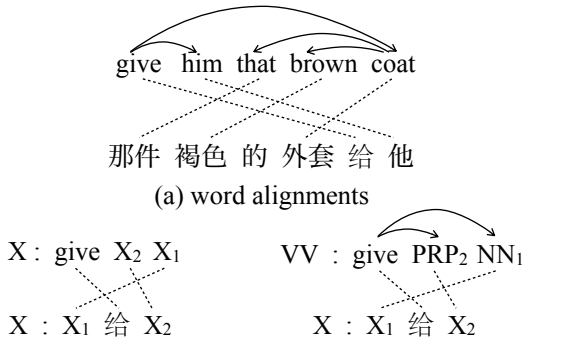


Figure 1: An example of extracting a string-to-dependency translation rule from word alignments. The nonterminals on the target side of the hierarchical rule (b) all correspond to fixed dependency structures and so they are labeled by the respective head tag in the string-to-dependency rule (c).

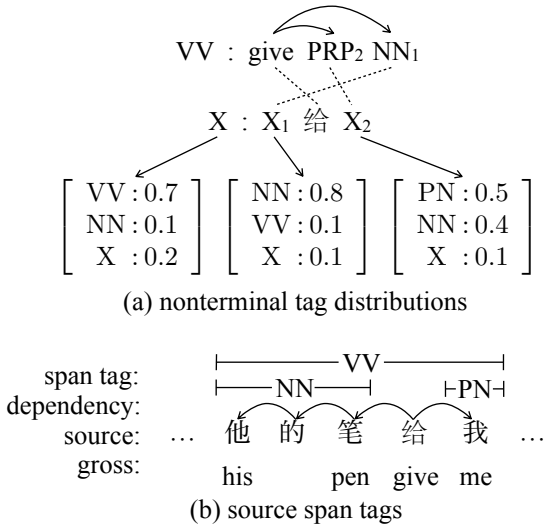


Figure 2: Example distribution of tags for nonterminals on the source side (a) and example tags for source spans (b)

4 Factored Syntactic Constraints

Although the string-to-dependency formulation helps to improve the grammaticality of translations, it lacks the ability to incorporate source syntax into the translation process. We next describe a factored approach to address this problem by utilizing source syntax through two models: one that introduces syntactic awareness to translation rules themselves, and another that focuses on reordering based on the syntactic analysis of the source.

4.1 Syntax Mismatch Model

A straightforward method to introduce awareness of source syntax to translation rules is to apply the same well-formed dependency constraint and head POS annotation on the target side of string-to-dependency translation rules to the source side. However, as discussed earlier, this would significantly reduce the number of rules that can be extracted, exacerbate data sparsity, and cause other problems, especially given that the target side is already constrained by the dependency requirement.

A relaxed method is to bypass the dependency constraint and only annotate source nonterminals whose underlying phrase is a fixed dependency structure with the head POS tag of the phrase. This method would still extract all of the rules that can be extracted from the baseline string-to-dependency translation model, but the extra annotation on nonterminals can split a rule into multiple rules, with the only difference being the nonterminal labels on the source side. Unfortunately, our experiments have shown that even this moderate annotation results in significantly lower translation quality due to the fragmentation of translation rules, and the increased derivational ambiguity. We have also tried to include some source tag mismatch features (with details described later) to measure the syntactic compatibility between the nonterminal labels of a translation rule and the corresponding tags of source spans. This improves translation accuracy, but not enough to compensate for the performance drop caused by annotating source nonterminals.

Our proposed method introduces syntax to translation rules without sacrificing performance. Instead of imposing dependency constraints or explicitly annotating source nonterminals, we keep the original string-to-dependency translation rules intact and associate each nonterminal on the source side with a distribution of tags. The tags are determined based on the dependency structure of training samples. If the underlying source phrase of a nonterminal is a fixed dependency structure in a training sample, we use the head POS tag of the phrase as the tag. Otherwise, we use the default tag "X" to denote float-

Feature	Condition	Value
f_1	t_s = X	P(t_r = X)
f_2	t_s = X	P($t_r \neq$ X)
f_3	$t_s \neq$ X	P(t_r = X)
f_4	$t_s \neq$ X	P($t_r = t_s$)
f_5	$t_s \neq$ X	P($t_r \neq$ X, $t_r \neq t_s$)

Table 1: Source tag mismatch features. The default value of each feature is zero if the source span tag t_s does not match the condition

ing structures and dependency structures that are not well formed. As a result, we still extract the same set of rules as in the baseline string-to-dependency translation model, and also obtain a distribution of tags for each nonterminal. Figure 2 (a) illustrates the example tag distributions of a string-to-dependency translation rule. The tag distributions provide an approximation of the source syntax of the training data from which the translation rules are extracted. They are used to measure the syntactic compatibility between a translation rule and the source spans it is applied to. At decoding time, we parse the source sentence and assign each span a tag in the same way as it is done during rule extraction, as shown in the example in Figure 2 (b). When a translation rule is used to expand a derivation, for each nonterminal (which can be on the LHS or RHS) on the source side of the rule, five source tag mismatch features are computed based on the distribution of tags P(t_r) on the rule nonterminal, and the tag t_s on the corresponding source span. The features are defined in Table 1. We use soft features instead of hard syntactic constraints, and allow the tuning process to choose the appropriate weight for each feature. As shown in Section 5, these source syntax mismatch features help to improve the baseline system.

4.2 Syntax-based Reordering Model

Most previous research on reordering models has focused on improving word reordering for statistical phrase-based translation systems (e.g., (Collins et al., 2005; Al-Onaizan and Papineni, 2006; Tromble and Eisner, 2009)). There has been less work on improving the reordering of hierarchical phrase-based translation systems (see (Xu et al., 2009; Gao et al., 2011; Xiong et al., 2012) for a few exceptions), except through explicit syntactic annotation of translation rules. It is generally assumed that hierarchical models are inherently capable of handling both local and non-local reorderings. However, many hierarchical translation rules are noisy and have limited context, and so may not be able to produce translations in the right order.

We propose a general framework that incorporates external reordering information into the decoding process of hierarchical translation models. To simplify the presentation, we make the assumption that every source word translates to one or more target words, and that the translations for a pair of source words is either straight or inverted. We discuss the general case later. Given a sentence $w_1, \cdots, w_n$, suppose we have a separate reordering model that predicts $\mathrm{P}_{\text{order}}(o_{ij})$, the probability distribution of ordering $o_{ij} \in \{\text{straight}, \text{inverted}\}$ between the translations of any source word pair (w_i, w_j). We can measure the goodness of a given hypothesis h with respect to the ordering predicted by the reordering model as the sum of log probabilities[2] for ordering each pair of source words, as defined in Equation 1:

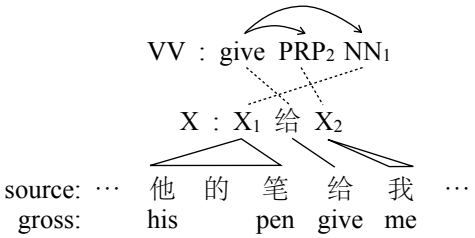
$$f_{\text{order}}(h) = \sum_{1 \leq i < j \leq n} \log \mathrm{P}_{\text{order}}(o_{ij} = o_{ij}^h) \qquad (1)$$

where o_{ij}^h is the ordering between the translations of source word pair (w_i, w_j) in hypothesis h. The reordering score $f_{\text{order}}(h)$ can be computed efficiently through recursion during hierarchical decoding as follows:

- Base case: for phrasal (i.e. non-hierarchical) rules, the ordering of translations for any word pair covered by the source phrase can be determined based on the word alignment of the rule. The value of the reordering score can be simply computed according to Equation 1.
- Recursive case: when a hierarchical rule is used to expand a partial derivation, two types of word pairs are encountered: a) word pairs that are covered exclusively by one of the nonterminals on the RHS of the rule, and b) other

[2] In practice, the log probability is thresholded to avoid negative infinity, which would otherwise result in a hard constraint.

(a)



(b)

word pair	translation order
(他, 给)	inverted
(他, 我)	inverted
(的, 给)	inverted
(的, 我)	inverted
(笔, 给)	inverted
(笔, 我)	inverted
(给, 我)	straight
(他, 的)	previously considered
(他, 笔)	previously considered
(的, 笔)	previously considered

Figure 3: An example rule application (a) with the translation order of new source word pairs covered by the rule shown in (b). The translation order of word pairs covered by X_1 is previously considered and is thus not shown.

word pairs. The reordering scores of the former would be already computed in previous rule applications, and can simply be retrieved from the partial derivation. Word pairs of the latter case are new word pairs introduced by the hierarchical rule, and their ordering can be determined based on the alignment of the hierarchical rule. The value of the reordering score of the new derivation is the sum of the reordering scores retrieved from the partial derivations for the nonterminals and the reordering scores of the new word pairs.

Figure 3 shows an example of determining the ordering of translations when applying a string-to-dependency rule. The alignment in the translation rule is able to fully determine the translation order for all new word pairs introduced by the rule. For example, "笔/pen" is covered by X_1 in the rule and the translation order for X_1 and "给/give" is inverted on the target side. Since "笔/pen" is translated together with other words covered by X_1 as a group, we can determine that the translation order between the source word pair "笔/pen" and "给/give" is also inverted on the target side. The words "他/his", "的", "笔 /pen" are all covered by the same nonterminal X_1 and thus their pairwise reordering scores have already been considered in previous rule applications.

Reordering features
The syntactic production rule
The syntactic labels of the nodes in the context
The head POS tags of the nodes in the context
The dep. labels of the nodes in the context
The seq. of dep. labels connecting the two nodes
The length of the nodes in the context

Table 2: Features in the reordering model

In practice, not all source words in a translation rule are translated to a target word; sometimes there is no clear ordering between the translations of two source words. In such cases we use a binary discount feature instead of the reordering feature.

This reordering framework relies on an external model to provide the ordering probability distribution of source word pairs. In this paper, we investigate a simple maximum-entropy reordering model based on the syntactic parse tree of the source sentence. This allows us to take advantage of the source syntax to improve reordering without using syntactic annotations in translation rules. The syntax-based reordering model attempts to predict the reordering probability of a pair of sibling constituents in the source parse tree, building on the fact that syntactic phrases tend to move in a group during translation (Fox, 2002). The reordering model is trained on a word-aligned corpus. For each pair of sibling constituents in the source parse tree, we determine the translation order on the target side based on word alignments. If there is a clear ordering[3], i.e., either straight or inverted, on the target side, we include the context of the constituent pair and its translation order as a sample for training or evaluating the maximum entropy reordering model. Table 2 lists the features of the reordering model.

The ordering distributions of source word pairs are determined based on the ordering distributions of sibling constituent pairs. For each pair of sib-

[3]If the translations overlap with other, the non-overlapping parts are used to determine the translation order.

ling constituents[4] in the parse tree of a source sentence, we compute its distribution of translation order using the reordering model. The distribution is shared among all word pairs covered by the respective constituents, which guarantees that the ordering distribution of any source word pair is computed exactly once. The ordering distributions of source word pairs are then used through the general reordering framework in the decoder to guide the decoding process.

5 Experiments

5.1 Experimental Setup

Our main experiments use the Chinese-English parallel training data and development sets released by the LDC, and made available to the DARPA GALE and BOLT programs. We train the translation model on 100 million words of parallel data. We use a 8 billion words of English monolingual data to train two language models: a trigram language model used in chart decoding, and a 5-gram language model used in n-best rescoring. The systems are tuned and evaluated on a mixture of newswire and web forum text from the development sets available for the DARPA GALE and BOLT programs, with up to 4 independent references for each source sentence. We also evaluate our final systems on both newswire and web text from the NIST MT06 and MT08 evaluations using an experimental setup compatible with the NIST MT12 Chinese-English constrained track. In this setup, the translation and language models are trained on 35 million words of parallel data and 3.8 billion words of English monolingual data, respectively. The systems are tuned on the MT02-05 development sets. All systems are tuned and evaluated on IBM BLEU (Papineni et al., 2002). The baseline string-to-dependency translation system uses more than 10 core features and a large number of sparse binary features similar to the method described in (Chiang et al., 2009). It achieves translation accuracies comparable to the top ranked systems in the NIST MT12 evaluation.

GIZA++ (Och and Ney, 2003) is used for automatic word alignment in all of the experiments. We use Charniak's parser (Charniak and Johnson, 2005) on the English side to obtain string-to-dependency translation rules, and use a latent variable PCFG parser (Huang and Harper, 2009) to parse the source side of the parallel training data as well as the test sentences for extracting syntax mismatch and reordering features. For both languages, dependency structures are read off constituency trees using manual head word percolation rules. We use a lexicon-based longest-match-first word segmenter to tokenize source Chinese sentences. Since the source tokenization used in our MT system is different from the treebank tokenization used to train the Chinese parser, the source sentences are first tokenized using the treebank-trained Stanford Chinese segmenter (Tseng et al., 2005), then parsed with the Chinese parser, and finally projected to MT tokenization based on the character alignment between the tokens. The syntax-based reordering model is trained on a set of Chinese-English manual word alignment corpora released by the LDC[5].

5.2 Syntax Mismatch Model

We first conduct experiments on the GALE/BOLT data sets to evaluate different strategies of incorporating source syntax into string-to-dependency translation rules. As mentioned in Section 4.1, constraining the source side of translation rules to only well-formed dependency structures is too restrictive given that our baseline system already has dependency constraint on the target side. We evaluate the relaxed method that only annotates source nonterminals with the head POS tag of the underlying phrase if the phrase is a fixed dependency structure. As shown in Table 3, nonterminal annotation results in a big drop in performance, decreasing the BLEU score of the baseline from 27.82 to 25.54. This suggests that it is undesirable to further fragment the translation rules. Introducing the syntax mismatch features described in Section 4.1 helps to improve

[4]Note that the constituent pairs used to train the reordering model are filtered to only contain these with clear ordering on the target side, while no such pre-filtering is applied to constituent pairs when applying the reordering model in translation. We leave it to future work to address this mismatch problem.

[5]The alignment corpora are LDC2012E24, LDC2012E72, LDC2012E95, and LDC2013E02. The reordering model can also be trained on automatically aligned data; however, our experiments show that using manual alignments results in a better accuracy for the reordering model itself and more improvements for the MT system.

	BLEU
baseline	27.82
+ tag annotation only	25.54
+ tag annotation, mismatch feat.	25.90
+ tag distribution, mismatch feat.	28.23

Table 3: Effects of tag annotation, tag distribution, and syntax mismatch features on MT performance on the GALE/BOLT data set.

BLEU from 25.54 to 25.90. This improvement is not large enough to compensate for the performance drop caused by annotating the nonterminals.

Our proposed approach, on the other hand, does not modify the translation rules in the baseline system, but only associates each nonterminal with a distribution of tags. For that reason, it does not suffer from the aforementioned problem. It achieves exactly the same performance as the baseline system if no source syntactic constraints are imposed during decoding. When the source syntax mismatch features are used, the proposed approach is able to achieve a gain of 0.41 in BLEU over the baseline system. Table 4 lists the learned weights of the syntax mismatch features after MT tuning. The negative weights of f_1 and f_2 mean that the MT system penalizes source spans that do not have a fixed dependency structure, and it assigns a higher penalty to rules whose nonterminals have a high probability of being extracted from source phrases that do not have a fixed dependency structure. When the source span has a fixed dependency structure, the MT system prefers translation rules that have a high probability of matching the tag on the source span (feature f_4) over the ones that do not match (features f_3 and f_5). This result is consistent with our expectations of the syntax mismatch features.

Feature	Description	Weight
f_1	t_s = X, t_r = X	−1.543
f_2	t_s = X, $t_r \neq$ X	−0.676
f_3	$t_s \neq$ X, t_r = X	0.380
f_4	$t_s \neq$ X, $t_r \neq$ X, $t_r = t_s$	1.677
f_5	$t_s \neq$ X, $t_r \neq$ X, $t_r \neq t_s$	0.232

Table 4: Learned syntax mismatch feature weight

5.3 Syntax-based Reordering Model

Before evaluating the syntax-based reordering model, we would like to establish the upper bound improvement that could be achieved using the general reordering framework for hierarchical translation models. Towards that goal, we conduct an oracle experiment on the GALE/BOLT development set that uses the oracle translation order from the reference as the external reordering model. For each source sentence in the development set, we pair it with the first reference translation (out of up to 4 independent translations). We then add the sentence pairs from the development set to the parallel training data and run GIZA++ to obtain word alignments. We consider the GIZA++ word alignments for the development set to be all correct, and use it to determine the oracle order in the reference translation. For the ordering distribution, we set the log probability of the reference translation order to 0 and the reverse order to -1 to avoid negative infinity. As shown in Table 5, the system tuned and evaluated with the oracle reordering model significantly outperforms the baseline by a large margin of 2.32 in BLEU on the GALE/BOLT test set. This suggests that there is room for potential improvement by using a fairly trained reordering model.

	BLEU
baseline	27.82
+ oracle reorder	30.14
+ syntax reorder	28.40

Table 5: Effects of external reordering features on MT performance on the GALE/BOLT test set.

We next evaluate the syntax-based reordering model. We train the model on manually aligned Chinese-English corpora. Since the tokenization used in the manual alignment corpora is different from the tokenization used in our MT system, the manual alignment is projected to the MT tokenization based on the character alignment between the tokens. Some extraneously tagged alignment links in the manual alignment corpora are not useful for machine translation and are thus removed before projecting the alignment. As described in Section 4.2, the syntax-based reordering method mod-

els the translation order of sibling constituent pairs in the source parse tree. As a result of strong phrasal cohesion (Fox, 2002), we find that 94% of constituent pairs have a clear ordering on the target side. We only retain these constituent pairs for training and evaluating the reordering model. In order to evaluate the accuracy of the maximum entropy reordering model, we divide the manual alignment corpora into 2/3 for training and 1/3 for evaluation. A baseline that only chooses the majority order (i.e. straight) has an accuracy of 69%, while the syntax-based reordering model improves the accuracy to 79%.

The final reordering model used in MT is trained on all of the samples extracted from the manual alignment corpora. As shown in Table 5, the syntax-based reordering feature improves the baseline by 0.58 in BLEU, which is a good improvement given our strong baseline. Table 6 lists the number of shifting errors in TER measurement (Snover et al., 2006) of various systems on the GALE/BOLT test set. The syntax-based reordering model achieves a 6.1% reduction in the number of shifting errors in the baseline system, and its combination with the syntax mismatch model achieves an additional reduction of 0.6%. This suggests that the proposed method helps to improve word reordering in translation.

	Shifting errors
baseline	3205
+ syntax mismatch	3089
+ syntax reorder	3010
+ syntax mismatch and reorder	2990

Table 6: Number of shifting errors in TER measurement of multiple systems on the GALE/BOLT test set

5.4 Final Results

Table 7 shows the final results on the GALE/BOLT test set, as well as the NIST MT06 and MT08 test sets. Both the syntax mismatch and the syntax-based reordering features improve the baseline system, resulting in moderate to significant gains in all of the five test sets. The two features are complementary to each other and their combination results in better improvement in four out of the five test sets compared to adding them separately. In three out of the five test sets, the improvement from the combination of the two features is statistically significant at the 95% confidence level over the baseline, with the largest absolute improvement of 1.43 in BLEU obtained on MT08 web.

6 Conclusion

In this paper, We have discussed problems resulting from explicitly decorating translation rules with syntactic annotations. We presented a factored approach to incorporate soft source syntax mismatch and reordering constraints to hierarchical machine translation, and showed how our models avoid the pitfalls of the explicit decoration approach. Experiments on Chinese-English translation show that the proposed approach significantly improves a strong string-to-dependency translation baseline on multiple evaluation sets. There are many directions in which this work can be continued. The syntax mismatch model can be extended to dynamically adjust the translation distribution based on the syntactic compatibility between a translation rule and a source sentence. It also might be beneficial to look beyond syntactic constituent pairs when modeling reordering, given that phrasal cohesion does not always hold in translation. The general framework that uses an external reordering model in hierarchical models via features can also be naturally extended to use multiple reordering models.

Acknowledgments

This work was supported in part by DARPA/IPTO Contract No. HR0011-12-C-0014 under the BOLT Program. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The authors would like to thank the anonymous reviewers for their helpful comments.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

	MT06 news	MT06 web	MT08 news	MT08 web	GALE/BOLT
baseline	43.76	36.13	40.52	27.78	27.82
+ syntax mismatch	43.89	36.72	40.82$^\frown$	28.54$^\approx$	28.23$^\approx$
+ syntax reorder	44.01	36.40	41.23$^{\approx\approx}$	28.95$^{\approx\approx}$	28.40$^{\approx\approx}$
+ syntax mismatch and reorder	44.28$^\frown$	36.43$^\frown$	41.14$^\approx$	29.21$^{\approx\approx}$	28.62$^{\approx\approx}$
improvement over baseline	+0.52	+0.30	+0.62	+1.43	+0.8

Table 7: Results on Chinese-English MT. The symbols $\frown$, $\approx$, and $\approx\approx$ indicate that the system is better than the baseline at the 85%, 95%, and 99% confidence levels, respectively, as defined in (Koehn, 2004).

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. 2006. Parsing Arabic dialects. In *Conference of the European Chapter of the Association for Computational Linguistics*.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Heidi Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2004. What's in a translation rule. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Yang Gao, Philipp Koehn, and Alexandra Birch. 2011. Soft dependency constraints for reordering in hierarchical phrase-based translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the International Conference on Computational Linguistics*.

Greg Hanneman and Alon Lavie. 2013. Improving syntax-augmented machine translation by coarsening the label set. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

Zhongqiang Huang, Martin Čmejrek, and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mitesh M. Khapra, Ananthakrishnan Ramanathan, and Karthik Visweswariah. 2013. Improving reordering performance using higher order and structural features. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 iwslt

speech translation evaluation. In *International Workshop on Spoken Language Translation*.

Philipp Koehn. 2004. Pharaoh: A bean search decoder for phrase-based statistical machine translation models. In *Proceedings of the Conference of Association for Machine Translation in the Americas*.

Junhui Li, Philip Resnik, and Hal Daume. 2013. Modeling syntactic and semantic structures in hierarchical phrase-based translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computional Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Proceedings of the SIGHAN Workshop on Chinese Language Processing*.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2009. Preference grammars: softening syntactic constraints to improve statistical machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of the International Conference on Computational Linguistics*.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceeding of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Nan Yang, Mu Li, Dongdong Zhang, and Nenghai Yu. 2012. A ranking-based approach to word reordering for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*.

Recursive Autoencoders for ITG-based Translation

Peng Li, Yang Liu and **Maosong Sun**
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
pengli09@gmail.com, {liuyang2011,sms}@tsinghua.edu.cn

Abstract

While inversion transduction grammar (ITG) is well suited for modeling ordering shifts between languages, how to make applying the two reordering rules (i.e., straight and inverted) dependent on actual blocks being merged remains a challenge. Unlike previous work that only uses boundary words, we propose to use recursive autoencoders to make full use of the entire merging blocks alternatively. The recursive autoencoders are capable of generating vector space representations for variable-sized phrases, which enable predicting orders to exploit syntactic and semantic information from a neural language modeling's perspective. Experiments on the NIST 2008 dataset show that our system significantly improves over the MaxEnt classifier by 1.07 BLEU points.

1 Introduction

Phrase-based models (Koehn et al., 2003; Och and Ney, 2004) have been widely used in practical machine translation (MT) systems due to their effectiveness, simplicity, and applicability. First, as sequences of consecutive words, phrases are capable of memorizing local word selection and reordering, making them an effective mechanism for translating idioms or translations with word insertions or omissions. Moreover, n-gram language models can be seamlessly integrated into phrase-based decoders since partial translations grow left to right in decoding. Finally, phrase-based systems can be applicable to most domains and languages, especially for resource-scarce languages without high-accuracy parsers.

However, as phrase-based decoding casts translation as a string concatenation problem and permits arbitrary permutations, it proves to be NP-complete (Knight, 1999). Therefore, phrase reordering modeling has attracted intensive attention in the past decade (e.g., Och et al., 2004; Tillman, 2004; Zens et al., 2004; Al-Onaizan and Papineni, 2006; Xiong et al., 2006; Koehn et al., 2007; Galley and Manning, 2008; Feng et al., 2010; Green et al., 2010; Bisazza and Federico, 2012; Cherry, 2013).

Among them, reordering models based on **inversion transduction grammar (ITG)** (Wu, 1997) are one of the important ongoing research directions. As a formalism for bilingual modeling of sentence pairs, ITG is particularly well suited to predicting ordering shifts between languages. As a result, a number of authors have incorporated ITG into left-to-right decoding to constrain the reordering space and reported significant improvements (e.g., Zens et al., 2004; Feng et al., 2010). Along another line, Xiong et al. (2006) propose a maximum entropy (MaxEnt) reordering model based on ITG. They use the CKY algorithm to recursively merge two blocks (i.e., a pair of source and target strings) into larger blocks, either in a straight or an inverted order. Unlike lexicalized reordering models (Tillman, 2004; Koehn et al., 2007; Galley and Manning, 2008) that are defined on individual bilingual phrases, the MaxEnt ITG reordering model is a two-category classifier (i.e., straight or inverted) for two arbitrary bilingual phrases of which the source phrases are adjacent. This potentially alleviates the data sparseness

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 567–577,
Seattle, Washington, USA, 18-21 October 2013.

problem since there are usually a large number of reordering training examples available (Xiong et al., 2006). As a result, the MaxEnt ITG model and its extensions (Xiong et al., 2008; Xiong et al., 2010) have achieved competing performance as compared with state-of-the-art phrase-based systems.

Despite these successful efforts, the ITG reordering classifiers still face a major challenge: how to extract features from training examples (i.e., a pair of bilingual strings). It is hard to decide which words are representative for predicting reordering, either manually or automatically, especially for long sentences. As a result, Xiong et al. (2006) only use boundary words (i.e., the first and the last words in a string) to predict the ordering. What if we look inside? Is it possible to avoid manual feature engineering and learn semantic representations from the data?

Fortunately, the rapid development of intersecting deep learning with natural language processing (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Glorot et al., 2011; Bordes et al., 2011; Socher et al., 2011a; Socher et al., 2011b; Socher et al., 2011c; Socher et al., 2012; Bordes et al., 2012; Huang et al., 2012; Socher et al., 2013; Hermann and Blunsom, 2013) brings hope for alleviating this problem. In these efforts, natural language words are represented as real-valued vectors, which can be naturally fed to neural networks as input. More importantly, it is possible to learn vector space representations for multi-word phrases using **recursive autoencoders** (Socher et al., 2011c), which opens the door to leveraging semantic representations of phrases in reordering models from a neural language modeling point of view.

In this work, we propose an ITG reordering classifier based on recursive autoencoders. The neural network consists of four autoencoders (i.e., the first source phrase, the first target phrase, the second source phrase, and the second target phrase) and a softmax layer. The recursive autoencoders, which are trained on reordering examples extracted from word-aligned bilingual corpus, are capable of producing vector space representations for arbitrary multi-word strings in decoding. Therefore, our model takes the whole phrases rather than only boundary words into consideration when predicting phrase permutations. Experiments on the NIST 2008 dataset show that our system significantly improves over the MaxEnt classifier by 1.07 in terms of case-insensitive BLEU score.

2 Recursive Autoencoders for ITG-based Translation

2.1 Inversion Transduction Grammar

Inversion transduction grammar (ITG) (Wu, 1997) is a formalism for synchronous parsing of bilingual sentence pairs. Xiong et al. (2006) apply bracketing transduction grammar (BTG), which is a simplified version of ITG, to phrase-based translation using the following production rules:

$$X \rightarrow [X^1, X^2] \quad (1)$$

$$X \rightarrow \langle X^1, X^2 \rangle \quad (2)$$

$$X \rightarrow f/e \quad (3)$$

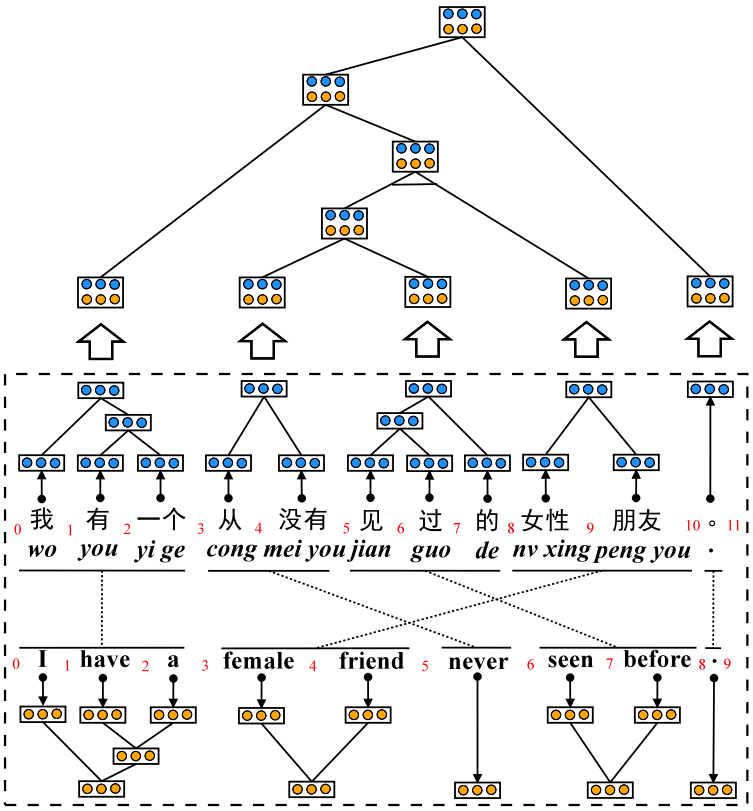
where X is a **block** that consists of a pair of source and target strings, f is a source phrase, and e is a target phrase. X^1 and X^2 are two neighboring blocks of which the two source phrases are adjacent. While rule (1) merges two target phrases in a **straight** order, rule (2) merges in an **inverted** order. Besides these two **reordering rules**, rule (3) is a **lexical rule** that translates a source phrase f into a target phrase e. This is exactly a bilingual phrase used in conventional phrase-based systems.

An ITG derivation, which consists of a sequence of production rules, explains how a sentence pair is generated simultaneously. Figure 1 shows an ITG derivation for a Chinese sentence and its English translation. We distinguish between two types of blocks:

1. **atomic blocks**: blocks generated by applying lexical rules,
2. **composed blocks**: blocks generated by applying reordering rules.

In Figure 1, the sentence pair is segmented into five atomic blocks:

$X_{0,3,0,3}$: *wo you yi ge* $\leftrightarrow$ I have a
$X_{3,5,5,6}$: *cong mei you* $\leftrightarrow$ never
$X_{5,8,6,8}$: *jian guo de* $\leftrightarrow$ seen before
$X_{8,10,3,5}$: *nv xing peng you* $\leftrightarrow$ female friend
$X_{10,11,8,9}$: . $\leftrightarrow$.



(1)	$X_{0,11,0,9} \rightarrow [X_{0,10,0,8}, X_{10,11,8,9}]$
(2)	$X_{0,10,0,8} \rightarrow [X_{0,3,0,3}, X_{3,10,3,8}]$
(3)	$X_{0,3,0,3} \rightarrow$ *wo you yi ge* / I have a
(4)	$X_{3,10,3,8} \rightarrow \langle X_{3,8,5,8}, X_{8,10,3,5} \rangle$
(5)	$X_{3,8,5,8} \rightarrow [X_{3,5,5,6}, X_{5,8,6,8}]$
(6)	$X_{3,5,5,6} \rightarrow$ *cong mei you* / never
(7)	$X_{5,8,6,8} \rightarrow$ *juan guo de* / seen before
(8)	$X_{8,10,3,5} \rightarrow$ *nv xing peng you*/ female friend
(9)	$X_{10,11,8,9} \rightarrow$. / .

Figure 1: An ITG derivation for a Chinese sentence and its translation. We use $X_{i,j,k,l} = \langle f_i^j, e_k^l \rangle$ to represent a block. Our neural ITG reordering model first assigns vector space representations to single words and then produces vectors for phrases using recursive autoencoders, which form atomic blocks. The atomic blocks are recursively merged into composed blocks, the vector space representations of which are produced by recursive autoencoders simultaneously. The neural classifier makes decisions at each node using the vectors of all its descendants.

where $X_{3,5,5,6}$ indicates that the block consists of a source phrase spanning from position 3 to position 5 (i.e., *"cong mei you"*) and a target phrase spanning from position 5 to position 6 (i.e., "never"). More formally, a block $X_{i,j,k,l} = \langle f_i^j, e_k^l \rangle$ is a pair of a source phrase $f_i^j = f_{i+1} \dots f_j$ and a target phrase $e_k^l = e_{k+1} \dots e_l$. Obviously, these atomic blocks are generated by lexical rules.

Two blocks of which the source phrases are adjacent can be merged into a larger one in two ways: concatenating the target phrases in a straight order using rule (1) or in an inverted order using rule (2). For example, atomic blocks $X_{3,5,5,6}$ and $X_{5,8,6,8}$ are merged into a composed block $X_{3,8,5,8}$ in a straight order, which is further merged with an atomic block $X_{8,10,3,5}$ into another composed block $X_{3,10,3,8}$ in an inverted order. This process recursively proceeds until the entire sentence pair is generated.

The major challenge of applying ITG to machine translation is to decide when to merge two blocks in a straight order and when in an inverted order. Therefore, the ITG reordering model can be seen as a two-category classifier $P(o|X^1, X^2)$, where $o \in \{straight, inverted\}$.

A naive way is to assign fixed probabilities to two reordering rules, which is referred to as *flat model* by Xiong et al. (2006):

$$P(o|X^1, X^2) = \begin{cases} p & o = straight \\ 1-p & o = inverted \end{cases} \quad (4)$$

The drawback of the flat model is ignoring the actual blocks being merged. Intuitively, different blocks should have different preferences between the two orders.

To alleviate this problem, Xiong et al. (2006) propose a maximum entropy (MaxEnt) classifier:

$$P(o|X^1, X^2) = \frac{\exp(\theta \cdot h(o, X^1, X^2))}{\sum_{o'} \exp(\theta \cdot h(o', X^1, X^2))} \quad (5)$$

where $h(\cdot)$ is a vector of features defined on the blocks and the order, θ is a vector of feature weights.

While MaxEnt is a flexible and powerful framework for including arbitrary features, feature engineering becomes a major challenge for the MaxEnt classifier. Xiong et al. (2006) find that boundary words (i.e., the first and the last words in a string) are informative for predicting reordering. Actually, it is hard to decide which internal words in a long composed blocks are representative and informative. Therefore, they only use boundary words as the main features.

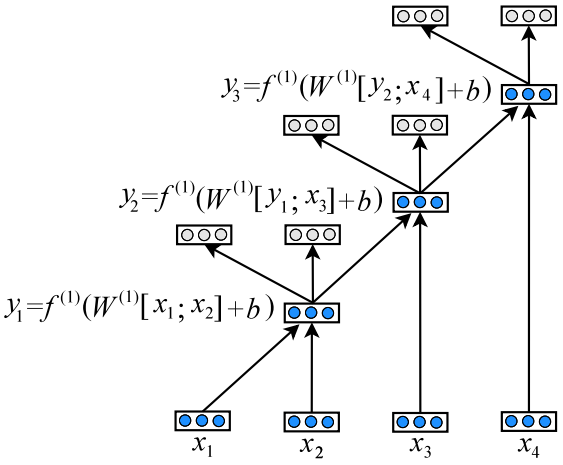


Figure 2: A recursive autoencoder for multi-word strings. The example is adapted from (Socher et al., 2011c). Blue and grey nodes are original and reconstructed ones, respectively.

However, it seems not enough to just consider boundary words and ignore all internal words when making order predictions, especially for long sentences.[1] Indeed, Xiong et al. (2008) find that the MaxEnt classifier with boundary words as features is prone to make wrong predictions for long composed blocks. As a result, they have to impose a hard constraint to always prefer merging long composed blocks in a monotonic way.

Therefore, it is important to consider more than boundary words to make more accurate reordering predictions. We need a new mechanism to achieve this goal.

2.2 Recursive Autoencoders

2.2.1 Vector Space Representations for Words

In neural networks, a natural language word is represented as a real-valued vector (Bengio et al., 2003; Collobert and Weston, 2008). For example, we can use $[0.1\ 0.8\ 0.4]^T$ to represent "female" and

[1] Strictly speaking, the ITG reordering model is not a phrase reordering model since phrase pairs are only the atomic blocks. Instead, it is defined to work on arbitrarily long strings because composed blocks become larger and larger until the entire sentence pair is generated.

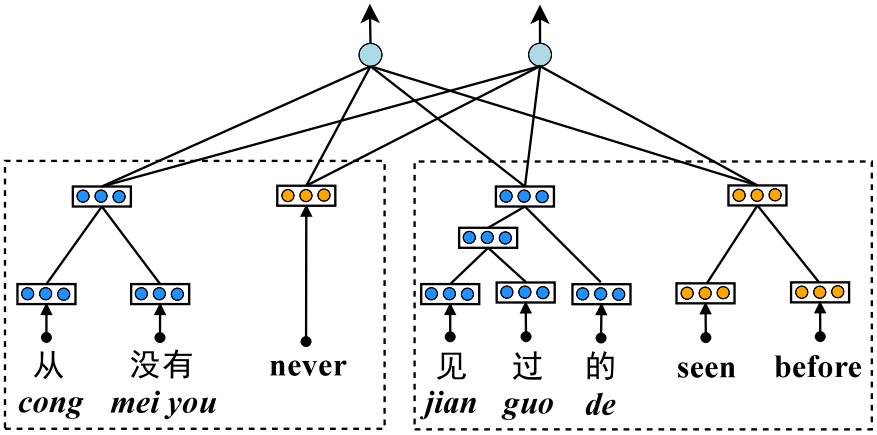


Figure 3: A neural ITG reordering model. The binary classifier makes decisions based on the vector space representations of the source and target sides of merging blocks.

$[0.7\ 0.1\ 0.5]^T$ to represent "friend". Such vector space representations enable natural language words to be fed to neural networks as input.

Formally, we denote each word as a vector $x \in \mathbb{R}^n$. These word vectors are then stacked into a word embedding matrix $L \in \mathbb{R}^{n \times |V|}$, where $|V|$ is the vocabulary size. Given a sentence that is an ordered list of m words, each word has an associated vocabulary index k into the word embedding matrix L that we use to retrieve the word's vector space representation. This look-up operation can be seen as a simple projection layer:

$$x_i = Lb_k \in \mathbb{R}^n \tag{6}$$

where b_k is a binary vector which is zero in all positions except for the kth index.

In Figure 1, we assume $n = 3$ for simplicity and can retrieve vectors for Chinese and English words from two embedding matrices, respectively.

2.2.2 Vector Space Representations for Multi-Word Strings

To apply neural networks to ITG-based translation, it is important to generate vector space representations for atomic and composed blocks.

For example, since the vector of "female" is $[0.1\ \ 0.8\ \ 0.4]^T$ and the vector of "friend" is $[0.7\ 0.1\ 0.5]^T$, what is the vector of the phrase "female friend"? If we denote "female friend" as p (i.e., parent), "female" as c_1 (i.e., the first child), and "friend" as c_2 (i.e., the second child), this can be done by applying a function $f^{(1)}$:

$$p = f^{(1)}(W^{(1)}[c_1; c_2] + b^{(1)}) \tag{7}$$

where $[c_1; c_2] \in \mathbb{R}^{2n \times 1}$ is the concatenation of c_1 and c_2, $W^{(1)} \in \mathbb{R}^{n \times 2n}$ is a parameter matrix, $b^{(1)} \in \mathbb{R}^{n \times 1}$ is a bias term, and $f^{(1)}$ is an element-wise activation function such as $\tanh(\cdot)$, which is used in our experiments.

Note that the resulting vector for the parent is also an n-dimensional vector, e.g, $[0.6\ 0.9\ 0.2]^T$. The same neural network can be recursively applied to two strings until the vector of the entire sentence is generated. As ITG derivation builds a binary parse tree, the neural network can be naturally integrated into CKY parsing.

To assess how well the learned vector p represents its children, we can **reconstruct** the children in a reconstruction layer:

$$[c_1'; c_2'] = f^{(2)}(W^{(2)}p + b^{(2)}) \tag{8}$$

where c_1' and c_2' are the reconstructed children, $W^{(2)}$ is a parameter matrix for reconstruction, $b^{(2)}$ is a bias term for reconstruction, and $f^{(2)}$ is an element-wise activation function, which is also set as $\tanh(\cdot)$ in our experiments. Similarly, the same reconstruction neural network can be applied to each node in an ITG parse.

These neural networks are called **recursive autoencoders** (Socher et al., 2011c). Figure 2 illustrates an application of a recursive autoencoder to a

binary tree. The blue and grey nodes are the original and reconstructed nodes, respectively. The autoencoder is re-used at each node of the tree. The binary tree is composed of a set of triplets in the form of $(p \rightarrow c_1\ c_2)$, where p is a parent vector and c_1 and c_2 are children vectors of p. Each child can be either an input word vector or a multi-word vector. Therefore, the tree in Figure 2 can be represented as three triplets: $(y_1 \rightarrow x_1\ x_2)$, $(y_2 \rightarrow y_1\ x_3)$, and $(y_3 \rightarrow y_2\ x_4)$.

In Figure 1, we use recursive autoencoders to generate vector space representations for Chinese and English phrases, which form the atomic blocks for further block merging.

2.2.3 A Neural ITG Reordering Model

Once the vectors for blocks are generated, it is straightforward to introduce a neural ITG reordering model. As shown in Figure 3, the neural network consists of an input layer and a softmax layer. The input layer is composed of the vectors of the first source phrase, the first target phrase, the second source phrase, and the second target phrase. Note that all phrases in the same language use the the same recursive autoencoder. The softmax layer outputs the probabilities of the two merging orders:

$$P(o|X^1, X^2) = \frac{\exp(g(o, X^1, X^2))}{\sum_{o'} \exp(g(o', X^1, X^2))} \quad (9)$$

$$g(o, X^1, X^2) = f(W^o c(X^1, X^2) + b^o) \quad (10)$$

where $o \in \{straight, inverted\}$, $W^o \in \mathbb{R}^{1\times 4n}$ is a parameter matrix, $b^o \in \mathbb{R}$ is a bias term, and $c(X^1, X^2) \in \mathbb{R}^{4n\times 1}$ is the concatenation of the vectors of the four phrases.

3 Training

There are three sets of parameters in our recursive autoencoders:

1. θ_L: word embedding matrix L for both source and target languages (Section 2.2.1);
2. θ_{rec}: recursive autoencoder parameter matrices $W^{(1)}$, $W^{(2)}$ and bias terms $b^{(1)}$, $b^{(2)}$ for both source and target languages (Section 2.2.2);
3. θ_{reo}: neural ITG reordering model parameter matrix W^o and bias term b^o (Section 2.2.3).

All these parameters are learned automatically from the training data. For clarity, we will use θ to denote all these parameters in the rest of the paper.

For training word embedding matrix, there are two settings commonly used. In the first setting, the word embedding matrix is initialized randomly. This works well in a supervised scenario, in which a neural network updates the matrix in order to optimize some task-specific objectives (Collobert et al., 2011; Socher et al., 2011c). In the second setting, the word embedding matrix is pre-trained using an unsupervised neural language model (Bengio et al., 2003; Collobert and Weston, 2008) with huge amount of unlabeled data. In this work, we prefer to the first setting because the word embedding matrices can be trained to minimize errors with respect to reordering modeling.

There are two kinds of errors involved

1. **reconstruction error**: how well the learned vector space representations represent the corresponding strings?
2. **reordering error**: how well the classifier predicts the merging order?

As described in Section 2.2.2, the input vector c_1 and c_2 of a recursive autoencoder can be reconstructed using Eq. 8 as c_1' and c_2'. We use Euclidean distance between the input and the reconstructed vectors to measure the *reconstruction error*:

$$E_{rec}([c_1; c_2]; \theta) = \frac{1}{2} \left\| [c_1; c_2] - [c_1'; c_2'] \right\|^2. \quad (11)$$

Given a sentence, there are exponentially many ways to obtain its vector space representation. Note that each way corresponds to a binary tree like Figure 2. To find a binary tree with minimal reconstruction error, we follow Socher et al. (2011c) to use a greedy algorithm. Taking Figure 2 as an example, the greedy algorithm begins with computing the reconstruction error $E_{rec}(\cdot)$ for each pair of consecutive vectors, i.e., $E_{rec}([x_1; x_2]; \theta)$, $E_{rec}([x_2; x_3]; \theta)$ and $E_{rec}([x_3; x_4]; \theta)$. Suppose $E_{rec}([x_1; x_2]; \theta)$ is the smallest, the algorithm will replace x_1 and x_2 with their vector representation y_1 produced by the recursive autoencoder. Then, the algorithm evaluates $E_{rec}([y_1; x_3]; \theta)$ and $E_{rec}([x_3; x_4]; \theta)$ and repeats the above replacing steps until only one vector

remains. Socher et al. (2011c) find that the greedy algorithm runs fast without significant loss in performance as compared with CKY-style algorithms.

Given a training example set $S = \{t_i = (o_i, X_i^1, X_i^2)\}$, the average reconstruction error on the source side on the training set is defined as

$$E_{rec,s}(S;\theta) = \frac{1}{N_s}\sum_i \sum_{p\in T_R^\theta(t_i,s)} E_{rec}([p.c_1, p.c_2];\theta) \quad (12)$$

where $T_R^\theta(t_i, s)$ denotes all the intermediate nodes on the source side in binary trees, N_s is the number of these intermediate nodes, and $p.c_k$ is the kth child vector of p. The average reconstruction error on the target side, denoted by $E_{rec,t}(S;\theta)$, can be computed in a similar way.

Therefore, the reconstruction error is defined as

$$E_{rec}(S;\theta) = E_{rec,s}(S;\theta) + E_{rec,t}(S;\theta). \quad (13)$$

Given a training example $t_i = (o_i, X_i^1, X_i^2)$, we assume the probability distribution d_{t_i} for its label is $[1, 0]$ when $o_i = straight$, and $[0, 1]$ when $o_i = inverted$. Then the cross-entropy error is

$$E_c(t_i;\theta) = -\sum_o d_{t_i}(o)\log\left(P_\theta(o|X^1, X^2)\right) \quad (14)$$

where $o \in \{straight, inverted\}$. As a result, the reordering error is defined as

$$E_{reo}(S;\theta) = \frac{1}{|S|}\sum_i E_c(t_i;\theta). \quad (15)$$

Therefore, the joint training objective function is

$$J = \alpha E_{rec}(S;\theta) + (1-\alpha)E_{reo}(S;\theta) + R(\theta) \quad (16)$$

where α is a parameter used to balance the preference between reconstruction error and reordering error, $R(\theta)$ is the regularizer and defined as [2]

$$R(\theta) = \frac{\lambda_L}{2}\|\theta_L\|^2 + \frac{\lambda_{rec}}{2}\|\theta_{rec}\|^2 + \frac{\lambda_{reo}}{2}\|\theta_{reo}\|^2. \quad (17)$$

As Socher et al. (2011c) stated, a naive way for lowering the reconstruction error is to make the magnitude of the hidden layer very small, which is not desirable. In order to prevent such behavior, we normalize all the output vectors of the hidden layers to have length 1 in the same way as (Socher et al., 2011c). Namely we set $p = \frac{p}{||p||}$ after computing p as in Eq. 7, and $c_1' = \frac{c_1'}{||c_1'||}$, $c_2' = \frac{c_2'}{||c_2'||}$ in Eq. 8.

Following Socher et al. (2011c), we use L-BFGS to estimate the parameters with respect to the joint training objective. Given a set of parameters, we construct binary trees for all the phrases using the greedy algorithm. The derivatives for these fixed binary trees can be computed via backpropagation through structures (Goller and Kuchler, 1996).

4 Experiments

4.1 Data Preparation

We evaluated our system on Chinese-English translation. The training corpus contains 1.23M sentence pairs with 32.1M Chinese words and 35.4M English words. We used SRILM (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 398.6M words. We used the NIST 2006 MT Chinese-English dataset as the development set and NIST 2008 dataset as the test set. The evaluation metric is case-insensitive BLEU. Because of the expensive computational cost for training our neural ITG reordering model, only the reordering examples extracted from about 1/5 of the entire parallel training corpus were used to train our neural ITG reordering model.

For the neural ITG reordering model, we set the dimension of the word embedding vectors to 25 empirically, which is a trade-off between computational cost and expressive power. We use the early stopping principle to determine when to stop L-BFGS. The hyper-parameters α, λ_L, λ_{rec} and λ_{reo} are optimized by random search (Bergstra and Bengio, 2012). As preliminary experiments show that classification accuracy has a high correlation with BLEU score, we optimize these hyper-parameters with respect to classification accuracy instead of BLEU to reduce computational cost. We randomly select 400,000 reordering examples as training set, 500 as development set, and another 500 as test set. The numbers of straight and inverted reordering examples in the development/test set are set to be equal to avoid biases. We draw α uniformly from 0.05

[2] The bias terms $b^{(1)}$, $b^{(2)}$ and b^o are not regularized. We do not exclude them from the equation explicitly just for clarity.

System	NIST 2006 (tune)	NIST 2008
maxent	30.40	23.75
neural	**31.61***	**24.82***

Table 1: BLEU scores on the NIST 2006 and 2008 datasets. *: significantly better ($p < 0.01$). "maxent" denotes the baseline maximum entropy system and "neural" denotes our recursive autoencoder system.

length	$>$	$=$	$<$
$[1, 10]$	43	121	57
$[11, 20]$	181	67	164
$[21, 30]$	170	11	152
$[31, 40]$	105	3	90
$[41, 50]$	69	1	53
$[51, 119]$	40	0	30

Table 2: Number of sentences that our system has a higher ($>$), equal ($=$) or lower ($<$) sentence-level BLEU-4 score on the NIST 2008 dataset.

to 0.3, and λ_L, λ_{rec}, λ_{reo} exponentially from 10^{-8} to 10^{-2}. We use the following hyper-parameters in our experiments: $\alpha = 0.11764$, $\lambda_L = 7.59 \times 10^{-5}$, $\lambda_{rec} = 1.30 \times 10^{-5}$ and $\lambda_{reo} = 3.80 \times 10^{-4}$. [3]

The baseline system is a re-implementation of (Xiong et al., 2006). Our system is different from the baseline by replacing the MaxEnt reordering model with a neural model. Both the systems have the same pruning settings: the threshold pruning parameter is set to 0.5 and the histogram pruning parameter to 40. For minimum-error-rate training, both systems generate 200-best lists.

4.2 MT Evaluation

Table 1 shows the case-insensitive BLEU-4 scores of the baseline system and our system on the development and test sets. Our system outperforms the baseline system by 1.21 BLEU points on the development set and 1.07 on the test set. Both the differences are statistically significant at $p = 0.01$ level (Riezler and Maxwell, 2005).

Table 2 shows the number of sentences that our system has a higher ($>$), equal ($=$) or lower ($<$) BLEU score on the NIST 2008 dataset. We find that our system is superior to the baseline system for long sentences.

[3]The choice of α is very important for achieving high BLEU scores. We tried a number of intervals and found that the classification accuracy is most stable in the interval [0.100,0.125].

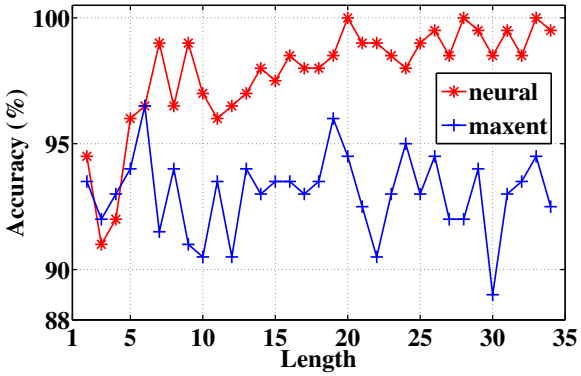


Figure 4: Comparison of reordering classification accuracies between the MaxEnt and neural classifiers over varying phrase lengths. "Length" denotes the sum of the lengths of two source phrases in a reordering example. Our classifier (neural) outperforms the MaxEnt classifier (maxent) consistently, especially for predicting long-distance reordering.

# of examples	NIST 2006 (tune)	NIST 2008
100,000	30.88	23.78
200,000	30.75	23.89
400,000	30.80	24.35
800,000	31.01	24.45
6,004,441	31.61	24.82

Table 3: The effect of reordering training data size on BLEU scores. The BLEU scores rise with the increase of training data size. Due to the computational cost, we only used 1/5 of the entire bilingual corpus to train our neural reordering model.

Figure 4 compares classification accuracies of the neural and MaxEnt classifiers. "Length" denotes the sum of the lengths of two source phrases in a reordering example. For each length, we randomly select 200 unseen reordering examples to calculate the classification accuracy. Our classifier outperforms the baseline consistently, especially for long composed blocks.

Xiong et al. (2008) find that the performance of the baseline system can be improved by forbidding inverted reordering if the phrase length exceeds a pre-defined distortion limit. This heuristic increases the BLEU score of the baseline system significantly to 24.46 but is still significantly worse ($p < 0.05$) than our system without the heuristic. We find that imposing this heuristic fails to improve our system

cluster 1	cluster 2	cluster 3	cluster 4	cluster 5
1.18	works for	alternative duties	these people who	of the three
accessibility	verify on	one-day conference	the reasons why	on the fundamental
wheelchair	tunnels from	armed groups	the story of how	over the entire
candies	transparency in	chinese language works	the system which	through its own
cough	opinion at	eating habits	the trend towards	with the best

Table 4: Words and phrases that are close in the Euclidean space. The words and phrases in the same cluster have similar behaviors from a reordering point of view rather than relatedness, suggesting that the vector representations produced by the recursive autoencoders are helpful for capturing reordering regularities.

significantly. One possible reason is that there is limited room for improvement as our system makes fewer wrong predictions for long composed blocks.

The above results suggest that our system does go beyond using boundary words and make a better use of the merging blocks by using vector space representations.

Table 3 shows the effect of training dataset size on BLEU scores. We find that BLEU scores on both the development and test sets rise with the increase of the training dataset size. As the training process is very time-consuming, only the reordering examples extracted from 1/5 of the entire parallel training corpus are used in our experiments to train our model. Obviously, with more efficient training algorithms, making full use of all the reordering examples extracted from the entire corpus will result in better results. We leave this for future work.

4.3 Qualitative Analysis on Vector Representations

Table 4 shows a number of words and phrases that are close (measured by Euclidean distance) in the n-dimensional space. We randomly select about 370K target side phrases used in our experiments and cluster them into 983 clusters using k-means algorithm (MacQueen, 1967). The distance between two phrases are measured by the Euclidean distance between their vector representations. As shown in Table 4, cluster 1 mainly consists of nouns, cluster 2 mainly contains verb/noun+preposition structures, cluster 3 contains compound phrases, cluster 4 consists of phrases which should be followed by a clause, and cluster 5 mainly contains the beginning parts of prepositional phrases that tend to be followed by a noun phrase or word. We find that the words and phrases in the same cluster have similar behaviors from a reordering point of view rather than relatedness. This indicates that the vector representations produced by the recursive autoencoders are helpful for capturing reordering regularities.

5 Conclusion

We have presented an ITG reordering classifier based on recursive autoencoders. As recursive autoencoders are capable of producing vector space representations for arbitrary multi-word strings in decoding, our neural ITG system achieves an absolute improvement of 1.07 BLEU points over the baseline on the NIST 2008 Chinese-English dataset.

There are a number of interesting directions we would like to pursue in the near future. First, replacing the MaxEnt classifier with a neural one redefines the conditions for risk-free hypothesis recombination. We find that the number of hypotheses that can be recombined reduces in our system. Therefore, we plan to use forest reranking (Huang, 2008) to alleviate this problem. Second, it is interesting to follow Socher et al. (2013) to combine linguistically-motivated labels with recursive neural networks. Another problem with our system is that the decoding speed is much slower than the baseline system because of the computational overhead introduced by RAEs. It is necessary to investigate more efficient decoding algorithms. Finally, it is possible to apply our method to other phrase-based and even syntax-based systems.

Acknowledgments

This research is supported by the 863 Program under the grant No. 2012AA011102, by the Boeing Tsinghua Joint Research Project on Language Processing (Agreement TBRC-008-SDB-2011 Phase 3

(2013)), by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, and by a Research Fund No. 20123000007 from Tsinghua MOE-Microsoft Joint Laboratory. Many thanks go to Chunyang Liu and Chong Kuang for their great help for setting up the computing platform. We also thank Min-Yen Kan, Meng Zhang and Yu Zhao for their insightful discussions.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 529–536, Sydney, Australia, July.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, February.

Arianna Bisazza and Marcello Federico. 2012. Modified distortion matrices for phrase-based statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 478–487, Jeju Island, Korea, July.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Atlanta, Georgia, June.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Yang Feng, Haitao Mi, Yang Liu, and Qun Liu. 2010. An efficient shift-reduce decoding algorithm for phrased-based machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 285–293, Beijing, China, August.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by back-propagation through structure. In *Proceedings of 1996 IEEE International Conference on Neural Networks (Volume:1)*, volume 1, pages 347–352.

Spence Green, Michel Galley, and Christopher D. Manning. 2010. Improved models of distortion cost for statistical machine translation. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 867–875, Los Angeles, California, June.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria, August.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 586–594, Columbus, Ohio, June.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, December.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June.

James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1.

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

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA, May.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan, June.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of Advances in Neural Information Processing Systems 24*, pages 801–809.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011c. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing, vol. 2*, pages 901–904, September.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, USA, May.

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

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia, July.

Deyi Xiong, Min Zhang, Aiti Aw, Haitao Mi, Qun Liu, and Shouxun Lin. 2008. Refinements in BTG-based statistical machine translation. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 505–512.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2010. Linguistically annotated reordering: Evaluation and analysis. *Computational Linguistics*, 36(3):535–568, September.

Richard Zens, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 205–211, Geneva, Switzerland, Aug 23–Aug 27.

Automatically Classifying Edit Categories in Wikipedia Revisions

Johannes Daxenberger[†] **and Iryna Gurevych**[†‡]

† Ubiquitous Knowledge Processing Lab
Department of Computer Science, Technische Universität Darmstadt

‡ Information Center for Education
German Institute for Educational Research and Educational Information

http://www.ukp.tu-darmstadt.de

Abstract

In this paper, we analyze a novel set of features for the task of automatic edit category classification. Edit category classification assigns categories such as spelling error correction, paraphrase or vandalism to edits in a document. Our features are based on differences between two versions of a document including meta data, textual and language properties and markup. In a supervised machine learning experiment, we achieve a micro-averaged F1 score of .62 on a corpus of edits from the English Wikipedia. In this corpus, each edit has been multi-labeled according to a 21-category taxonomy. A model trained on the same data achieves state-of-the-art performance on the related task of fluency edit classification. We apply pattern mining to automatically labeled edits in the revision histories of different Wikipedia articles. Our results suggest that high-quality articles show a higher degree of homogeneity with respect to their collaboration patterns as compared to random articles.

1 Introduction

Due to its ever-evolving and collaboratively built content, Wikipedia has been the subject of many NLP studies. While the number of newly created articles in the online encyclopedia declined in the last few years (Suh et al., 2009), the number of edits in existing articles is rather stable.[1] It is reasonable to assume that the latter will not change in the near future. One of the major reasons for the popularity of Wikipedia is its up-to-dateness (Keegan et al., 2013), which in turn requires constant editing activity. Wikipedia's revision history stores all changes made to any page in the encyclopedia in separate revisions. Previous studies have exploited revision history data in tasks such as preposition error correction (Cahill et al., 2013), spelling error correction (Zesch, 2012) or paraphrasing (Max and Wisniewski, 2010). However, they all use different approaches to extract the information needed for their task. Ferschke et al. (2013) outline several applications benefiting from revision history data. They argue for a unified approach to extract and classify edits from revision histories based on a predefined edit category taxonomy.

In this work, we show how the extraction and automatic multi-label classification of any edit in Wikipedia can be handled with a single approach. Therefore, we use the 21-category edit classification taxonomy developed in previous work (Daxenberger and Gurevych, 2012). This taxonomy enables a fine-grained analysis of edit activity in revision histories. We present the results from an automatic classification experiment, based on an annotated corpus of edits in the English Wikipedia. Additional information necessary to reproduce our results, including word lists and training, development and test data, is released online.[2] To the best of our knowledge, this is the first approach allowing to classify each single edit in Wikipedia into one or more of 21 different edit categories using a supervised machine learning

[1] http://stats.wikimedia.org/EN/TablesDatabaseEdits.htm

[2] http://www.ukp.tu-darmstadt.de/data/edit-classification

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 578–589,
Seattle, Washington, USA, 18-21 October 2013.

approach.

We define our task as edit category classification. An *edit* is a coherent, local change which modifies a document and which can be related to certain meta data (e.g. its author, time stamp etc.). In edit category classification, we aim to detect all n edits $e^k_{v-1,v}$ with $0 \leq k < n$ in adjacent versions r_{v-1}, r_v of a document (we refer to the older revision as r_{v-1} and to the newer as r_v) and assign each of them to one or more edit categories. There exist at least two main applications of edit category classification: First, a fine-grained classification of edits in collaboratively created documents such as Wikipedia articles, scientific papers or research proposals, would help us to better understand the collaborative writing process. This includes answers to questions about the kind of contribution of individual authors (Who has added substantial contents?, Who has improved stylistic issues?) and about the kind of collaboration which characterizes different articles (Liu and Ram, 2011). Second, automatic classification of edits generates huge amounts of training data for the above mentioned NLP systems.

Edit category classification is related to the better known task of document pair classification. In *document pair classification*, a pair of documents has to be assigned to one or more categories (e.g. paraphrase/non-paraphrase, plagiarism/non-plagiarism). Here, the document may be a very short text, such as a sentence or a single word. Applications of document pair classification include plagiarism detection (Potthast et al., 2012), paraphrase detection (Madnani et al., 2012) or text similarity detection (Bär et al., 2012). In *edit category classification*, we also have two documents. However, these documents are different versions of the same text. This scenario implies certain characteristics for a well-designed feature set as we will demonstrate in this study.

The main contributions of this paper are: First, we introduce a novel feature set for edit category classification. Second, we evaluate the performance of this feature set on different tasks within a corpus of Wikipedia edits. We propose the new task of edit category classification and show that our model is able to classify edits from a 21-category taxonomy. Furthermore, our model achieves state-of-the-art performance in a fluency edit classification task (Bronner and Monz, 2012). Third, we analyze collaboration patterns based on edit categories on two subsets of Wikipedia articles, namely featured and non-featured articles. We detect correlations between collaboration patterns and high-quality articles. This is demonstrated by the fact that featured articles have a higher degree of homogeneity with respect to their collaboration patterns as compared to random articles.

The rest of this paper is structured as follows. In Section 2, we motivate our experiments based on previous work. Section 3 explains our training data and the features we use for the machine learning experiments. In Section 4, we present and discuss the results of our experiments. We also demonstrate an application of our classifier model in Section 5 by mining frequent collaboration patterns in the revision histories of different articles. Finally, we draw a conclusion in Section 6.

2 Related Work

Wikipedia is a huge data source for generating training data for edit category classification, as all previous versions of each page in the encyclopedia are stored in its revision history. Unsurprisingly, the number of studies extracting certain kinds of Wikipedia edits keeps growing. Most of these use manually defined rules or filters find the right kind of edits. Among the latter, there are NLP applications such as the detection of lexical errors (Nelken and Yamangil, 2008), spelling error correction (Max and Wisniewski, 2010; Zesch, 2012), preposition error correction (Cahill et al., 2013), sentence compression (Nelken and Yamangil, 2008; Yamangil and Nelken, 2008), summarization (Nelken and Yamangil, 2008), simplification (Yatskar et al., 2010; Woodsend and Lapata, 2011), paraphrasing (Max and Wisniewski, 2010; Dutrey et al., 2011), textual entailment (Zanzotto and Pennacchiotti, 2010; Cabrio et al., 2012), information retrieval (Aji et al., 2010; Nunes et al., 2011) and bias detection (Recasens et al., 2013).

Bronner and Monz (2012) define features for the supervised classification of factual and fluency edits. Their features are calculated both on character- and word-level. Furthermore, they use features based on POS tags, named entities, acronyms, and a lan-

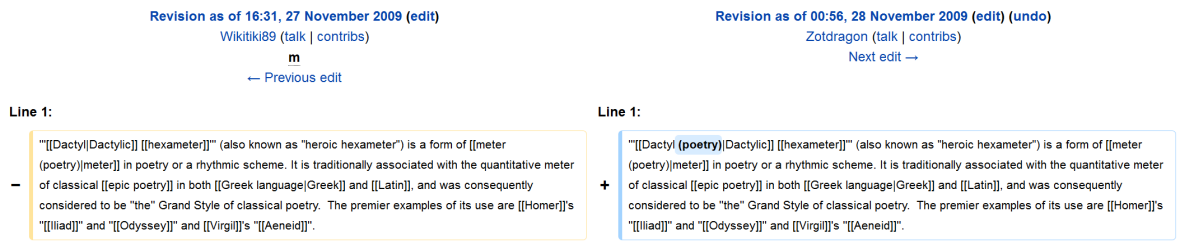


Figure 1: An example edit from WPEC labeled with REFERENCE-M, as displayed by Wikimedia's diff page tool.

guage model (word n-grams). In their experiments, character-level features and named entity features show the highest improvement over the baseline.

Vandalism detection in Wikipedia has mostly been defined as a binary machine learning task, where the goal is to classify a pair of adjacent revisions as vandalized or not-vandalized based on edit category features. In Adler et al. (2011), the authors group these features into meta data (author, comment and time stamp of a revision), reputation (author and article reputation), textual (language independent, i.e. token- and character-based) and language features (language dependent, mostly dictionary-based). They carry out cross-validation experiments on the PAN-WVC-10 corpus (Potthast and Holfeld, 2011). Classifiers based on reputation and text performed best. Adler et al. (2011) use Random Forests as classifier (Breiman, 2001) in their experiments. This classifier was also used in the vandalism detection study of Javanmardi et al. (2011) where it outperformed the classifiers based on Logistic Regression and Naive Bayes.

Different to the approach of Bronner and Monz (2012) and previous vandalism classification studies, we built a model which accounts for multi-labeling and a fine-grained edit category system. Our feature set builds upon existing work while adding a substantial number of new features.

3 Experiments

3.1 Wikipedia Edit Category Corpus

For our experiments, we used the freely available Wikipedia Edit Category Corpus (WPEC) compiled in previous work (Daxenberger and Gurevych, 2012). In this corpus, each pair of adjacent revisions is segmented into one or more edits. This enables an accurate picture of the editing process, as an author may perform several independent edits in the same revision. Furthermore, edits are multi-labeled, i.e. each edit is assigned one or more categories. This is important for a precise description of major edits, e.g. when an entire new paragraph including text, references and markup is added. There are four basic types of edits, namely Insertions, Deletions, Modifications and Relocations. These are calculated via a line-based diff comparison on the source text (including wiki markup). As previously suggested (Daxenberger and Gurevych, 2012), inside modified lines, only the span of text which has actually been changed is marked as edit (either Insertion, Deletion or Modification), not the entire line. We extracted the data which is not contained in WPEC (meta data and plain text of r_{v-1} and r_v) using the Java Wikipedia Library (JWPL) with the Revision Toolkit (Ferschke et al., 2011).

In Daxenberger and Gurevych (2012), we divide the 21-category taxonomy into text-base (meaning-changing edits), surface (non meaning-changing edits) and Wikipedia policy (VANDALISM and REVERT) edits. Among the text-base edits, we include categories for templates, references (internal and external links), files and information, each of which is further divided into an insertion (I), deletion (D) and modification (M) category. Surface edits consist of paraphrases, spelling and grammar corrections, relocations and markup edits. The latter category contains all edits which affect markup elements that are not covered by any of the other categories and is divided into insertions, deletions and modifications. This includes, for example, apostrophes in '''bold text'''. We also suggested an OTHER category, which is intended for edits which cannot be labeled due to segmentation errors. Figure 1 shows an example edit from WPEC, labeled with the REFERENCE-

	Feature	Value	Explanation
Meta Data	Author group	user	Wikimedia user group of the author
	Author is registered*	true	Author is registered (otherwise: IP user)
	Same author*	false	Authors of r_v and r_{v-1} are the same
	Comment length*	0	Number of characters in the comment
	Vulgarism in comment	false	Comment contains a word from in the vulgarism word list
	Comment is auto-generated	false	Entire comment has been auto-generated
	Auto-generated comment ratio	0	Auto-generated part of comment divided by length of the comment
	Incorrect comment ratio	0	Out-of-dictionary word count divided by word count in the comment
	Comment n-grams[1]	—	Presence or absence of token n-grams in the comment
	Is revert*	false	Comment contains a word from in the revert word list
	Is minor	false	Revision has been marked as minor change
	Time difference*	505	Time difference between r_{v-1} and r_v (in minutes)
	Number of edits	1	Absolute number of edits in the (r_{v-1}, r_v)-pair
Textual	Diff capitals*	0	Difference in the number of capitals
	Diff digits*	0	Difference in the number of digits
	Diff special characters*	2	Difference in the number of non-alphanumeric characters
	Diff whitespace characters	1	Difference in the number of whitespace characters
	Diff characters*	9	Difference in the number of characters
	Diff tokens*	1	Difference in the number of whitespace-separated tokens
	Diff repeated characters	0	Difference in the number of repeated characters
	Diff repeated tokens	0	Difference in the number of repeated white-space separated tokens
	Cosine similarity	0	Cosine similarity
	Levenshtein distance*	9	Levenshtein distance
	Optimal string alignment distance	9	Optimal string alignment distance (Damerau-Levenshtein distance)
	Ratio diff to paragraph characters	0.02	Diff characters divided by the length of the edited paragraph
	Ratio diff to revision characters	0.0005	Diff characters divided by the length of r_{v-1}
	Ratio diff to paragraph tokens	0.04	Diff tokens divided by the length of the edited paragraph
	Ratio diff to revision tokens	0.0003	Diff tokens divided by the length of r_{v-1}
	Ratio old to new paragraph	0	Difference in the number of characters in the edited paragraph
	Character n-grams[1]	p,o,e,t,r,y[2]	Presence or absence of n-grams of edited characters
	Token n-grams[1]	poetry[2]	Presence or absence of n-grams of edited tokens
	Simple edit type	Insertion	Modification, Insertion, Deletion or Relocation
Markup	Diff number m	0	Difference in the number of m
	Diff type m	false	Different types of m
	Diff type context m	true[3]	Different types of m within the immediate context of the edit
	Is covered by m	true[3]	Edit is covered by m in r_{v-1}
	Covers m	false	Edit covers m in r_{v-1}
Language	Diff spelling errors*	0	Difference in the number of out-of-dictionary words
	Diff vulgar words*	0	Difference in the number of tokens contained in vandalism word list
	Semantic similarity	-1	Explicit Semantic Analysis with vector indexes from Wiktionary
	Diff POS tags*	false	POS tag sets are symmetrically different
	Diff type POS tags*	0	Number of distinct POS tags

[1] N-gram features are represented as boolean features.
[2] In this example, n = 1 (unigrams).
[3] True if m corresponds to internal link, false otherwise.

Table 1: List of edit category classification features with explanations. The values correspond to the the example edit from Figure 1. m may refer to internal link, external link, image, template or markup element. Features marked with * have previously been mentioned in Adler et al. (2011), Javanmardi et al. (2011) or Bronner and Monz (2012).

M category. WPEC was created in a manual annotation study with three annotators. The overall inter-annotator agreement measured as Krippendorf's α is .67 (Daxenberger and Gurevych, 2012). The experiments in this study are based on the gold standard annotations in WPEC, which have been derived by means of a majority vote for each edit.

WPEC consists of 981 revision pairs, segmented into 1,995 edits. We define edit category classification as a multi-label classification task. For the sake of readability, in the following we will refer to an edit $e^k_{v-1,v}$ as e_i, with $e_i \in E$, where $0 \leq i < 1995$ and E is the set of all edits. An edit e_i is the basic classification unit in our task. Each e_i has to be labeled with a set of categories $y \subseteq C$, where C is the set of all edit categories, $|C| = 21$.

3.2 Features for Edit Category Classification

We grouped our features into *meta data*, *textual*, *markup* and *language* features. An overview and explanation of all features can be found in Table 1. The scheme we apply to group edit category classification features is similar to the system used by Adler et al. (2011). We re-use some of the features suggested by Adler et al. (2011), Javanmardi et al. (2011) and Bronner and Monz (2012), as marked in Table 1. Features are calculated on edited text spans. We label the edited text span corresponding to e_i in r_{v-1} as t_{v-1} and the edited text span in r_v as t_v. In edits which are insertions, we consider t_{v-1} to be empty, while t_v is considered empty for deletions. For Relocations, $t_{v-1} = t_v$.

Table 1 includes the value of each feature for the example edit from Figure 1. This edit modifies the link `[[Dactyl|Dactylic]]` by adding a specification to the target of that link. For spell-checking, we use British and US-American English Jazzy dictionaries.[3] Markup elements are detected by the Sweble Wikitext parser (Dohrn and Riehle, 2011).

Meta data features We consider the comment, author, time stamp or any other flag ("minor change") of r_v as meta data. The Wikimedia user group[4] of an author specifies the edit permissions of this user (e.g. bot, administrator, blocked user). We indicate whether the revision comments or parts of it have been auto-generated. This happens when a page is blanked, i.e. all of its content has been deleted or replaced or when a new page or redirect is created (denoted by the *Comment is auto-generated* feature). Furthermore, edits within a specific section of an article are automatically marked by adding a prefix with the name of this section to the comment of the revision (denoted by the *Auto-generated comment ratio* feature). Meta data features have the same value for all edits in a (r_{v-1}, r_v)-pair.

Textual features Textual features are calculated based on a certain property of the changed text. In a preprocessing step, any wiki markup inside t_{v-1} and t_v is deleted. As for the example edit from Figure 1, t_{v-1} would correspond to an empty string and t_v would be represented as " (poetry)". The n-gram feature spaces are composed of n-grams that are present either in t_{v-1} but not t_v, or vice verse. Character n-grams only contain English alphabet characters, token n-grams consist of words excluding special characters.

Markup features As opposed to textual features, wiki markup features account for the Wikimedia specific markup elements. Markup features are calculated based on the number and type of a markup element m and the surrounding context of an edit. Here, m can be a template, an external or internal link, an image or any other element used to describe markup including HTML tags. The type of m is defined by the link target for internal and external links and images, by the name of the template for templates and by the wiki markup element name for markup elements. Markup features are calculated on text spans t_{v-1} and t_v. Naturally, wiki markup is not deleted beforehand. The edited text spans t_{v-1} and t_v may be located inside a markup element m (e.g. a link or a template). In such cases, our diff algorithm will not label the entire element m, but rather the actually modified text. However, such an edit may change the name of a template or the target of a link (as in the example edit from Figure 1). We therefore include the immediate context s_{v-1} and s_v of each edit and compare the type of potential markup elements m in s_{v-1} and s_v. Here, s_v (s_{v-1}) is defined as t_v (t_{v-1}) including all preceding and follow-

[3] http://sourceforge.net/projects/jazzydicts

[4] http://meta.wikimedia.org/wiki/User_classes

	Revisions	Edits	Cardinality
Train	713	1,597	1.20
Test	89	229	1.24
Dev	89	169	1.21

Table 2: Statistics of the training, test and development set. Cardinality is the average number of edit categories assigned to an edit.

ing characters in r_v (r_{v-1}) which are not separated from t_v (t_{v-1}) by a boundary character (whitespace or line break). The above described features model *what* is actually edited in the text. A number of features are calculated on t_{v-1} only. These features are more likely to inform about *where* an edit is conducted. They specify whether t_{v-1} covers (i.e. contains) a certain wiki markup element and vice versa, i.e. whether t_{v-1} is located inside a text span that belongs to a markup element.

Language Language features are calculated on the context s_{v-1} and s_v of edits, any wiki markup is deleted. For the Explicit Semantic Analysis, we use Wiktionary (Zesch et al., 2008) and not Wikipedia assuming that the former has a better coverage with respect to different lexical classes. POS tagging was carried out using the OpenNLP POS tagger.[5] The vandalism word list contains a hand-crafted set of around 100 vandalism and spam words from various places in the web.

3.3 Experimental Setup

We extract features with the help of ClearTK (Ogren et al., 2008). For the machine learning part, we use Weka (Hall et al., 2009) with the Meka[6] and Mulan (Tsoumakas et al., 2010) extensions for multi-label classification. We use DKPro Lab (Eckart de Castilho et al., 2011) to test different parameter combinations. We randomly split the gold standard data from WPEC into 80% training, 10% test and 10% development set, as shown in Table 2.

Multi-label Classification We report the performance of various machine learning algorithms. A comprehensive overview of multi-label classification algorithms and evaluation measures can be found in Madjarov et al. (2012). Multi-label classification problems are solved by either transforming the multi-label classification task into one or more single-label classification tasks (problem transformation method) or by adapting single-label classification algorithms (algorithm adaption method). Several algorithms have been developed on top of the former methods and use ensembles of such classifiers (ensemble methods). We applied the Binary Relevance approach (BR), a simple transformation method which converts the multi-label problem into $|C|$ binary single-label problems, where $|C|$ is the number of categories. Hence, this method trains a classifier for each category in the corpus (one-against-all). It is the most straightforward approach when dealing with multi-labeled data. However, it does not consider possible relationships or dependencies between categories. Therefore, we tested two more sophisticated methods. Hierarchy of multi-label classifiers HOMER (Tsoumakas et al., 2008) is a problem transformation method. It accounts for possibly hierarchical relationships among categories by dividing the overall category set into a tree-like structure with nodes of small category sets of size k and leaves of single categories. Subsequently, a multi-label classifier is applied to each node in the tree. Random k-labelsets RAKEL (Tsoumakas et al., 2011) is an ensemble method, which randomly chooses l typically small subsets with k categories from the overall set of categories. Subsequently, all k-labelsets which are found in the multi-labeled data set are converted into new categories in a single-labeled data set using the la-

		Random	Majority	BR	HOMER	RAKEL
	Threshold	–	–	.10	.25	.33
Example	**Accuracy**	.09	.13	.50	.44	**.53**
	Exact Match	.06	.13	.35	.36	**.44**
	F1	.09	.13	.55	.47	**.56**
	Precision	.10	.13	.54	.46	**.56**
	Recall	.10	.13	**.61**	.50	.60
Label	**Macro-F1**	.10	.06	.49	.35	**.51**
	Micro-F1	.10	.12	.59	.49	**.62**
Ranking	**One Error**	.90	.87	.42	.48	**.34**

Table 3: Overall classification results with 3 multi-label classifiers and a C4.5 decision tree base classifier, as compared to random and majority category baselines.

[5] Maxent model for English, `http://opennlp.apache.org`

[6] `http://meka.sourceforge.net`

bel powerset transformation (Trohidis et al., 2008). HOMER and BR are among the multi-label classifiers, which Madjarov et al. (2012) recommend as benchmark methods. As underlying single-label classification algorithm, we used a C4.5 decision tree classifier (Quinlan, 1993), as decision tree classifiers yield state-of-the-art performance in the related work.

Multi-label Evaluation We denote the set of relevant categories for each edit $e_i \in E$ as $y_i \in C$ and the set of predicted categories as $h(e_i)$. Evaluation measures for multi-label classification systems are based on either bipartitions or rankings. Among the former, we report example-based (weighting each edit equally) and label-based (weighting each edit category equally) measures. The accuracy of a multi-label classifier is defined as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{|h(e_i)\cap y_i|}{|h(e_i)\cup y_i|}$, which corresponds to the Jaccard similarity of $h(e_i)$ and y_i averaged over all edits. We report subset accuracy (exact match), calculated as $\frac{1}{|E|}\sum_{i=1}^{|E|} I$, with $I = 1$ if $h(e_i) = y_i$ and $I = 0$ otherwise. Example-based precision is defined as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{|h(e_i)\cap y_i|}{|h(e_i)|}$, recall as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{|h(e_i)\cap y_i|}{|y_i|}$, and F1 as $\frac{1}{|E|}\sum_{i=1}^{|E|}\frac{2\times|h(e_i)\cap y_i|}{|h(e_i)|+|y_i|}$. For the label-based measures, we report macro- and micro-averaged F1 scores. As a ranking-based measure, we report one error, which is defined as $\frac{1}{|E|}\sum_{i=1}^{|E|}[\![[arg\ \underset{c\in C}{max} f(e_i,c)] \notin y_i]\!]$, $[\![expr]\!] = 1$ if $expr$ is true and $[\![expr]\!] = 0$ otherwise. $f(e_i, c)$ denotes the rank of category $c \in C$ as predicted by the classifier. The one error measure evaluates the number of edits where the highest ranked category in the predictions is not in the set of relevant categories. It becomes smaller when the performance of the classifier increases.

Table 3 shows the overall classification scores. We calculated a random baseline, which multi-labels edits at random considering the label powerset frequencies it has learned from the training data. Furthermore, we calculated a majority category baseline, which labels all edits with the most frequent edit category in the training data. In Figure 2, we list the results for each category, together with the average pair-wise inter-rater agreement (F1 scores). The F1 scores are calculated based on the study we carried out in Daxenberger and Gurevych (2012).

Parameters and Feature selection All parameters have been adjusted on the development set using the RAKEL classifier, aiming to optimize accuracy. With respect to the n-gram features, we tested values for n = 1, 2 and 3. For comment n-grams, unigrams turned out to yield the best overall performance, and bigrams for character and token n-grams. The word and character n-gram spaces are limited to the 500 most frequent items, the comment n-gram space is limited to the 1,500 most frequent items. To transform ranked output into bipartitions, it is necessary to set a threshold. This threshold is reported in Table 3 and has been optimized for each classifier with respect to label cardinality (average number of labels assigned to edits) on the development set. Since most of the traditional feature selection methods cannot be applied directly to multi-labeled data, we used the label powerset approach to transform the multi-labeled data into single-labeled data and subsequently applied χ^2. Feature reduction to the highest-ranked features clearly improved the classifier performance on the development set. We therefore limited the feature space to the 150 highest-ranked features in our experiments.

For the RAKEL classifier, we set $l = 42$ (twice the size of the category set) and $k = 3$. In HOMER, we used BR as transformation method, random distribution of categories to the children nodes and $k = 3$. For all other classifier parameters, we used the default settings as configured in Meka respective Mulan.

4 Discussion

The classifiers significantly outperformed both baselines. RAKEL shows best performance for almost all measures in Table 3. The simpler BR approach, which assumes no dependencies between categories, still outperforms HOMER.

We trained and tested the classifier with different feature groups (see Table 1), to analyze the importance of single types of features. As shown in Figure 2, textual features had the highest impact on classification performance. On the opposite, language features played a minor role in our experiments. Among the highest ranked individual features for the entire set of categories, we find textual (*Levenshtein distance*, *Simple edit type*), markup (*Diff number*

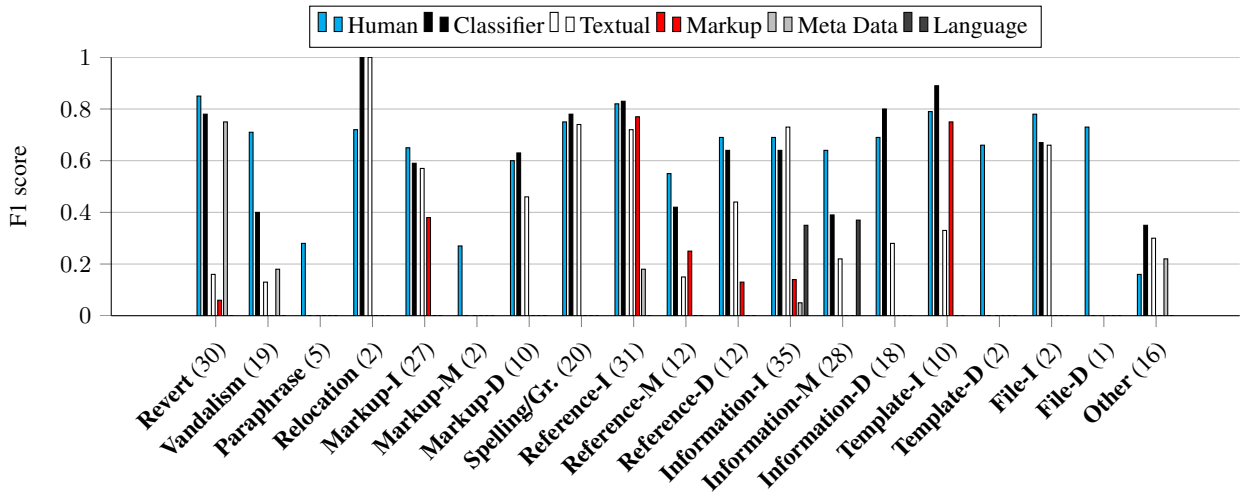


Figure 2: F1 scores of RAKEL with C4.5 as base classifier for individual categories. We add human inter-annotator agreement as average pair-wise F1 scores as well as F1 scores for classifiers trained and tested on single feature groups, cf. Table 1. The number of edits labeled with each category in the test set is given in brackets. The FILE-M and TEMPLATE-M categories are omitted in this Figure, as they had no examples in the development or test set.

markup elements) and meta data (*Number of edits*) features.

Bronner and Monz (2012) report an accuracy of .88 for their best performing system on the binary classification task of distinguishing fluency and factual edits. The best performing classifier in their study was Random Forests (Breiman, 2001). To compare our features with their approach, we mapped the 21 edit categories from Daxenberger and Gurevych (2012) to the binary category set (factual vs. fluency) of Bronner and Monz (2012). Edits labeled as SPELLING/GRAMMAR, MARKUP, RELOCATION and PARAPHRASE are considered fluency edits, the remaining categories factual edits. We removed all edits labeled as OTHER, REVERT or VANDALISM from WPEC. After applying the category mapping, we deleted all edits which were labeled with both the fluency and factual category. The latter may happen due to multi-labeling. This resulted in 1,262 edits labeled as either fluency or factual. On the 80% training split from Table 2, we trained a Random Forests classifier with the optimized feature set and feature reduction as described in Section 3.3. The number of trees was set to 100, with unlimited depth. On the remaining data (test and development split), we achieved an accuracy of .90. Although we did not use the same data set as Bronner and Monz (2012), this result suggests that our feature set is suited for related tasks such as fluency detection.

With respect to vandalism detection in Wikipedia, state-of-the-art systems have a performance of around .82 to .85 AUC-PR on the English Wikipedia (Adler et al., 2011). We suspect that the low performance of our system for Vandalism edits is mostly due to a lower amount of training data, a higher skew in the training and test data and the fact that we did not include features which inform about future actions (e.g. whether a revision is reverted).

Error Analysis Sparseness is a major problem for some of the 21 categories, as shown in Figure 2 by categories such as FILE-D, TEMPLATE-D, MARKUP-M or PARAPHRASE which have only very few examples in training, development and test set. Categories with low inter-annotator agreement in WPEC such as MARKUP-M, PARAPHRASE or OTHER also yielded low classification accuracy. We analyzed frequent errors of the classifier with the help of a confusion matrix. PARAPHRASE edits have been confused with INFORMATION-M by the classifier. Furthermore, the classifier had problems to distinguish between VANDALISM and REVERT as well as INFORMATION-I. Generally, modifications as compared to insertions or deletions perform worse. All of the classifiers we tested, build

their predictions by thresholding over a ranking, cf. Table 3. This generates a source of errors, because the classifier is not able to make a prediction, if it does not have enough confidence for any of the categories. The imbalance of the data, because of the high skew in the category distribution, is another reason for classification errors. In ambiguous cases, the classifier will be biased toward the category with more examples in the training data.

5 A closer look at edit sequences: Mining collaboration patterns

An edit category classifier allows us to label entire article revision histories. We applied the best-performing model from Section 3.3 trained on the entire WPEC to automatically classify all edits in the Wikipedia Quality Assessment Corpus (WPQAC) as presented in previous work (Daxenberger and Gurevych, 2012). WPQAC consists of 10 featured and 10 non-featured articles[7], with an overall number of 21,578 revisions (9,986 revisions from featured articles and 11,592 from non-featured articles), extracted from the April 2011 English Wikipedia dump. The articles in WPQAC are carefully chosen to form comparable pairs of featured and non-featured articles, which should reduce the noise of external influences on edit activity such as popularity or visibility. In Daxenberger and Gurevych (2012), we have shown significant differences in the edit category distribution of articles with featured status before and after the articles were featured. We concluded that articles become more stable after being featured, as shown by the higher number of surface edits and lower number of meaning-changing edits.

Different to our previous approach which is based on the mere distribution of edit categories, in the present study we include the chronological order of edits and use a 10 times larger amount of data for our experiments. We segmented all adjacent revisions in WPQAC into edits, following the approach explained in Daxenberger and Gurevych (2012). During the classification process, we discarded revisions where the classifier could not assign any of the 21 edit categories with a confidence higher than the threshold, cf. Table 3. This resulted in 17,640 remaining revisions. We applied a sequential pattern mining algorithm with time constraints (Hirate and Yamana, 2006; Fournier-Viger et al., 2008) to the data. The latter is based on the PrefixSpan algorithm (Pei et al., 2004). Calculations have been carried out within the open-source SPMF Java data mining platform.[8]

We created one time-extended sequence database for the 10 featured articles and one for the 10 non-featured articles. The sequence databases consist of one row per article. Each row is a chronologically ordered list of revisions. Each revision is represented by the itemset of all edit categories for all edits in that revision (in alphabetical order).

The output of the algorithm are sequential patterns with time constraints. To obtain meaningful results, we constrained the output with the following parameters:

- Minimum support: 1 (the patterns have to be present in each article)
- Time interval allowed between two successive itemsets in the patterns: 1 (patterns are extracted only from adjacent revisions)
- Minimum time interval between the first itemset and the last itemset in the patterns: 1 (the length of the patterns is 2 or higher)

As this output reflects recurring sequences of adjacent revisions labeled with edit categories, we refer to it as *collaboration patterns*. With these parameters, the algorithm discovered 1,358 sequential patterns for featured articles and 968 for non-featured articles. The number of shared patterns in featured and non-featured articles is 427, this corresponds to the number of frequent patterns in a sequence database which contains all 20 featured and non-featured articles. The maximum length of patterns which were found was 6 for featured articles, and 5 for non-featured articles. These numbers show that the defined collaboration patterns seem to have discriminative power for different kinds of articles. Featured articles can be characterized by a higher

[7] http://en.wikipedia.org/wiki/Wikipedia:FA

[8] http://www.philippe-fournier-viger.com/spmf

Featured	① INFORMATION-I ② INFORMATION-I ③ INFORMATION-I ④ INFORMATION-I ⑤ INFORMATION-I
	① INFORMATION-D, INFORMATION-I ② INFORMATION-I ③ INFORMATION-I ④ REFERENCE-I
	① TEMPLATE-D ② REFERENCE-I
Non-Featured	① INFORMATION-I ② INFORMATION-I, REFERENCE-I ③ INFORMATION-I ④ REFERENCE-I ⑤ MARKUP-I
	① MARKUP-I ② REFERENCE-D ③ MARKUP-I
	① VANDALISM ② REVERT

Table 4: Examples of collaboration patterns which have been found in either all featured or all non-featured articles of WPQAC.

degree of homogeneity with respect to their collaborative patterns due to a higher number and length of frequent sequential patterns in featured articles as compared to non-featured articles.

In Table 4, we list some examples of collaboration patterns with a minimum support of 1 which we found in featured, but not non-featured articles, or vice verse. Unsurprisingly, patterns which contain combinations of the most frequent categories (INFORMATION-I, REFERENCE-I), have a high overall frequency. The diversity inside collaboration patterns measured by the number of different edit categories was higher in non-featured articles. For example, the VANDALISM - REVERT pattern was only found in non-featured articles. Patterns in featured articles tended to be more homogeneous, as shown by the first pattern in Table 4, a repetition of additions of information. We conclude that distinguished, high-quality articles, show a higher degree of homogeneity as compared to a subset of non-featured articles and the overall corpus.

6 Conclusion

In this study, we evaluated a novel feature set for building a model to automatically classify Wikipedia edits. Using a freely available corpus (Daxenberger and Gurevych, 2012), our model achieved a micro-averaged F1 score of .62 classifying edits within a range of 21 categories. Textual features had the highest impact on classifier performance, whereas language features play a minor role. The same classifier model obtained state-of-the-art performance on the related task of fluency edit classification. Applications which potentially benefit from our work include the analysis of the writing process in collaboratively created documents, such as wikis or research papers. We have demonstrated how our model can be used to detect collaboration patterns in article revision histories. On a subset of articles from the English Wikipedia, we found that high-quality articles show a higher degree of homogeneity in their collaborative patterns as compared to random articles. Furthermore, automatic edit category classification allows to generate huge amounts of category-filtered training data for NLP tasks, e.g. spelling and grammar correction or vandalism detection. With respect to future work, we plan to include more resources, e.g. the PAN-WVC-10 (Potthast and Holfeld, 2011) or WiCoPaCo (Max and Wisniewski, 2010) to increase the size of training data. A larger amount of labeled data would certainly help to improve the classifier performance for weak categories (e.g. VANDALISM and PARAPHRASE) and sparse categories (e.g. TEMPLATE-D, MARKUP-M). Based on our trained classifier, annotating more examples can be alleviated with the help of active learning.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz" (LOEWE) as part of the research center "Digital Humanities". We thank the anonymous reviewers for their valuable feedback.

References

B Thomas Adler, Luca Alfaro, Santiago M Mola-Velasco, Paolo Rosso, and Andrew G West. 2011. Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features. In Alexander Gelbukh, editor, *Computational Linguistics*

and Intelligent Text Processing, Lecture Notes in Computer Science, pages 277–288. Springer.

Ablimit Aji, Yu Wang, and Eugene Agichtein. 2010. Using the Past To Score the Present: Extending Term Weighting Models Through Revision History Analysis. *ReCALL*, pages 629–638.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. In *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pages 435–40, Montreal, Canada, USA.

Leo Breiman. 2001. Random Forests. *Machine Learning*, 45(1):5–32.

Amit Bronner and Christof Monz. 2012. User Edits Classification Using Document Revision Histories. In *European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 356–366, Avignon, France.

Elena Cabrio, Bernardo Magnini, and Angelina Ivanova. 2012. Extracting Context-Rich Entailment Rules from Wikipedia Revision History. In *Proceedings of the 3rd Workshop on The People's Web meets NLP*, pages 34–43, Jeju Island, Republic of Korea.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 507–517, Atlanta, GA, USA.

Johannes Daxenberger and Iryna Gurevych. 2012. A Corpus-Based Study of Edit Categories in Featured and Non-Featured Wikipedia Articles. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 711–726, Mumbai, India.

Hannes Dohrn and Dirk Riehle. 2011. Design and implementation of the Sweble Wikitext parser. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 72–81, Mountain View, CA, USA.

Camille Dutrey, Houda Bouamor, Delphine Bernhard, and Aurélien Max. 2011. Local modifications and paraphrases in Wikipedia's revision history. *Procesamiento del Lenguaje Natural*, 46:51–58.

Richard Eckart de Castilho, Iryna Gurevych, and Richard Eckart de Castilho. 2011. A Lightweight Framework for Reproducible Parameter Sweeping in Information Retrieval. In *Proceedings of the Workshop on Data Infrastructures for Supporting Information Retrieval Evaluation*, pages 7–10, Glasgow, UK.

Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. 2011. Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia's Edit History. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. System Demonstrations*, pages 97–102, Portland, OR, USA.

Oliver Ferschke, Johannes Daxenberger, and Iryna Gurevych. 2013. A Survey of NLP Methods and Resources for Analyzing the Collaborative Writing Process in Wikipedia. In Iryna Gurevych and Jungi Kim, editors, *The Peoples Web Meets NLP: Collaboratively Constructed Language Resources*, Theory and Applications of Natural Language Processing, chapter 5. Springer.

Philippe Fournier-Viger, Roger Nkambou, and Engelbert Mephu Nguifo. 2008. A Knowledge Discovery Framework for Learning Task Models from User Interactions in Intelligent Tutoring Systems. In Alexander Gelbukh and Eduardo F. Morales, editors, *Proceedings of the 7th Mexican International Conference on Artificial Intelligence*, Lecture Notes in Computer Science, pages 765–778. Springer.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

Yu Hirate and Hayato Yamana. 2006. Generalized Sequential Pattern Mining with Item Intervals. *Journal of Computers*, 1(3):51–60.

Sara Javanmardi, David W. McDonald, and Cristina V. Lopes. 2011. Vandalism Detection in Wikipedia: A High-Performing, Feature-Rich Model and its Reduction Through Lasso. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 82–90, Mountain View, CA, USA.

Brian Keegan, Darren Gergle, and Noshir Contractor. 2013. Hot Off the Wiki: Structures and Dynamics of Wikipedia's Coverage of Breaking News Events. *American Behavioral Scientist*, 57(5):595–622, May.

Jun Liu and Sudha Ram. 2011. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. *ACM Trans. Management Inf. Syst.*, 2(2):11.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, pages 182–190, Montréal, Canada.

Aurélien Max and Guillaume Wisniewski. 2010. Mining Naturally-occurring Corrections and Paraphrases from Wikipedias Revision History. In *Proceedings of the 7th Conference on International Language Resources and Evaluation*, Valletta, Malta.

Rani Nelken and Elif Yamangil. 2008. Mining Wikipedia's Article Revision History for Training Computational Linguistics Algorithms. In *Proceedings of the 1st AAAI Workshop on Wikipedia and Artificial Intelligence*, pages 31–36, Chicago, IL, USA.

Sérgio Nunes, Cristina Ribeiro, and Gabriel David. 2011. Term weighting based on document revision history. *Journal of the American Society for Information Science and Technology*, 62(12):2471–2478.

Philip V. Ogren, Philipp G. Wetzler, and Steven Bethard. 2008. ClearTK: A UIMA toolkit for statistical natural language processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*, pages 32–38, Marrakech, Morocco.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440.

Martin Potthast and Teresa Holfeld. 2011. Overview of the 2nd International Competition on Wikipedia Vandalism Detection. In *Notebook Papers of CLEF 2011 Labs and Workshops*, Amsterdam, Netherlands.

Martin Potthast, Tim Gollub, Matthias Hagen, Johannes Kiesel, Maximilian Michel, Arnd Oberländer, Martin Tippmann, Alberto Barrón-Cedeño, Parth Gupta, Paolo Rosso, and Benno Stein. 2012. Overview of the 4th International Competition on Plagiarism Detection. In *CLEF 2012 Evaluation Labs and Workshop Working Notes Papers*, Rome, Italy.

J. Ross Quinlan. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, pages 1650–1659, Sofia, Bulgaria.

Bongwon Suh, Gregorio Convertino, Ed H. Chi, and Peter Pirolli. 2009. The singularity is not near: slowing growth of Wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, Orlando, FL, USA.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis Vlahavas. 2008. Multi-label classification of music into emotions. In *9th International Conference on Music Information Retrieval*, pages 325–330, Philadelphia, PA, USA.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2008. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proceedings of the ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, Antwerp, Belgium.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, chapter 34, pages 667–685. Springer.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2011. Random k-Labelsets for Multi-Label Classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089.

Kristian Woodsend and Mirella Lapata. 2011. Learning to Simplify Sentences with Quasi-Synchronous Grammar and Integer Programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK.

Elif Yamangil and Rani Nelken. 2008. Mining Wikipedia Revision Histories for Improving Sentence Compression. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Short Papers*, pages 137–140, Columbus, OH, USA.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: unsupervised extraction of lexical simplifications from Wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 365–368, Los Angeles, CA, USA.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora from Wikipedia using co-training. In *Proceedings of the COLING-Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 28–36, Beijing, China.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using Wiktionary for Computing Semantic Relatedness. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 861–866, Chicago, IL, USA.

Torsten Zesch. 2012. Measuring Contextual Fitness Using Error Contexts Extracted from the Wikipedia Revision History. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538, Avignon, France.

Semi-Markov Phrase-based Monolingual Alignment

Xuchen Yao and **Benjamin Van Durme**
Johns Hopkins University
Baltimore, MD, USA

Chris Callison-Burch*
University of Pennsylvania
Philadelphia, PA, USA

Peter Clark
Allen Institute for Artificial Intelligence
Seattle, WA, USA

Abstract

We introduce a novel discriminative model for phrase-based monolingual alignment using a semi-Markov CRF. Our model achieves state-of-the-art alignment accuracy on two phrase-based alignment datasets (RTE and paraphrase), while doing significantly better than other strong baselines in both non-identical alignment and phrase-only alignment. Additional experiments highlight the potential benefit of our alignment model to RTE, paraphrase identification and question answering, where even a naive application of our model's alignment score approaches the state of the art.

1 Introduction

Various NLP tasks can be treated as an alignment problem: machine translation (aligning words in one language with words in another language), question answering (aligning question words with the answer phrase), textual entailment recognition (aligning premise with hypothesis), paraphrase detection (aligning semantically equivalent words), etc. Even though most of these tasks involve only a single language, alignment research has primarily focused on the bilingual setting (i.e., machine translation) rather than monolingual. Moreover, most work has considered token-based approaches over phrase-based.[1] Here we seek to address this imbalance by proposing better phrase-based models for monolingual word alignment.

Most token-based alignment models can extrinsically handle phrase-based alignment to some extent. For instance, in the case of `NYC` aligning to `New York City`, the single source word `NYC` may align three times separately to the target words: `NYC`↔`New`, `NYC`↔`York`, `NYC`↔`City`. Or in the case of identical alignment, `New York City` aligning to `New York City` is simply `New`↔`New`, `York`↔`York`, `City`↔`City`. However, it is not as clear how to token-align `New York` (as a city) with `New York City`. The problem is more prominent when aligning phrasal paraphrases or multiword expressions, such as `pass away` and `kick the bucket`. This suggests an intrinsically phrase-based alignment model.

The token aligner jacana-align (Yao et al., 2013a) has achieved state-of-the-art result on the task of monolingual alignment, based on previous work of Blunsom and Cohn (2006). It employs a Conditional Random Field (Lafferty et al., 2001) to align tokens from the source sentence to tokens in the target sentence, by treating source tokens as "observation" and target tokens as "hidden states". However, it is not designed to handle phrase-based alignment, largely due to the Markov nature of the underlying model: a state can only span one token each time, making it unable to align multiple consecutive tokens (i.e. a phrase). We extend this model by introducing semi-Markov states for phrase-based alignment: a state can instead span multiple consecutive time steps, thus aligning phrases on the source side. Also, we merge phrases on the target side to phrasal states, allowing the model to align phrases on the target side as well. We evaluate the resulting semi-Markov

*Performed while faculty at Johns Hopkins University.

[1] In this paper we use the term token-based alignment for one-to-one alignment and phrase-based for non one-to-one alignment, and word alignment in general for both.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 590–600,
Seattle, Washington, USA, 18-21 October 2013.

CRF model on the task of phrase-based alignment, and then show a basic application in the NLP tasks of recognizing textual entailment, paraphrase identification, and question answering sentence ranking. The final phrase-based aligner is open-source.[2]

2 Related Work

Most work in monolingual alignment employs dependency tree/graph matching algorithms, including tree edit distance (Punyakanok et al., 2004; Kouylekov and Magnini, 2005; Heilman and Smith, 2010; Yao et al., 2013b), Particle Swarm Optimization (Mehdad, 2009), linear regression/classification models (Chambers et al., 2007; Wang and Manning, 2010), and min-cut (Roth and Frank, 2012). These works inherently only support token-based alignment, with phrase-like alignment achieved by first merging tokens to phrases as a *preprocessing* step.

The MANLI aligner (MacCartney et al., 2008) and its derivations (Thadani and McKeown, 2011; Thadani et al., 2012) are the first known phrase-based aligners specifically designed for aligning English sentence pairs. It applies discriminative perceptron learning with various features and handles phrase-based alignment of arbitrary phrase lengths. MANLI suffers from slow decoding time due to its large search space. This was optimized by Thadani and McKeown (2011) through Integer Linear Programming (ILP), where benefiting from modern ILP solvers they showed an order-of-magnitude speedup in decoding. Also, various syntactic constraints can be easily added, significantly improving exact alignment match rate for whole sentence pairs. Besides the common application of textual entailment and question answering, monolingual alignment has also been applied in the field of text generation (Barzilay and Lee, 2003; Pang et al., 2003).

Word alignment has been more explored in machine translation. The IBM models (Brown et al., 1993) allow many-to-one alignment and are essentially asymmetric. Phrase-based MT historically relied on heuristics (Koehn, 2010) to merge two sets of word alignment in opposite directions to yield phrasal alignment. Later, researchers explored non-heuristic phrase-based methods. Among them, Marcu and Wong (2002) described a joint probability model that generates both the source and target sentences simultaneously. All possible pairs of phrases in both sentences are enumerated and then pruned with statistical evidence. Deng and Byrne (2008) explored token-to-phrase alignment based on HMM models (Vogel et al., 1996) by explicitly modeling the token-to-phrase probability and phrase lengths. However, the token-to-phrase alignment is only in one direction: each target state still only spans one source word, and thus alignment on the source side is limited to tokens. Andrés-Ferrer and Juan (2009) extended the HMM-based method to Hidden Semi-Markov Models (HSMM) (Ostendorf et al., 1996), allowing phrasal alignments on the source side. Finally, Bansal et al. (2011) unified the HSMM models with the alignment by agreement framework (Liang et al., 2006), achieving phrasal alignment that agreed in both directions.

Despite successful usage of generative semi-Markov models in bilingual alignment, this has not been followed with models in discriminative monolingual alignment. Essentially monolingual alignment would benefit more from discriminative models with various feature extractions (just like those defined in MANLI) than generative models without any predefined feature (just like how they were used in bilingual alignment). To combine the strengths of both semi-Markov models and discriminative training, we propose to use the semi-Markov Conditional Random Field (Sarawagi and Cohen, 2004), which was first used in information extraction to tag continuous segments of input sequences and outperformed conventional CRFs in the task of named entity recognition. We describe this model in the following section.

3 The Alignment Model

Our objective is to define a model that supports phrase-based alignment of arbitrary phrase length. In this section we first describe a regular CRF model that supports one-to-one token-based alignment (Blunsom and Cohn, 2006; Yao et al., 2013a), then extend it to phrase-based alignment with the semi-Markov model.

[2] http://code.google.com/p/jacana/

3.1 Token-based Model

Given a source sentence $\mathbf{s}$ of length M, and a target sentence $\mathbf{t}$ of length N, the alignment from $\mathbf{s}$ to $\mathbf{t}$ is a sequence of target word indices $\mathbf{a}$, where $a_{i\in[1,M]} \in [0, N]$. We specify that when $a_i = 0$, source word s_i is aligned to a NULL state, i.e., deleted. This models a many-to-one alignment from source to target: multiple source words can be aligned to the same target word, but not vice versa. One-to-many alignment can be obtained by running the aligner in the other direction. The probability of alignment sequence $\mathbf{a}$ conditioned on both $\mathbf{s}$ and $\mathbf{t}$ is then:

$$p(\mathbf{a} \mid \mathbf{s}, \mathbf{t}) = \frac{\exp(\sum_{i,k} \lambda_k f_k(a_{i-1}, a_i, \mathbf{s}, \mathbf{t}))}{Z(\mathbf{s}, \mathbf{t})}$$

This assumes a first-order Conditional Random Field (Lafferty et al., 2001). Since the word alignment task is evaluated over F_1, instead of directly optimizing it, we choose a much easier objective (Gimpel and Smith, 2010) and add a cost function to the normalizing function $Z(\mathbf{s}, \mathbf{t})$ in the denominator:

$$Z(\mathbf{s}, \mathbf{t}) = \sum_{\hat{\mathbf{a}}} \exp(\sum_{i,k} \lambda_k f_k(\hat{a}_{i-1}, \hat{a}_i, \mathbf{s}, \mathbf{t}) + cost(\mathbf{a_y}, \hat{\mathbf{a}}))$$

where $\mathbf{a_y}$ is the true alignments. $cost(\mathbf{a_y}, \hat{\mathbf{a}})$ can be viewed as special "features" that encourage decoding to be consistent with true labels. It is only computed during training in the denominator because in the numerator $cost(\mathbf{a_y}, \mathbf{a_y}) = 0$. Hamming cost is used in practice without learning the weights (i.e., uniform weights). The more inconsistence there is between $\mathbf{a_y}$ and $\hat{\mathbf{a}}$, the more penalized is the decoding sequence $\hat{\mathbf{a}}$ through the cost function.

3.2 Phrase-based Model

The token-based model supports $1 : 1$ alignment. We first extend it in the direction of $l_s : 1$, where a target state spans l_s words on the source side (l_s source words align to 1 target word). Then we extend it in the direction of $1 : l_t$, where l_t is the target phrase length a source word aligns to (1 source word aligns to l_t target words). The final combined

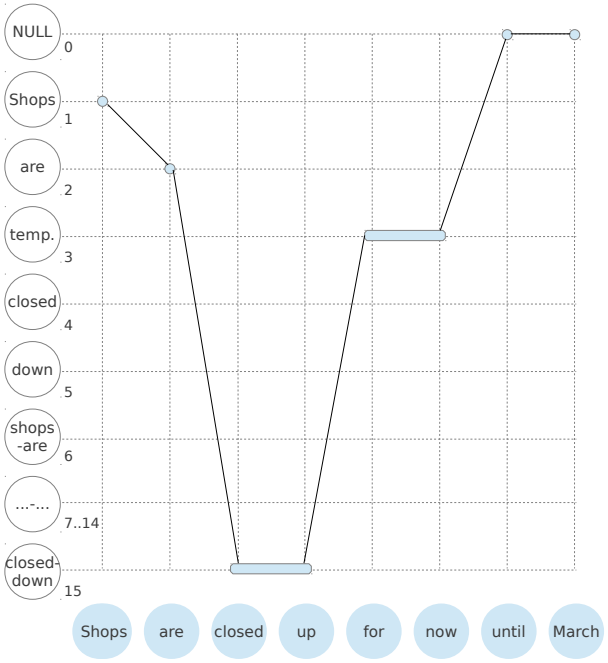


Figure 1: A semi-Markov phrase-based model example and the desired Viterbi decoding path. Shaded horizontal circles represent the source sentence (`Shops are closed up for now until March`) and hollow vertical circles represent the hidden states with state IDs for the target sentence (`Shops are temporarily closed down`). State 0, a NULL state, is designated for deletion. One state (e.g. state 3 and 15) can span multiple consecutive source words (a semi-Markov property) for aligning phrases on the source side. States with an ID larger than the target sentence length indicate "phrasal states" (states 6-15 in this example), where consecutive target tokens are merged for aligning phrases on the target side. Combining the semi-Markov property and phrasal states yields for instance, a 2×2 alignment between `closed up` in the source and `closed down` in the target.

model supports $l_s : l_t$ alignment. Throughout this section we use Figure 1 as an illustrative example, which shows phrasal alignment between the source sentence: (`Shops are closed up for now until March`) and the target sentence: (`Shops are temporarily closed down`).

$1 : 1$ alignment is a special case of $l_s : 1$ alignment where the target side state spans $l_s = 1$ source word, i.e., at *each* time step i, the source side word

s_i aligns to one state a_i and the next aligned state a_{i+1} only depends on the current state a_i. This is the Markovian property of the CRF. When $l_s > 1$, during the time frame $[i, i + l_s)$, all source words $[a_i, a_{i+l_s})$ share the same state a_i. Or in other words, the state a_i "spans" the following l_s time steps. The Markovian property still holds "outside" the time frame l_s, i.e., a_{i+l_s} still only depends on a_i, the previous state l_s time steps ago. But "within" the time frame l_s, the Markovian property does not hold any more: $[a_i, ..., a_{i+l_s-1}]$ are essentially the same state a_i. This is the semi-Markov property . States can be distinguished by this property into two types: *semi-Markovian states* and *Markovian states*.

We have generalized the regular CRF to a semi-Markov CRF. Now we define it by generalizing the feature function:

$$p(\mathbf{a} \mid \mathbf{s}, \mathbf{t}) = \frac{\exp(\sum_{i,k,l_s} \lambda_k f_k(a_{i-l_s}, a_i, \mathbf{s}, \mathbf{t}))}{Z(\mathbf{s}, \mathbf{t})}$$

At time i, the k-th feature function f_k mainly extracts features from the pair of source words $(s_{i-l_s}, ..., s_i]$ and target word t_{a_i} (still with a special case that $a_i = 0$ marks for deletion). Inference is still Viterbi-like: except for the fact during maximization, the Viterbi algorithm not only checks the previous *one* time step, but *all* l_s time steps. Suppose the allowed maximal source phrase length is L_s, define $V_i(a \mid \mathbf{s}, \mathbf{t})$ as the highest score along the decoding path until time i ending with state a:

$$V_i(a \mid \mathbf{s}, \mathbf{t}) = \max_{a_1, a_2, ... a_{i-1}} p(a_1, a_2, \ldots, a_i = a \mid \mathbf{s}, \mathbf{t})$$

then the recursive maximization is:

$$\begin{aligned} V_i(a \mid \mathbf{s}, \mathbf{t}) &= \max_{a'} \max_{l_s = 1 ... L_s} [V_{i-l_s}(a' \mid \mathbf{s}, \mathbf{t}) \\ &\quad + \Psi_i(a', a, l_s, \mathbf{s}, \mathbf{t})] \end{aligned}$$

with factor:

$$\Psi_i(a', a, l_s, \mathbf{s}, \mathbf{t}) = \sum_k \lambda_k f_k(a'_{i-l_s}, a_i, \mathbf{s}, \mathbf{t})$$

and the best alignment $\mathbf{a}$ can be obtained by backtracking the last state a_M from $V_M(a_M \mid \mathbf{s}, \mathbf{t})$.

Training a semi-Markov CRF is very similar to the inference, except for replacing maximization with summation. The forward-backward algorithm should also be used to dynamically compute the normalization function $Z(\mathbf{s}, \mathbf{t})$. Compared to regular CRFs, a semi-Markov CRF has a decoding time complexity of $O(L_s M N^2)$, a constant factor L_s (usually 3 or 4) slower.

To extend from $1:1$ alignment to $1:l_t$ alignment with one source word aligning to l_t target words, we simply explode the state space by L_t times with L_t the maximal allowed target phrase length. Thus the states can be represented as an $N \times L_t$ matrix. The state at (j, l_t) represents the target phrase $[t_j, ..., t_{j+l_t})$. In this paper we distinguish states by three types: NULL state ($j = 0, l_t = 0$), *token state* ($l_t = 1$) and *phrasal state* ($l_t > 1$).

To efficiently store and compute these states, we linearize the two dimensional matrix with a linear function mapping uniquely between the state ID and the target phrase offset/span. Suppose the target phrase t_j of length $l_{t_j} \in [1, L_t]$ holds a position $p_{t_j} \in [1, N]$, and the source word s_i is aligned to this state (p_{t_j}, l_{t_j}), a tuple for (position, span). Then state ID a_{s_i} is computed as:

$$a_{s_i}(p_{t_j}, l_{t_j}) = \begin{cases} p_{t_j} & l_{t_j} = 1 \\ N + (p_{t_j} - 1) \times L_t + l_{t_j} & 1 < l_{t_j} \leq L_t \end{cases}$$

Assume in Figure 1, $L_t = 2$, then the state ID for the phrasal state (5, 2) `closed-down` with $p_{t_j} = 5$ for the position of word `down` and $l_{t_j} = 2$ for the span of 2 words (looking "backward" from the word `down`) is: $5 + (5 - 1) \times 2 + 2 = 15$.

Similarly, given a state id a_{s_i}, the original target phrase position and length can be recovered through integer division and modulation. Thus during decoding, if one output state is 15, we would know that it uniquely comes from the phrasal state (5,2), representing the target phrase `closed down`.

This two dimensional definition of state space expands the number of states from $1 + N$ to $1 + L_t N$. Thus the decoding complexity becomes $O(M(L_t N)^2) = O(L_t^2 M N^2)$ with a usual value of 3 or 4 for L_t.

Now we have defined separately the $l_s : 1$ model and the $1 : l_t$ model. We can simply merge them to

have an $l_s : l_t$ alignment model. The semi-Markov property makes it possible for any target states to align phrases on the source side, while the two dimensional state mapping makes it possible for any source words to align phrases on the target side. For instance, in Figure 1, the phrasal state a_{15} represents the two-word phrase `closed down` on the target side, while still spanning for two words on the source side, allowing a 2×2 alignment. State a_{15} is phrasal, and at source word position 3 and 4 (spanning `closed up`) it is semi-Markovian. The final decoding complexity is $O(L_s L_t^2 M N^2)$, a factor of $30 \sim 60$ times slower than the token-based model (with a typical value of 3 or 4 for L_s and L_t).

In the following we describe features.

3.3 Feature Design

We reused features in the original token-based model based on string similarity, POS tags, position, WordNet, distortion and context. Then we used an additional chunker to mark phrase boundaries *only* for feature extraction:

Chunking Features are binary indicators of whether the phrase types of two phrases match. Also, we added indicators for mappings between source phrase types and target phrase types, such as "vp2np", meaning that a verb phrase in the source is mapped to a noun phrase in the target.

Moreover, we introduced the following lexical features:

PPDB Features (Ganitkevitch et al., 2013) include various similarity scores derived from a paraphrase database with 73 million phrasal and 8 million lexical paraphrases. Various paraphrase conditional probability was employed. For instance, for the ADJP/VP phrase pair `capable of` and `able to`, there are the following minus-log probabilities:

$$p(lhs|e1) = 0.1, p(lhs|e2) = 0.3, p(e1|lhs) = 5.0$$
$$p(e1|e2) = 1.3, p(e2|lhs) = 6.7, p(e2|e1) = 2.8$$
$$p(e1|e2, lhs) = 0.6, p(e2|e1, lhs) = 2.3$$

where $e1/e2$ are the phrase pair, and lhs is the left hand side syntactic non-terminal symbol. We did not use the syntactic part (e.g., `the` NP `of` NNS $\leftrightarrow$ `the` NNS `of` NP) of PPDB as we did not make the assumption that the input sentence pairs were well-formed (and newswire-like) English, or even of a language with a parser available. Also, for phrasal alignments, we ruled out those paraphrases spanning multiple syntactic structures, or of different syntactic structures (indicated as `[X]` in PPDB), for instance, `and crazy` $\leftrightarrow$ `, mad`.

Semantic Relatedness Feature is a single scaled number in $[0, 1]$ from the best performing system (Han et al., 2013) of the *Sem 2013 Semantic Textual Similarity (STS) task. We included this feature mainly to deal with cases where "related" words cannot be well measured by either paraphrases or distributional similarities. For instance, in one alignment dataset annotators aligned `married` with `wife`. Adding a few other words as comparison, the Han et al. (2013) system gives the following similarity scores:

```
married/wife:  0.85
married/husband:  0.84
married/child:  0.10
married/stone:  0.01
```

Name Phylogeny Feature (Andrews et al., 2012) is a similarity feature with a string transducer to model how one name evolves to another. Examples below show how similar is the name `Bill` associated with other names in log probability:

```
Bill/Bill:  -0.8
Bill/Billy:  -5.2
Bill/William:  -13.6
Bill/Mary:  -18.6
```

Finally, one decision we made during feature design was not to use any parsing-based features, with a permissive assumption that the input might not be well-formed English, or even not complete sentences (such as fragmented snippets from web search). The "deepest" linguistic processing stays at the level of tagging and chunking, making the model more easily extendable to other languages.

3.4 Feature Value

In this phrase-based model, the width of a state span over the source words depends on the competition between features fired on the phrases as a whole vs. the consecutive but individual tokens. We found it critical to assign feature values "fairly" among tokens and phrases to make sure that semi-Markov states and phrasal states fire up often enough for phrasal alignments.

	train	test	length	%align.
MSR06	800	800	29/11	36%
Edinburgh++	715	305	22/22	78%

Table 1: Statistics of the two manually aligned corpora, divided into training and test in sentence pairs. The length column shows average lengths of source and target sentences in a pair. %align. is the percentage of aligned tokens.

	1x1	1x2	1x3	2x2	2x3	3x3	more
MSR06	89.2	1.9	0.3	5.7	0.0	1.9	0.8
EDB++	81.9	3.5	0.8	8.3	0.4	3.0	2.1

Table 2: Percentage of various alignment sizes (undirectional, e.g., 1x2 and 2x1 are merged) after synthesizing phrasal alignment from token alignment in the *training* portion of two corpora.

To illustrate this in a simplified way, take `closed up↔closed down` in Figure 1, and assume the only feature is the normalized number of matching tokens in the pair. Then this feature firing on the following pairs would have values (the normalization factor is the maximal phrase length):

`closed↔closed`	1.0
`closed up↔closed`	0.5
`closed up↔up`	0.5
`closed up↔closed down`	0.5
`...↔...`	...

The desired alignment `closed up↔closed down` would not have survived the state competition due to its weak feature value. In this case the model would simply prefer a token alignment `closed↔closed` and up↔`...` (probably NULL).

Thus we upweighted feature values by the maximum source or target phrase length to encourage phrasal alignments, in this case `closed up ↔closed down`:1.0. Then this alignment would have a better chance to be picked out with additional features, such as with the PPDB and Semantic Relatedness Features, which are also upweighted by maximum phrase lengths.

4 Experiment

4.1 Data Preparation

There are two annotated datasets for training and testing. **MSR06**[3] (Brockett, 2007) has annotated alignments on the 2006 PASCAL RTE2 development and test corpora, with 1600 pairs in total. Semantically equivalent words and phrases in the premise and hypothesis sentences are aligned in a manner analogous to alignments in statistical machine translation. This dataset is asymmetric: on average the premises contain 29 words and the hypotheses 11 words. **Edinburgh++**[4] (Thadani et al., 2012) is a revised version of the Edinburgh paraphrase corpus(Cohn et al., 2008) with sentences from the following resources: 1. the Multiple-Translation Chinese corpus; 2. Jules Verne's novel Twenty Thousand Leagues Under the Sea. 3. the Microsoft Research paraphrase corpus (Dolan et al., 2004). The corpus is more balanced and symmetric: the source and target sentences are both 22 words long on average. Table 1 shows some statistics.

Both corpora contain mostly token-based alignment. For MSR06, MacCartney et al. (2008) showed that setting the allowable phrase size to be greater than one only increased F_1 by 0.2%. For Edinburgh++, the annotation guideline[5] explicitly instructs to "prefer smaller alignments whenever possible". Statistics shows that single token alignment counts 96% and 95% of total alignments in these two corpora separately. With such a heavy imbalance towards only token-based alignment, a phrase-based aligner would learn feature weights that award token alignments more than phrasal alignments.

Thus we synthesized phrasal alignments from continuous monotonic token alignments in these two corpora. We first ran the OpenNLP chunker through the corpora. Then for each phrase pair, if each token in the source phrase is aligned to a token in the target phrase in a monotonic way, and vice versa, we

[3] `http://www.cs.biu.ac.il/~nlp/files/RTE_2006_Aligned.zip`

[4] `http://www.ling.ohio-state.edu/~scott/#edinburgh-plusplus`

[5] `http://staffwww.dcs.shef.ac.uk/people/T.Cohn/paraphrase_guidelines.pdf`

merge these alignments to form one single phrasal alignment.[6] Table 2 lists the percentage of various alignment sizes after the merge. Two observations can be made: first, the portion of phrasal alignments increases to $10\% \sim 20\%$ after merging; second, allowing a maximal phrase length of 3 covers $98\% \sim 99\%$ of total alignments, thus a phrase length larger than 3 would be a bad trade-off for coverage vs speed.

4.2 Baselines and Evaluation Metrics

MacCartney et al. (2008) and Yao et al. (2013a) showed that the traditional MT bilingual aligner GIZA++ (Och and Ney, 2003) presented weak results on the task of monolingual alignment. Thus we instead used four other strong baselines:

Meteor (Denkowski and Lavie, 2011): a system for evaluating machine translation by aligning MT output with reference sentences. It is designed for the task of monolingual alignment and supports phrasal alignment. We used version 1.4 and default weights to optimize by maximum accuracy.

MANLI-constraint (Thadani and McKeown, 2011): a re-implemented MANLI system with ILP-powered decoding for speed and hard syntactic constraints to boost exact match rate, with reported numbers on MSR06.

MANLI-joint (Thadani et al., 2012): an improved version of MANLI-constraint that not only models phrasal alignments, but also alignments between dependency arcs, with reported numbers on the original Edinburgh paraphrase corpus.

jacana-token (Yao et al., 2013a): a token-based aligner with state-of-the-art performance on MSR06.

Note that the jacana-token aligner is open-source, so we were able to re-train it with exactly the same feature set used by our phrase-based model. This allows a fair comparison of model performance (token-based vs. phrase-based). The MANLI* systems are not available, thus we only reported their numbers from published papers.

The standard evaluation metrics for alignments are precision (P), recall (R), F_1, and exact matching rate (E) based on either tokens (two tokens are considered aligned iff they are aligned) or phrases (two tokens are considered aligned iff they are contained within phrases that are aligned). Following Thadani et al. (2012), we only report the results based on token alignments (which allows a partial credit if their containing phrases are not aligned), even for the phrase-based alignment task. The reasoning is that if a phrase-based aligner is already doing better than a token aligner in terms of token alignment scores, then the difference in terms of phrase alignment scores will be even larger. Thus showing the superiority of token alignment scores is sufficient.

4.3 Implementation and Training

The elements in the phrase-based model: dynamic state indices, semi-Markov and phrasal states, are not typically found in standard CRF implementations. Thus we implemented the phrase-based model in the Scala programming language, which is fully interoperable with Java, using one semi-Markov CRF package[7] as a reference. We used the L2 regularizer and LBFGS for optimization. OpenNLP[8] provided the POS tagger and chunker and JWNL[9] interfaced with WordNet (Fellbaum, 1998).

4.4 Results

Table 3 gives scores (in bigger fonts) of different aligners on MSR06 and Edinburgh++ and their corresponding phrasal versions. Overall, the token-based aligner did the best on the original corpora, in which single token alignment counts more than 95% of total alignment. The phrase-based aligner did slightly worse. We think the main reason was that it output more phrasal alignment, which in turn harms scores in token-based evaluation (for instance, if the gold alignment is `New`↔`New`, `York`↔`York`, then the phrasal alignment of `New York`↔`New York` would only have half the precision because it inherently also aligns `New` in the source with `York` in the target.). Further investigation showed that on the Edinburgh++ corpus, over-generated phrase-based alignment, when evaluated under just token alignment, contributed hurting about 1.1% of overall F_1,

[6] a few examples: `two Atlanta-based companies`↔`two Atlanta companies`, `the UK`↔`the UK`, `the 17-year-old`↔`the teenager`, `was held`↔`was held`.

[7] `http://crf.sf.net`

[8] `http://opennlp.apache.org/`

[9] `http://jwordnet.sf.net/`

a gap that would make the phrase aligner (85.9%) outperform the token aligner (86.4%).

On the phrasal alignment corpora (represented by MSR06P and EDB++P in Table 3), the phrase-based aligner did significantly better. Note that the overall F_1 and exact match rate are still much lower than those scores obtained from the original corpora, suggesting that the phrasal corpora present a much harder task. Furthermore, as a more "fair" comparison between the two aligners, we synthesized phrasal alignments from the output of the token-based aligner, just as how the phrased-based corpora were prepared, then evaluated its performance again. Still, on the EDB++P corpus, the token aligner was about 1.6% (current difference is 69.1% vs. 72.8%) worse than the phrase-based aligner.

Also, we want to emphasize that since the token-based aligner and the phrase-based aligner shared exactly the same features and lexical resources, the performance boost of the phrase-based aligner on the phrasal corpora results from a better model design: it is the semi-Markov property and phrasal states making the phrase-based aligner better.

To further investigate the performance of aligners with respect to different types of alignment, we divided the scores into those for identical alignments (such as `New`↔`New`) and non-identical alignments (such as `wife`↔`spouse`), indicated by the subscripts *i* and *n* in Table 3. In terms of identical alignment, most aligners were able to score more than 90%, but for non-identical alignment there was noticeable decrease. Still, on the phrasal alignment corpora, the phrase-based model has a much larger recall score for non-identical alignment than others.

We also divided scores with respect to token-only alignment and phrase-only alignment. Due to space limit, we only show results on synthesized Edinburgh++, in Table 4. Meteor and the token aligner inherently have either very limited or no support for phrasal alignment, thus they had very low scores on phrase-only alignment. We then ran the aligners in two directions and merged the results with the "union" MT heuristic to get better phrase support. But that still did not bring F_{1p}'s up to over 5%.

The phrase-based aligner baseline Meteor did worse than our aligners. We think there are two reasons: First, Meteor was not trained on these corpora. Second, Meteor only does strict word, stem, synonym and paraphrase matching but does not use any string similarity measures; this can be supported by the large difference between, for instance, F_{1i} and F_{1n}. In general Meteor did well on identical alignment, but not so well on non-identical alignment.

	System	P% P_i/P_n	R% R_i/R_n	F1% F_{1i}/F_{1n}	E%
MSR06 (78.6%)	Meteor	82.5 89.9/39.9	81.2 97.3/24.6	81.9 93.5/30.5	15.0
	MANLI-cons.	89.5	**86.2**	87.8	**33.0**
	token	**93.6** 96.6/77.7	83.5 96.9/35.6	**88.3** 96.8/48.8	32.1
	phrase	92.1 95.7/65.0	82.8 95.9/34.7	86.8 95.8/45.2	29.1
MSR06P (59.0%)	Meteor	82.5 89.9/40.1	68.3 97.3/8.8	74.7 93.5/14.5	7.3
	token	**92.9** 95.5/77.5	66.1 94.3/11.1	77.2 94.9/19.5	13.5
	phrase	83.5 94.9/55.5	**77.0** 94.2/48.1	**80.1** 94.5/51.5	**14.3**
EDB++ (75.2%)	Meteor	88.3 94.0/61.4	80.5 97.8/24.1	84.2 95.9/34.7	12.7
	MANLI-jnt*	76.6	**83.8**	79.2	12.2
	token	**91.3** 96.4/63.9	82.0 97.4/36.4	**86.4** 96.9/46.4	**15.0**
	phrase	90.4 96.0/57.4	81.9 97.8/38.3	85.9 96.9/46.0	13.7
EDB++P (51.7%)	Meteor	88.4 94.0/61.9	60.6 97.0/6.5	71.9 95.5/11.7	**2.9**
	token	**90.7** 96.2/58.6	55.8 91.3/7.1	69.1 93.7/12.7	2.3
	phrase	82.3 95.6/60.4	**65.3** 93.1/34.3	**72.8** 94.4/43.8	1.6

Table 3: Results on original (mostly token) and phrasal (P) alignment corpora, where (x%) indicates how much alignment is identical alignment, such as `New`↔`New`. E% stands for exact (perfect) match rate. Subscript *i* stands for corresponding scores for "identical" alignment and *n* for "non-identical". *: scores of MANLI-joint were for the original Edinburgh corpus instead of Edinburgh++ (with hand corrections) so it is not a direct comparison.

5 Applications

Natural language alignment can be applied to various NLP tasks. While how to most effectively apply

	System	P% P_t/P_p	R% R_t/R_p	F1% F_{1t}/F_{1p}	E%
EDB++P	Meteor	88.4 59.5/14.9	60.6 90.6/1.1	71.9 71.8/2.0	**2.9**
	token	**90.7** 59.4/21.4	55.8 85.5/0.9	69.1 70.1/1.7	2.3
	phrase	82.3 73.3/48.0	**65.3** 73.5/44.2	**72.8** 73.4/46.0	1.6

Table 4: Same results on the phrasal Edinburgh++ corpus but with scores divided by token-only alignment (subscript t) and phrase-only alignment (subscript p).

it is another topic, we simply show in this section using *just* alignment scores in binary prediction problems. Specifically, we pick the tasks of recognizing textual entailment (RTE), paraphrase identification (PP), and question answering sentence ranking (QA) described in Heilman and Smith (2010):

RTE: predicting whether a hypothesis can be inferred from the premise, with training data from RTE-1/2 and RTE-3 dev, and test from RTE-3 test.

PP: predicting whether two sentences are paraphrases, with training and test data from the MSR Paraphrase Corpus (Dolan et al., 2004).

QA: predicting whether a sentence contains the answer to the question, with training data from TREC-8 to TREC-12 and test data from TREC-13.

For each aligned pair, we can compute a normalized decoding score. Following MacCartney et al. (2008), we select a threshold score and predict true if the decoding score is above this threshold. For the tasks of RTE and PP, we tuned this threshold w.r.t the maximal accuracy on the training set, then reported performance on the test set. For the task of QA, since the evaluation methods in Mean Average Precision and Mean Reciprocal Rank only need a ranked list of answer sentences, and the scores on the test set are sufficient to provide the ranking, we did not tune anything on training but instead directly ran the aligner on the test set. All three tasks shared the same aligner model trained on the superset of MSR06 and Edinburgh++. Results are reported in Table 5. We could not report on Meteor as Meteor does not explicitly output alignment scores.

We did not expect the aligners to beat any of the

system	A%	P%	R%
de Marneffe et al. (2006)	60.5	61.8	60.2
MacCartney and Manning (2008)	**64.3**	**65.5**	63.9
Heilman and Smith (2010)	62.8	61.9	**71.2**
the token aligner	59.1	61.2	55.4
our phrasal aligner	57.6	57.2	68.8

(a) Recognizing Textual Entailment

system	A%	P%	R%
Wan et al. (2006)	**75.6**	**77**	90
Das and Smith (2009)	73.9	74.9	91.3
Heilman and Smith (2010)	73.2	75.7	87.8
the token aligner	70.0	72.6	88.1
our phrasal aligner	68.1	68.6	**95.8**

(b) Paraphrase Identification

system	MAP	MRR
Cui et al. (2005)	0.4271	0.5259
Wang et al. (2007)	0.6029	0.6852
Heilman and Smith (2010)	0.6091	0.6917
Yao et al. (2013b)	**0.6307**	**0.7477**
the token aligner	0.5982	0.6582
our phrasal aligner	0.6165	0.7333

(c) Question Answering Sentence Ranking

Table 5: Results (Accuracy, Precision, Recall, Mean Average Precision, Mean Reciprocal Rank) on the tasks of RTE, PP and QA.

state-of-the-art result since no sophisticated models were additionally used but only the alignment score. Still, the aligners showed competitive performance. It still follows the pattern from the alignment experiment that the phrasal aligner had higher recall and lower precision than the token aligner in the task of RTE and PP. In the QA task, the phrasal aligner performed better than all systems except for the top one.

6 Conclusion

We have introduced a phrase-to-phrase alignment model based on semi-Markov Conditional Random Fields. The combination of semi-Markov states and phrasal states makes phrasal alignment on both the source and target sides possible. The final phrase-

based aligner performed the best on two phrasal alignment corpora and showed its potential usage in three NLP tasks. Future work includes aligning discontinuous (gappy) phrases and integrating alignment more closely in NLP applications.

Acknowledgement

We thank Vulcan Inc. for funding this work. We also thank Jason Smith, Travis Wolfe, Frank Ferraro and the three anonymous reviewers for their comments and suggestion.

References

Jesús Andrés-Ferrer and Alfons Juan. 2009. A phrase-based hidden semi-markov approach to machine translation. In *Procedings of European Association for Machine Translation (EAMT)*, Barcelona, Spain, May. European Association for Machine Translation.

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2012. Name phylogeny: a generative model of string variation. In *Proceedings of EMNLP 2012*.

Mohit Bansal, Chris Quirk, and Robert Moore. 2011. Gappy phrasal alignment by agreement. In *Proceedings of ACL*, Portland, Oregon, June.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL*, pages 16–23.

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of ACL2006*, pages 65–72.

Chris Brockett. 2007. Aligning the RTE 2006 corpus. Technical report, Microsoft Research.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614, December.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 400–407, New York, NY, USA. ACM.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 468–476, Suntec, Singapore, August. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D Manning. 2006. Learning to distinguish valid textual entailments. In *Second Pascal RTE Challenge Workshop*.

Yonggang Deng and William Byrne. 2008. HMM word and phrase alignment for statistical machine translation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(3):494–507.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of COLING*, Stroudsburg, PA, USA.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: training log-linear models with cost functions. In *NAACL 2010*, pages 733–736.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC-EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL 2010*, pages 1011–1019, Los Angeles, California, June.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Milen Kouylekov and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *PASCAL Challenges on RTE*, pages 17–20.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of ACL 2008*, pages 521–528.

Bill MacCartney, Michel Galley, and Christopher D Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*, pages 802–811.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP-2002*, pages 133–139.

Yashar Mehdad. 2009. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 289–292.

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

Mari Ostendorf, Vassilios V Digalakis, and Owen A Kimball. 1996. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of NAACL*, pages 102–109.

Vasin Punyakanok, Dan Roth, and Wen T. Yih. 2004. Mapping Dependencies Trees: An Application to Question Answering. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida.

Michael Roth and Anette Frank. 2012. Aligning predicates across monolingual comparable texts using graph-based clustering. In *Proceedings of EMNLP-CoNLL*, pages 171–182, Jeju Island, Korea, July.

Sarawagi Sarawagi and William Cohen. 2004. Semimarkov conditional random fields for information extraction. *Advances in Neural Information Processing Systems*, 17:1185–1192.

Kapil Thadani and Kathleen McKeown. 2011. Optimal and syntactically-informed decoding for monolingual phrase-based alignment. In *Proceedings of ACL short*.

Kapil Thadani, Scott Martin, and Michael White. 2012. A joint phrasal and dependency model for paraphrase alignment. In *Proceedings of COLING 2012: Posters*, pages 1229–1238, Mumbai, India, December. The COLING 2012 Organizing Committee.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 836–841.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of COLING*, pages 1164–1172, Stroudsburg, PA, USA.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *Proceedings of EMNLP-CoNLL*, pages 22–32, Prague, Czech Republic, June.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. A Lightweight and High Performance Monolingual Word Aligner. In *Proceedings of ACL 2013 short*, Sofia, Bulgaria.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013b. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of NAACL 2013*.

A Constrained Latent Variable Model for Coreference Resolution

Kai-Wei Chang Rajhans Samdani Dan Roth
University of Illinois at Urbana-Champaign
{kchang10|rsamdan2|danr}@illinois.edu

Abstract

Coreference resolution is a well known clustering task in Natural Language Processing. In this paper, we describe the Latent Left Linking model (L^3M), a novel, principled, and linguistically motivated latent structured prediction approach to coreference resolution. We show that L^3M admits efficient inference and can be augmented with knowledge-based constraints; we also present a fast stochastic gradient based learning. Experiments on ACE and Ontonotes data show that L^3M and its constrained version, CL^3M, are more accurate than several state-of-the-art approaches as well as some structured prediction models proposed in the literature.

1 Introduction

Coreference resolution is a challenging task, that involves identification and clustering of noun phrases mentions that refer to the same real-world entity. Most machine learning approaches to coreference resolution learn a scoring function to estimate the compatibility between two mentions or two sets of previously clustered mentions. Then, a decoding algorithm is designed to aggregate these scores and find an optimal clustering assignment.

The most popular of these frameworks is the pairwise mention model (Soon et al., 2001; Ng and Cardie, 2002; Bengtson and Roth, 2008), which learns a compatibility score of mention-pairs and uses these pairwise scores to obtain a global clustering. Recently, efforts have been made (Haghighi and Klein, 2010; Rahman and Ng, 2011b; Rahman and Ng, 2011c) to consider models that capture higher order interactions, in particular, between mentions and previously identified entities (that is, between mentions and clusters). While such models are potentially more expressive, they are largely based on heuristics to achieve computational tractability.

This paper focuses on a novel and principled machine learning framework that pushes the state-of-the-art while operating at a mention-pair granularity. We present two models — the Latent Left-Linking Model (L^3M), and a version of that is augmented with domain knowledge-based constraints, the Constrained Latent Left-Linking Model (CL^3M). L^3M admits efficient inference, linking each mention to a previously occurring mention to its left, much like the existing best-left-link inference models (Ng and Cardie, 2002; Bengtson and Roth, 2008). However, unlike previous best-link techniques, learning in our case is performed jointly with decoding — we present a novel latent structural SVM approach, optimized using a fast stochastic gradient-based technique. Furthermore, we present a probabilistic generalization of L^3M that is more expressive in that it is capable of considering mention-entity interactions using scores at the mention-pair granularity. We augment this model with a temperature-like parameter (Samdani et al., 2012) to provide additional flexibility.

CL^3M augments L^3M with knowledge-based constraints following (Roth and Yih, 2004; Denis and Baldridge, 2007). This capability is very desirable as shown by the success of the rule-based deterministic approach of Raghunathan et al. (2010) in the CoNLL shared task 2011 (Pradhan et al., 2011). In L^3M, domain-specific constraints are incorporated into learning and inference in a straightforward way. CL^3M scores a mention's contribution to its cluster by combining the corresponding score

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 601–612,
Seattle, Washington, USA, 18-21 October 2013.

of the underlying L^3M model with that from a set of constraints.

Most importantly, in our experiments on benchmark coreference datasets, we show that CL^3M, with just five constraints, compares favorably with other, more complicated, state-of-the-art algorithms on a variety of evaluation metrics. Overall, the main contribution of this paper is a principled machine learning model operating at mention-pair granularity, using easy to implement constraint-augmented inference and learning, that yields competitive results on coreference resolution on Ontonotes-5.0 (Pradhan et al., 2012) and ACE 2004 (NIST, 2004).

2 Related Work

The idea of Latent Left-linking Model (L^3M) is inspired by a popular inference approach to coreference which we call the *Best-Left-Link* approach (Ng and Cardie, 2002; Bengtson and Roth, 2008). In the best-left-link strategy, each mention i is connected to the best antecedent mention j with $j < i$ (i.e. a mention occurring to the left of i, assuming a left-to-right reading order), thereby creating a *left-link*. The "best" antecedent mention is the one with the highest pairwise score, w_{ij}; furthermore, if w_{ij} is below some threshold, say 0, then i is not connected to any antecedent mention. The final clustering is a transitive closure of these "best" links. The intuition behind best-left-link strategy is based on how humans read and decipher coreference links – they mostly rely on information to the left of the mention when deciding whether to add it to a previously constructed cluster or not. This strategy has been successful and commonly used in coreference resolution (Ng and Cardie, 2002; Bengtson and Roth, 2008; Stoyanov et al., 2009). However, most works have developed ad-hoc approaches to implement this idea. For instance, Bengtson and Roth (2008) train a model $\mathbf{w}$ on binary training data generated by taking for each mention, the closest antecedent coreferent mention as a positive example, and all the other mentions as negative examples. Similar approaches to training and, additionally, decoupling the training stage from the clustering stage were used by other systems. In this paper, we formalize the learning problem of the best-left-link model as a structured prediction problem and analyze our system with detailed experiments. Furthermore, we generalize this approach by considering multiple pairwise left-links instead of just the best link, efficiently capturing the notion of a mention-to-cluster link.

Many techniques in the coreference literature break away from the mention pair-based, best-left-link paradigm. Denis and Baldridge (2008) and Ng (2005) learn a local ranker to rank the mention pairs based on their compatibility. While these approaches achieve decent empirical performance, it is unclear why these are the right ways to train the model. Some techniques consider a more expressive model by using features defined over mention-cluster or cluster-cluster (Rahman and Ng, 2011c; Stoyanov and Eisner, 2012; Haghighi and Klein, 2010). For these models, the inference and learning algorithms are usually complicated. Very recently, Durrett et al. (2013) propose a probabilistic model which enforces structural agreement constraints between specified properties of mention cluster when using a mention-pair model. This approach is very related to the probabilistic extension of our method as both models attempt to leverage entity-level information from mention-pair features. However, our approach is simpler because it directly considers the probabilities of multiple links. Furthermore, while their model performs only slightly better than the Stanford rule-based system (Lee et al., 2011), we significantly outperform this system. Most importantly, our model obtains state-of-the-art performance on OntoNotes-5.0 while still operating at the mention-pair granularity. We believe that this is due to our novel and principled structured prediction framework which results in accurate (and efficient) training.

Several structured prediction techniques have been applied to coreference resolution in the machine learning literature. For example, McCallum and Wellner (2003) and Finley and Joachims (2005) model coreference as a correlational clustering problem (Bansal et al., 2002) on a complete graph over the mentions with edge weights given by the pairwise classifier. However, correlational clustering is known to be NP Hard (Bansal et al., 2002); nonetheless, an ILP solver or an approximate inference algorithm can be used to solve this problem. Another approach proposed by Yu and Joachims (2009) formu-

lates coreference with latent spanning trees. However, their approach has no directionality between mentions, whereas our latent structure captures the natural left-to-right ordering of mentions. In our experiments (Sec. 5), we show that our technique vastly outperforms both the spanning tree and the correlational clustering techniques. We also compare with (Fernandes et al., 2012) and the publicly available Stanford coreference system (Raghunathan et al., 2010; Lee et al., 2011), a state-of-the-art rule-based system.

Finally, some research (Ratinov and Roth, 2012; Bansal and Klein, 2012; Rahman and Ng, 2011a) has tried to integrate world knowledge from web-based statistics or knowledge bases into a coreference system. World knowledge is potentially useful for resolving coreference and can be injected into our system in a straightforward way via the constraints framework. We will show an example of incorporating our system with name-entity and WordNet-based similarity metric (Q. Do, 2009) in Sec. 5. Including massive amount of information from knowledge resources is not the focus of this paper and may distort the comparison with other relevant models but our results indicate that this is doable in our model, and may provide significant improvements.

3 Latent Left-Linking Model with Constraints

In this section, we describe our Constrained Latent Left-Linking Model (CL^{3}M). CL^{3}M is inspired by a few ideas from the literature: (a) the popular *Best-Left-Link* inference approach to coreference (Ng and Cardie, 2002; Bengtson and Roth, 2008), and (b) the injection of domain knowledge-based constraints for structured prediction (Roth and Yih, 2004; Clarke and Lapata, 2006; Chang et al., 2012b; Ganchev et al., 2010; Koo et al., 2010; Pascal and Baldridge, 2009).

We first introduce the notion of a pairwise mention-scorer, then introduce our Left-Linking Model (L^{3}M), and finally describe how to inject constraints into our model.

Let d be a document with m_d mentions. Mentions are denoted solely using their indices, ranging from 1 to m_d. A coreference clustering $\mathcal{C}$ for document d is a collection of disjoint sets partitioning the set $\{1, \ldots, m_d\}$. We represent $\mathcal{C}$ as a binary function with $\mathcal{C}(i, j) = 1$ if mentions i and j are coreferent, otherwise $\mathcal{C}(i, j) = 0$. Let $s(\mathcal{C}; \mathbf{w}, d)$ be the score of a given clustering $\mathcal{C}$ for a given document and a given pairwise weight vector $\mathbf{w}$. Then, during inference, a clustering $\mathcal{C}$ is predicted by maximizing the scoring function $s(\mathcal{C}; \mathbf{w}, d)$, over all valid (i.e. satisfying symmetry and transitivity) clustering binary functions $\mathcal{C} : \{1, \ldots, m_d\} \times \{1, \ldots, m_d\} \rightarrow \{0, 1\}$.

3.1 Mention Pair Scorer

We model the task of coreference resolution using a pairwise scorer which indicates the compatibility of a pair of mentions. The inference routine then predicts the final clustering — a structured prediction problem — using these pairwise scores.

Specifically, for any two mentions i and j (w.l.o.g. $j < i$), we produce a pairwise compatibility score w_{ji} using extracted features $\phi(j, i)$ as

$$w_{ji} = \mathbf{w} \cdot \phi(j, i) \ , \qquad (1)$$

where $\mathbf{w}$ is a weight parameter that is learned.

3.2 Latent Left-Linking Model

Our inference algorithm is inspired by the best-left-link approach. In particular, the score $s(\mathcal{C}; d, \mathbf{w})$ is defined so that each mention links to the antecedent mention (to its left) with the highest score (as long as the score is above some threshold, say, 0). Specifically:

$$s(\mathcal{C}; d, \mathbf{w}) = \sum_{i=1}^{m_d} \max_{0 \leq j < i, \mathcal{C}(i,j)=1} \mathbf{w} \cdot \phi(j, i) \ . \qquad (2)$$

In order to simplify the notation, we introduce a dummy mention with index 0, which is to the left (i.e. appears before) of all other mentions and has $w_{0i} = 0$ for all *actual* mentions $i > 0$. For a given clustering $\mathcal{C}$, if a mention i is not co-clustered with any previous actual mention $j, 0 < j < i$, then we assume that i links to 0 and $\mathcal{C}(i, 0) = 1$. In other words, $C(i, 0) = 1$ *iff* i is the first actual item of a cluster in $\mathcal{C}$. However, such an item i is **not** considered to be co-clustered with 0 and for any valid clustering, item 0 is always in a singleton dummy cluster, which is eventually discarded. The important property of the score s is that it is exactly maximized

by the best-left-link inference, as it maximizes individual left link scores and the creation of one left-link does not affect the creation of other left-links.

3.3 Learning

We use a max-margin approach to learn $\mathbf{w}$. We are given a training set D of documents where for each document $d \in D$, $\mathcal{C}_d$ refers to the annotated ground truth clustering. Then we learn $\mathbf{w}$ by minimizing

$$L(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{|D|}\sum_{d\in D}\frac{1}{m_d}\Big(\max_{\mathcal{C}}\big(s(\mathcal{C};d,\mathbf{w}) + \Delta(\mathcal{C},\mathcal{C}_d)\big) - s(\mathcal{C}_d;d,\mathbf{w})\Big)\ ,$$

where $\Delta(\mathcal{C},\mathcal{C}_d)$ is a loss function used in coreference. In order to achieve tractable loss-augmented minimization — something not possible with standard loss functions used in coreference (e.g. B^3 (Bagga and Baldwin, 1998)) — we use a decomposable loss function that just counts the number of mention pairs on which $\mathcal{C}$ and $\mathcal{C}_d$ disagree: $\Delta(\mathcal{C},\mathcal{C}_d) = \sum_{i,j=0,j<i}^{m_d}\mathbb{I}_{\mathcal{C}(i,j)=\mathcal{C}_d(i,j)}$, where $\mathbb{I}$ is a binary indicator function. This loss function is equivalent to the numerator of the Rand index loss (Rand, 1971). With this form of loss function and using the scoring function in Eq. (2), we can write $L(\mathbf{w})$ as

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{|D|}\sum_{d\in D}\frac{1}{m_d}\sum_{i=1}^{m_d}\Big(\max_{0\le j<i}\big(\mathbf{w}\cdot\phi(j,i) + \delta(\mathcal{C}_d,i,j)\big) - \max_{0\le j<i, C(i,j)=1}(\mathbf{w}\cdot\phi(j,i))\Big)\ , \quad (3)$$

where $\delta(\mathcal{C}_d,i,j) = 1 - \mathcal{C}_d(i,j)$ is the loss-based margin that is 1 if i and j are not coreferent in $\mathcal{C}_d$, and is 0 otherwise. In the above objective function, the left-links remain latent while we get to observe the clustering. This objective function is related to latent structural SVMs (Yu and Joachims, 2009). However Yu and Joachims (2009) use a spanning tree based latent structure which does not have the left-to-right directionality we exploit. We can minimize the above function using Concave Convex Procedure (Yuille and Rangarajan, 2003), which is guaranteed to reach the local minima. However, such a procedure is costly as it requires doing inference on all the documents to compute a single gradient update. Consequently, we choose a faster stochastic sub-gradient descent (SGD) approach. Since $L(\mathbf{w})$ in Eq. (3) decomposes not only over training documents, but also over individual mentions in each document, we can perform SGD on a per-mention basis. The stochastic sub-gradient w.r.t. mention i in document d is given by

$$\begin{aligned}\nabla L(\mathbf{w})_d^i &\propto \phi(j',i) - \phi(j'',i) + \lambda\mathbf{w}, \text{ where} \quad (4)\\ j' &= \arg\max_{0\le j<i}(\mathbf{w}\cdot\phi(j,i) + 1 - \mathcal{C}_d(i,j))\\ j'' &= \arg\max_{0\le j<i, C(i,j)=1}\mathbf{w}\cdot\phi(j,i)\end{aligned}$$

While SGD has no theoretically convergence guarantee, it works excellently in our experiments. Specifically, we observe that SGD achieves similar training performance to CCCP with a speed-up of around 10,000.

3.4 Incorporating Constraints

Next, we show how to incorporate domain knowledge-based constraints into L^3M and generalize it to CL^3M. In CL^3M, we obtain a clustering by maximizing a constraint-augmented scoring function f given by

$$s(\mathcal{C};d,\mathbf{w}) + \sum_{p=1}^{n_c}\rho_p\psi_p(d,\mathcal{C}),$$

where the second term on the R.H.S. is the score contributed by domain specific constraints $\psi_1,\ldots,\psi_{n_c}$ with their respective scores $\rho_1,\ldots,\rho_{n_c}$. In particular, $\psi_p(d,\mathcal{C})$ measures the extent to which a given clustering $\mathcal{C}$ satisfies the p^{th} constraint. Note that this framework is general and can be applied to inject mention-to-cluster or cluster-to-cluster level constraints too. However, for simplicity, we consider here only constraints between mention pairs. This allows us derive fast greedy algorithm to solve the inference problem. The details of our constraints are presented in Sec. 5.

All of our constraints can be categorized into two groups: "must-link" and "cannot-link"."Must-link" constraints encourage a pair of mentions to connect, while "cannot-link" constraints discourage mention pairs from being linked. Consequently, the coefficients ρ_p associated with "must-link" constraints are positive while ρ_p for "cannot-link" constraints are negative. In the following, we briefly discuss how to

solve the inference problem with these two types of constraints.

We slightly abuse notations and use $\psi_p(j,i)$ to indicate the p^{th} constraint on a pair of mentions (i,j). $\psi_p(j,i)$ is a binary function that is 1 *iff* two mentions i and j satisfy the conditions specified in constraint p. Chang et al. (2011) shows that best-left-link inference can be formulated as an ILP problem. When we add constraints, the ILP becomes:

$$\begin{aligned} \underset{B,C\in\{0,1\}}{\arg\max} \quad & \sum_{i,j:j<i} w_{ji}B_{ji} + \sum_{i,j} \rho_p \psi_p(j,i) C_{ij} \\ \text{s.t} \quad & C_{kj} \geq C_{ij} + C_{ki} - 1,\ \forall i,j,k, \\ & \sum_{j=0}^{i-1} B_{ji} = 1,\ \forall i \\ & B_{ji} < C_{ji}, C_{ji} = C_{ji}, \forall i,j, \end{aligned} \tag{5}$$

where $C_{ij} \equiv C(i,j)$ is a binary variable indicating whether i and j are in the same cluster or not and B_{ji} is an auxiliary variable indicating the best-left-link for mention i. The first set of inequality constraints in (5) enforces the transitive closure of the clustering. The constraints $B_{ji} < C_{ji}, \forall i,j$ enforce the consistency between these two sets of variables.

One can use an off-the-shelf solver to solve Eq. (5). However, when the absolute values of the constraint scores ($|\rho_p|$) are high (the hard constraint case), then the following greedy algorithm approximately solves the inference efficiently. We scan the document from left-to-right (or in any other arbitrary order). When processing mention i, we find

$$j^* = \underset{j<i}{\arg\max} \quad w_{ji} + \sum_{k:\hat{C}(k,j)=1} \sum_p \rho_p \psi_p(k,i), \tag{6}$$

where $\hat{C}$ is the current clustering obtained from the previous inference steps. Then, we add a link between mention i and j^*. The rest of the inference process is the same as in the original best-left-link inference. Specifically, this inference procedure combines the classifier score for mention pair i,j, with the constraints score of all mentions currently co-clustered with j. We discuss this further in Section 5.

4 Probabilistic Latent Left-Linking Model

In this section, we extend and generalize our left-linking model approach to a probabilistic model, Probabilistic Latent Left-Linking Model (PL^{3}M), that allows us to naturally consider mention-to-entity (or mention-to-cluster) links. While in L^{3}M, we assumed that each mention links deterministically to the max-scoring mention on its left, in PL^{3}M, we assume that mention i links to mention j, $j \leq i$, with probability given by

$$Pr[j \leftarrow i; d, \mathbf{w}] = \frac{e^{\frac{1}{\gamma}(\mathbf{w}\cdot\phi(i,j))}}{Z_i(\mathbf{w},\gamma)} \ . \tag{7}$$

Here $Z_i(\mathbf{w},\gamma) = \sum_{0\leq k<i} e^{\frac{1}{\gamma}(\mathbf{w}\cdot\phi(i,k))}$ is a normalizing constant and $\gamma \in (0,1]$ is a constant temperature parameter that is tuned on a development set (Samdani et al., 2012). We assume that the event that mention i links to a mention j is independent of the event that mention i' links to j' for $i \neq i'$.

Inference with PL^{3}M: Given the probability of a link as in Eq. (7), the probability that mention i joins an existing cluster c, $Pr[c \odot i; d, \mathbf{w}]$, is simply the sum of the probabilities of i linking to the mentions inside c:

$$\begin{aligned} Pr[c \odot i; d, \mathbf{w}] &= \sum_{j\in c, 0\leq j<i} Pr[j \leftarrow i; d, \mathbf{w}] \\ &= \sum_{j\in c, 0\leq j<i} \frac{e^{\frac{1}{\gamma}(\mathbf{w}\cdot\phi(i,j))}}{Z_i(d,\mathbf{w},\gamma)} \ . \end{aligned} \tag{8}$$

Based on Eq. (8) and making use of the independence assumption of left-links, we follow a simple greedy clustering (or inference) algorithm: sequentially add each mention i to a previously formed cluster c^*, where $c^* = \arg\max_c Pr[c \odot i; d, \mathbf{w}]$. If the $\arg\max$ cluster is the singleton cluster with the dummy mention 0 (i.e. the score of all other clusters is below the threshold of 0), then i starts a new cluster and is not included in the dummy cluster. Note that we link a mention to a cluster taking into account all the mentions inside that cluster, mimicking the notion of a mention-to-cluster link. This provides more expressiveness than the Best-Left-Link inference, where a mention connects to a cluster solely based on a single pairwise link to some antecedent mention (the best-link mention) in that cluster.

The case of $\gamma = 0$: As γ approaches zero, it is easy to show that the probability $P[j \leftarrow i; d, w]$

in Eq. (7) approaches a Kronecker delta function that puts probability 1 on the *max-scoring mention* $j = \arg\max_{0 \leq k < i} \mathbf{w} \cdot \phi(i, j)$ (assuming no ties), and 0 everywhere else (Pletscher et al., 2010; Samdani et al., 2012). Consequently, as $\gamma \to 0$, $Pr[c \odot i; d, \mathbf{w}]$ in Eq. 8 approaches a Kronecker delta function centered on the cluster containing the max-scoring mention, thus reducing to the best-link case of L^3M. Thus, PL^3M, when tuning the value of γ, is a strictly more general model than L^3M.

Learning with PL^3M We use a likelihood-based approach to learning with PL^3M, and first compute the probability $Pr[\mathcal{C}; d, \mathbf{w}]$ of generating a clustering $\mathcal{C}$, given $\mathbf{w}$. We then learn $\mathbf{w}$ by minimizing the regularized negative log-likelihood of the data, augmenting the partition function with a loss-based margin (Gimpel and Smith, 2010). We omit the details of likelihood computation due to lack of space.

With PL^3M, we again follow a stochastic gradient descent technique instead of CCCP for the same reasons mentioned in Sec. 3.3. The stochastic gradient (subgradient when $\gamma = 0$) w.r.t. mention i in document d is given by

$$\nabla LL(\mathbf{w})^i_d \propto \sum_{0 \leq j < i} p_j \phi(i, j) - \sum_{0 \leq j < i} p'_j \phi(i, j) + \lambda \mathbf{w},$$

where p_j and p'_j, $j = 0, \ldots, i-1$, are non-negative weights that sum to one and are given by

$$p_j = \frac{e^{\frac{1}{\gamma}(\mathbf{w} \cdot \phi(i,j) + \delta(\mathcal{C}_d, i, j))}}{\sum_{0 \leq k < i} e^{\frac{1}{\gamma}(\mathbf{w} \cdot \phi(i,k) + \delta(\mathcal{C}_d, i, k))}} \text{ and}$$

$$p'_j = \frac{\mathcal{C}_d(i, j) Z_i(d, \mathbf{w}, \gamma)}{Z_i(\mathcal{C}_d; d, \mathbf{w}, \gamma)} Pr[j \leftarrow i; d, \mathbf{w}] \ .$$

Interestingly, the above update rule generalizes the one for L^3M, as we are incorporating a weighted sum of all previous mentions in the update rule. With $\gamma \to 0$, the SGD in Eq. (4) converges to the SGD update in L^3M (Eq. (4)). Finally, in the presence of constraints, we can fold them inside the pairwise link probabilities as in Eq. (6).

5 Experiments and Results

In this section, we present our experiments on the two commonly used benchmarks for coreference — Ontonotes-5.0 (Pradhan et al., 2012) and ACE 2004 (NIST, 2004). Table 1 exhibits our bottom line results: CL^3M achieves the best result reported on Ontonotes-5.0 development set and essentially ties with (Fernandes et al., 2012) on the test set. As shown in Table 3, CL^3M is also the best algorithm on ACE and when evaluated on the gold mentions of Ontonotes. We show that CL^3M performs particularly well on clusters containing named entity mentions, which are more important for many information extraction applications. In the rest of this section, after describing our experimental setting, we provide careful analysis of our algorithms and compare them to competitive coreference approaches in the literature.

5.1 Experimental Setup

Datasets: ACE 2004 contains 443 documents — we used a standard split of these documents into 268 training, 68 development, and 106 testing documents used by Culotta et al. (2007) and Bengtson and Roth (2008). OntoNotes-5.0 dataset, released for the CoNLL 2012 Shared Task (Pradhan et al., 2012), is by far the largest annotated corpus on coreference. It contains 3,145 annotated documents drawn from a wide variety of sources — newswire, bible, broadcast transcripts, magazine articles, and web blogs. We report results on both development set and test set. To test on the development set, we further split the training data into training and development sets.

Classifier details: For each of the pairwise approaches, we assume the pairwise score is given by $\mathbf{w} \cdot \phi(\cdot, \cdot) + t$ where ϕ are the features, $\mathbf{w}$ is the weight vector learned by the approach, and t is a threshold which we set to 0 during learning (as in Eq. (1)), but use a tuned value (tuned on a development set) during testing. For learning with L^3M, we do stochastic gradient descent with 5 passes over the data. Empirically, we observe that this is enough to generate a stable model. For PL^3M (Sec. 4), we tune the value of γ using the development set picking the best γ from $\{0.0, 0.2, \ldots, 1.0\}$. Recall that when $\gamma = 0$, PL^3M is the same as L^3M. We refer to L^3M and PL^3M with incorporating constraints during inference as CL^3M and CPL^3M (Sec. 3.4), respectively.

Metrics: We compare the systems using three popular metrics for coreference — MUC (Vilain et al., 1995), BCUB (Bagga and Baldwin, 1998), and

Entity-based CEAF (CEAF$_e$) (Luo, 2005). Following, the CoNLL shared tasks (Pradhan et al., 2012), we use the average F1 scores of these three metrics as the main metric of comparison.

Features: We build our system on the publicly available Illinois-Coref system[1] primarily because it contains a rich set of features presented in Bengtson and Roth (2008) and Chang et al. (2012a) (the latter adds features for pronominal anaphora resolution). We also compare with the Best-Left-Link approach described by Bengtson and Roth (2008).

Constraints: We consider the following constraints in CL^{3}M and CPL^{3}M.

- SameSpan: two mentions must be linked to each other if they share the same surface text span and the number of words in the text span is larger than a threshold (set as 5 in our implementation).
- SameDetNom: two mentions must be linked to each other if both mentions start with a determiner and the [0,1] wordnet-based similarity score between the mention head words is above a threshold (set to 0.8).
- SameProperName: two mentions must be linked if they are both proper names and the similarity score measured by a named entity-based similarity metric, Illinois NESim[2], are higher than a threshold (set to 0.8). For a person entity we add additional rules to extract the first name, last name and professional title as properties.
- ModifierMismatch: the constraint prevents two mentions to be linked if the head modifiers conflict. For example, the constraint prevents "northern Taiwan" from linking to "southern Taiwan". We gather a list of mutual exclusive modifiers from the training data.
- PropertyMismatch: the constraint prevents two mentions to be linked if their properties conflict. For example, it prevents male pronouns to link to female pronouns and "Mr. Clinton" to link to "Mrs. Clinton" by checking the gender property. The properties we consider are gender, number, professional title and the nationality.

[1]The system is available at `http://cogcomp.cs.illinois.edu/page/software_view/Coref/`

[2]`http://cogcomp.cs.illinois.edu/page/software_view/NESim`

	MUC	BCUB	CEAF$_e$	AVG
Dev Set				
Stanford	64.30	70.46	46.35	60.37
(Chang et al., 2012a)	65.75	70.25	45.30	60.43
(Martschat et al., 2012)	66.76	71.91	47.52	62.06
(Björkelund and Farkas,)	67.12	71.18	46.84	61.71
(Chen and Ng, 2012)	66.4	71.8	**48.8**	62.3
(Fernandes et al., 2012)	**69.46**	71.93	48.66	63.35
L^{3}M	67.88	71.88	47.16	62.30
CL^{3}M	69.20	**72.89**	48.67	**63.59**
Test Set				
Stanford	63.83	68.52	45.36	59.23
(Chang et al., 2012a)	66.38	69.34	44.81	60.18
(Martschat et al., 2012)	66.97	70.36	46.60	61.31
(Björkelund and Farkas,)	67.58	70.26	45.87	61.24
(Chen and Ng, 2012)	63.7	69.0	46.4	59.7
(Fernandes et al., 2012)	**70.51**	71.24	**48.37**	**63.37**
L^{3}M	68.31	70.81	46.73	61.95
CL^{3}M	69.64	**71.93**	48.32	63.30

Table 1: Performance on OntoNotes-5.0 with predicted mentions. We report the F1 scores (%) on various coreference metrics (MUC, BCUB, CEAF). The column AVG shows the average scores of the three. We observe that PL^{3}M and CPL^{3}M (see Sec. 4) yields the same performance as L^{3}M and CL^{3}M, respectively as the tuned γ for all the datasets turned out to be 0.

While the "must-link" constraints described in the paper can be treated as features, due to their high precision, treating them as hard constraints (set ρ to a high value) is a safe and direct way to inject human knowledge into the learning model. Moreover, our framework allows a constraint to use information from previous decisions (such as "cannot-link" constraints). Treating such constraints as features will complicate the learning model.

5.2 Performance of the End-to-End System

We compare our system with the top systems reported in the CoNLL shared task 2012 as well as with the Stanford's publicly released rule-based system (Lee et al., 2013; Lee et al., 2011), which won the CoNLL 2011 Shared Task (Pradhan et al., 2011). Note that all the systems use the same annotations (e.g., gender prediction, part-of-speech tags, name entity tags) provided by the shared task organizers.

However, each system implements its own mention detector and pipelines the identified mentions into the coreference clustering component. Moreover, different systems use a different set of features. In order to partially control for errors on mention detection and better evaluate the clustering component in our coreference system, we will also present results on correct (gold) mentions in the next section.

Table 1 shows the end-to-end results. On the development set, only the best performing system of Fernandes et al. (2012) is better than L^{3}M, but this difference disappears when we use our system with constraints, CL^{3}M. Although our system is much simple, it achieves the best B^3 score on the test set and is competitive with the best system participated in the CoNLL shared task 2012.

Performance on named entities: The coreference annotation in Ontonotes 5.0 includes various types of mentions. However, not all mention types are equally interesting. In particular, clusters which contain at least one proper name or a named entity mention are more important for information extraction tasks like Wikification (Mihalcea and Csomai, 2007; Ratinov et al., 2011), cross-document coreference resolution (Bagga and Baldwin, 1998), and entity linking and knowledge based population (Ji and Grishman, 2011).

Inspired by this, we compare our system to the best systems in the CoNLL shared task of 2011 (Stanford (Lee et al., 2011)) and 2012 (Fernandes (Fernandes et al., 2012)) on the following specific tasks on Ontonotes-5.0.

- **ENT-C**: Evaluate the system on clusters that contain at least one proper name mention. We generate the gold annotation and system outputs by using the gold and predicted name entity tag annotations provided by the CoNLL shard task 2012. That is, if a cluster does not include any name entity mention, then it will be removed from the final clustering.
- **PER-C**: As in the construction of ENT-C, but here we only consider clusters which contain at least one "Person (PER)" entity.
- **ORG-C**: As in the construction of Entity-C, but here we only consider clusters which contain at least one "Organization (ORG)" entity.

Typically, the clusters that get ignored in the above definitions contain only first and second person pronouns (which often happens in transcribed discourse.) Also note that all the systems are trained with the same name entity tags, provided by the shared task organizers, and we use the same name entity tags to construct the specific clustering. Also, in order to further ensure fairness, we do not tune our system to favor the evaluation of these specific types of clusters. We chose to do so because we only have access to the system output of Fernandes et al. (2012).

Task	Stanford	Fernandes	L^{3}M	CL^{3}M
ENT-C	44.06	47.05	46.63	**48.02**
PER-C	34.04	36.43	37.01	**37.57**
ORG-C	25.02	26.23	26.22	**27.01**

Table 2: Performance on named entities for OntoNotes-5.0 data. We compare our system to Fernandes (Fernandes et al., 2012) and Stanford (Lee et al., 2013) systems.

Table 2 shows the results. The performance of all systems degrades when considering only clusters that contain name entities, indicating that ENT-C is actually a harder task than the original coreference resolution problem. In particular, resolving ORG coreferent clusters is hard, because names of organizations are sometimes confused with person names, and they can be referred to using a range of pronouns (including "we" and "it"). Overall, CL^{3}M outperforms all the competing systems on the clusters that contain at least one specific type of entity by a margin larger than that for the overall coreference.

5.3 Analysis on Gold Mentions

To better understand the contribution of our joint learning and clustering model, we present experiments assuming that gold mentions are given. The definitions of gold mentions in ACE and Ontonotes are different because Ontonotes-5.0 excludes singleton clusters in the annotation. In addition, Ontonotes includes longer mentions; for example, it includes NP and appositives in the same mention. We compare with the publicly available Stanford (Lee et al., 2011) and IllinoisCoref (Chang et al., 2012a) systems; the system of Fernandes et al. (2012) is not publicly available. In addition, we also compare with the following two structured prediction baselines that use the same set of features as L^{3}M and PL^{3}M.

	MUC	BCUB	CEAF$_e$	AVG
ACE 2004 Gold Ment.				
All-Link-Red.	77.45	81.10	77.57	78.71
Spanning	73.31	79.25	74.66	75.74
IllinoisCoref	76.02	81.04	77.6	78.22
Stanford	75.04	80.45	76.75	77.41
(Stoyanov and Eisner, 2012)	**80.1**	81.8	-	-
L^{3}M	77.57	81.77	78.15	79.16
PL^{3}M	78.18	82.09	79.21	79.83
CL^{3}M	78.17	81.64	78.45	79.42
CPL^{3}M	78.29	**82.20**	**79.26**	**79.91**
Ontonotes 5.0 Gold Ment.				
All-Link-Red.	83.72	75.59	64.00	74.44
Spanning	83.64	74.83	61.07	73.18
IllinoisCoref	80.84	74.29	65.96	73.70
Stanford	82.26	76.82	61.69	73.59
L^{3}M	83.44	78.12	64.56	75.37
PL^{3}M	83.97	78.25	65.69	75.97
CL^{3}M	84.10	78.30	68.74	77.05
CPL^{3}M	**84.80**	**78.74**	**68.75**	**77.43**

Table 3: Performance on ACE 2004 and OntoNotes-5.0. All-Link-Red. is based on correlational clustering; Spanning is based on latent spanning forest based clustering (see Sec. 2). Our proposed approach is L^{3}M (Sec. 3) and PL^{3}M (sec. 4). CL^{3}M and CPL^{3}M are the version with incorporating constraints.

1. **All-Link-Red:** a reduced and faster alternative to the correlational clustering based approach (Finley and Joachims, 2005). We implemented this algorithm as an ILP and droped one of the three transitivity constraints for each triplet of mention variables. Following Pascal and Baldridge (2009) and Chang et al. (2011) we observe that this slightly improves the accuracy over a pure correlation clustering approach, in addition to speeding up inference.
2. **Spanning:** the latent spanning forest based approach presented by Yu and Joachims (2009). We use the publicly available implementation provided by the authors[3] for the ACE data; since their CCCP implementation is slow, we implemented our own stochastic gradient descent version to scale it to the much larger Ontonotes data.

[3]Available at http://www.cs.cornell.edu/ cnyu/latentssvm/

Table 3 lists the results. Although L^{3}M is simple and use only the features defined on pairwise mentions, it compares favorably with all recently published results. Moreover, the probabilistic generalization of L^{3}M, PL^{3}M, achieves even better performance. For example, L^{3}M with $\gamma = 0.2$ improves L^{3}M with $\gamma = 0$ by 0.7 points in ACE 2004. In particular, This shows that considering more than a one left-links is helpful. This is in contrast with the predicted mentions where $\gamma = 0$ performed best. We suspect that this is because noisy mentions can hurt the performance of PL^{3}M that takes into account not just the best scoring links, but also weaker links which are likely to be less reliable (more false positives). Also, as opposed to what is reported by Yu and Joachims (2009), the correlation clustering approach performs better than the spanning forest approach. We think that this is because we compare the systems on different metrics than they did and also because we use exact ILP inference for correlational clustering whereas Yu and Joachims (2009) used approximate greedy inference.

Both L^{3}M and PL^{3}M can be benefit from using constraints. However, The constraints improve only marginally on the ACE 2004 data because ACE uses shorter phrases as mentions. Consequently, constraints designed for leveraging information from long mention spans are less effective. Overall, the experiments show that L^{3}M and PL^{3}M perform well on modeling coreference clustering.

5.4 Ablation Study of Constrains

Finally, we study the value of individual constraints by adding one constraint at a time to the coreference system starting with the simple L^{3}M model. The system with all the constraints added is the CL^{3}M model introduced in Table 1. We then remove individual constraints from CL^{3}M to assess its contribution. Table 4 shows the results on the Ontonotes dataset with predicted mentions. Overall, it is shown that each one of the constraints has a contribution, and that using all the constraints improves the performance of the system by 1.29% in the AVG F1 score. In particular, most of this improvement (1.19%) is due to the must-link constraints (the first four constraints in the table). The must-link constraints are more useful for L^{3}M as L^{3}M achieves higher precision than recall (e.g., the precision and

	MUC	BCUB	$CEAF_e$	AVG
L^3M	67.88	71.88	47.16	62.30
+SameSpan	68.27	72.27	47.73	62.75
+SameDetNom	68.79	72.57	48.30	63.22
+SameProperName	69.11	72.81	48.56	63.49
+ModifierMismatch	69.11	72.81	48.58	63.50
+PropertyMismatch (i.e. CL^3M)	69.20	72.89	48.67	63.59
-SameSpan	68.91	72.66	48.36	63.31
-SameDetNom	68.62	72.51	48.06	63.06
-SameProperName	68.97	72.69	48.50	63.39
-ModifierMismatch	69.12	72.80	48.63	63.52
-PropertyMismatch	69.11	72.81	48.58	63.50

Table 4: Ablation study on constraints. We first show cumulative performance on OntoNotes-5.0 data with predicted mentions as constraints are added one at a time into the coreference system. Then we demonstrate the value of individual constraints by leaving out one constraint at each time.

recall of L^3M are 78.38% and 67.96%, respectively in B^3). As a result, the must-link constraints, which aim at improving the recall, do better when optimizing F1.

6 Conclusions

We presented a principled yet simple framework for coreference resolution. Furthermore, we showed that our model can be augmented in a straightforward way with knowledge based constraints, to improve performance. We also presented a probabilistic generalization of this model that can take into account entity-mention links by considering multiple possible coreference links. We proposed a fast stochastic gradient-based learning technique for our model. Our model, while operating at mention pair granularity, obtains state-of-the-art results on OntoNotes-5.0, and performs especially well on mention clusters containing named entities. We provided a detailed analysis of our experimental results.

Acknowledgments Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20155. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*.

M. Bansal and D. Klein. 2012. Coreference semantics from web features. In *Proceedings of ACL*, Jeju Island, South Korea, July.

N. Bansal, A. Blum, and S. Chawla. 2002. Correlation clustering. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*.

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *EMNLP*, 10.

A. Björkelund and R. Farkas.

K.-W. Chang, R. Samdani, A. Rozovskaya, N. Rizzolo, M. Sammons, and D. Roth. 2011. Inference protocols for coreference resolution. In *CoNLL Shared Task*.

K.-W. Chang, R. Samdani, A. Rozovskaya, M. Sammons, and D. Roth. 2012a. Illinois-coref: The UI system in the CoNLL-2012 Shared Task. In *CoNLL Shared Task*.

M. Chang, L. Ratinov, and D. Roth. 2012b. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

C. Chen and V. Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*.

J. Clarke and M. Lapata. 2006. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151, Sydney, Australia, July. ACL.

A. Culotta, M. Wick, R. Hall, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *HLT/NAACL*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

P. Denis and J. Baldridge. 2008. Specialized models and ranking for coreference resolution. In *EMNLP*, pages 660–669.

G. Durrett, D. Hall, and D. Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Proceedings of ACL*, August.

E. R. Fernandes, C. N. dos Santos, and R. L. Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*.

T. Finley and T. Joachims. 2005. Supervised clustering with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*.

K. Gimpel and N. A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *NAACL*.

A. Haghighi and D. Klein. 2010. Coreference resolution in a modular, entity-centered model. In *NAACL*.

H. Ji and R. Grishman. 2011. Knowledge base population: successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *EMNLP*.

H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the CoNLL-2011 Shared Task*.

H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

X. Luo. 2005. On coreference resolution performance metrics. In *EMNLP*.

S. Martschat, J. Cai, S. Broscheit, É. Mújdricza-Maydt, and M. Strube. 2012. A multigraph model for coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*, July.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

R. Mihalcea and A. Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*.

Vincent Ng. 2005. Supervised ranking for pronoun resolution: Some recent improvements. In *AAAI*, pages 1081–1086.

NIST. 2004. The ACE evaluation plan.

D. Pascal and J. Baldridge. 2009. Global joint models for coreference resolution and named entity classification. In *Procesamiento del Lenguaje Natural*.

P. Pletscher, C. S. Ong, and J. M. Buhmann. 2010. Entropy and margin maximization for structured output learning. In *ECML PKDD*.

S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *CoNLL*.

S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *CoNLL 2012*.

M. Sammons Y. Tu V. Vydiswaran Q. Do, D. Roth. 2009. Robust, light-weight approaches to compute lexical similarity. Technical report.

K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning. 2010. A multi-pass sieve for coreference resolution. In *EMNLP*.

A. Rahman and V. Ng. 2011a. Coreference resolution with world knowledge. In *ACL*, pages 814–824.

A. Rahman and V. Ng. 2011b. Ensemble-based coreference resolution. In *IJCAI*.

A. Rahman and V. Ng. 2011c. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *JAIR*.

W.M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

L. Ratinov and D. Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *EMNLP*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Dan Roth and Wen Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL-04*, pages 1–8.

R. Samdani, M. Chang, and D. Roth. 2012. Unified expectation maximization. In *NAACL*.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*

V. Stoyanov and J. Eisner. 2012. Easy-first coreference resolution. In *COLING*, pages 2519–2534.

V. Stoyanov, N. Gilbert, C. Cardie, and E. Riloff. 2009. Conundrums in noun phrase coreference resolution: making sense of the state-of-the-art. In *ACL*.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference

scoring scheme. In *Proceedings of the 6th conference on Message understanding*.

C. Yu and T. Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the International Conference on Machine Learning (ICML)*.

A. L. Yuille and A. Rangarajan. 2003. The concave-convex procedure. *Neural Computation*, 15(4).

Centering Similarity Measures to Reduce Hubs

Ikumi Suzuki
National Institute of Genetics
Mishima, Shizuoka, Japan
suzuki.ikumi@gmail.com

Kazuo Hara
National Institute of Genetics
Mishima, Shizuoka, Japan
kazuo.hara@gmail.com

Masashi Shimbo
Nara Institute of Science and Technology
Ikoma, Nara, Japan
shimbo@is.naist.jp

Marco Saerens
Université catholique de Louvain
Louvain-la-Neuve, Belgium
marco.saerens@uclouvain.be

Kenji Fukumizu
The Institute of Statistical Mathematics
Tachikawa, Tokyo, Japan
fukumizu@ism.ac.jp

Abstract

The performance of nearest neighbor methods is degraded by the presence of *hubs*, i.e., objects in the dataset that are similar to many other objects. In this paper, we show that the classical method of *centering*, the transformation that shifts the origin of the space to the data centroid, provides an effective way to reduce hubs. We show analytically why hubs emerge and why they are suppressed by centering, under a simple probabilistic model of data. To further reduce hubs, we also move the origin more aggressively towards hubs, through weighted centering. Our experimental results show that (weighted) centering is effective for natural language data; it improves the performance of the k-nearest neighbor classifiers considerably in word sense disambiguation and document classification tasks.

1 Introduction

1.1 Background

The k-nearest neighbor (kNN) algorithm is a simple nonparametric method of classification. It has been applied to various natural language processing (NLP) tasks such as document classification (Masand et al., 1992; Yang and Liu, 1999), part-of-speech tagging (Søgaard, 2011), and word sense disambiguation (Navigli, 2009).

To apply the kNN algorithm, data is typically represented as a vector object in a feature space, and (dis)similarity between data is measured by the distance between the vectors, their inner product, or cosine of the angle between them (Jurafsky and Martin, 2008). With such a (dis)similarity measure, the unknown class label of a test object is predicted by a majority vote of the classes of its k most similar objects in the labeled training set.

Recent studies (Radovanović et al., 2010a; Radovanović et al., 2010b) have shown that if the feature space is high-dimensional, some objects in the dataset emerge as *hubs*; i.e., these objects frequently appear in the k nearest neighbors of other objects.

The emergence of hubs may deteriorate the performance of kNN classification and nearest neighbor search in general:

- If hub objects exist in the training set, they have a strong chance to be a kNN of many test objects. Because the class of a test object is predicted by a majority vote from its k nearest neighbors, prediction is biased toward the labels of the hubs.
- In information retrieval, nearest neighbor search finds objects in the database that are most relevant, or similar, to user-provided queries. If particular objects, such as hubs, are nearly always returned for any query, the retrieved results are probably not very useful.

These drawbacks may hinder application of nearest neighbor methods in NLP, as typical natural language data are extremely high-dimensional (Jurafsky and Martin, 2008) and thus prone to produce hubs.

1.2 Contributions

Centering (Mardia et al., 1979; Fisher and Lenz, 1996; Eriksson et al., 2006) is a standard technique

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 613–623,
Seattle, Washington, USA, 18-21 October 2013.

for removing observation bias in the data. It is a transformation of feature space in a way that the origin of the space is moved to the data centroid (sample mean). The distance between data objects is not changed by centering, but their inner product and cosine are affected; see Section 3 for detail.

In this paper, we advocate the use of centering as a means of reducing hubs. Specifically, we propose to measure the similarity of objects by the inner product (not distance or cosine) in the centered feature space.

Our approach is motivated by the observation that the objects similar to the data centroid tend to become hubs (Radovanović et al., 2010a). This observation suggests that the number of hubs may be reduced if we can define a similarity measure that makes all objects in a dataset equally similar to the centroid (Suzuki et al., 2012). The inner product in the centered space indeed enjoys this property.

In Section 4, we analyze why hubs emerge under a simple probabilistic model of data, and also give an account of why they are suppressed by centering.

Using both synthetic and real datasets, we show that objects similar to the centroid also emerge as hubs in multi-cluster data (Section 5), so the application of centering is wider than expected. To further reduce hubs, we also propose to move the origin of the space more aggressively towards hubs, through *weighted* centering (Section 6).

In Section 7, we show that centering and weighted centering are effective for natural language data. these methods markedly improve the performance of kNN classifiers in word sense disambiguation and document classification tasks.

2 Related work

Centering is a classical technique widely used in many fields of science. For instance, centering forms a preprocessing step in principal component analysis and Fisher linear discriminant analysis.

In NLP, however, centering is seldom used; the use of cosine and inner product similarities is quite common, but they are nearly always used uncentered. Non-centered cosine is used, for instance, in word sense disambiguation (Schütze, 1998; Navigli, 2009), paraphrasing (Erk and Padó, 2008; Thater et al., 2010), and compositional semantics (Mitchell and Lapata, 2008), to name a few.

There have been several approaches to improving kNN classification: learning similarity/distance measures from training data (metric learning) (Weinberger and Saul, 2009; Qamar et al., 2008), weighting nearest neighbors for similarity-based classification (Chen et al., 2009), and neighborhood size selection (Wang et al., 2006; Guo and Chakraborty, 2010). However, none of these have addressed the reduction of hubs.

More recently, Schnitzer et al. (2012) proposed the *Mutual Proximity* transformation that rescales distance measures to decrease hubs in a dataset. Suzuki et al. (2012) showed that kernels based on graph Laplacian, such as the commute-time kernels (Saerens et al., 2004) and the regularized Laplacian (Chebotarev and Shamis, 1997; Smola and Kondor, 2003), make all objects equally similar to the data centroid, which in turn reduce hubs.

In Section 7, we evaluate centering, Mutual Proximity, and Laplacian kernels in NLP tasks, and demonstrate that centering is equally or even more effective. Section 4 presents a theoretical justification for using centering to reduce hubs, but this kind of analysis is missing for the Laplacian kernels.

Centering is easier to compute as well. For a dataset of n objects, it takes $O(n^2)$ time to compute, whereas computing a Laplacian-based kernel requires $O(n^3)$ time for matrix inversion. Mutual Proximity also has a time complexity of $O(n^2)$.

3 Centering

Consider a dataset of n objects in an m-dimensional feature space, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^m$. Throughout this paper, we use the inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ as a measure of similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Let $\mathbf{K}$ be the Gram matrix of the n feature vectors, i.e., the $n \times n$ matrix whose (i, j) element holds $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. Using $m \times n$ data matrix $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$, we can write $\mathbf{K}$ as

$$\mathbf{K} = \mathbf{X}^{\mathrm{T}}\mathbf{X},$$

where $\mathbf{X}^{\mathrm{T}}$ represents the matrix transpose of $\mathbf{X}$.

Centering is a transformation in which the origin of the feature space is shifted to the data centroid

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i, \tag{1}$$

and object $\boldsymbol{x}$ is mapped to the centered feature vector

$$\boldsymbol{x}^{\mathsf{cent}} = \boldsymbol{x} - \bar{\boldsymbol{x}}. \tag{2}$$

The similarity between two objects $\boldsymbol{x}$ and $\boldsymbol{x}'$ is now measured by $\langle \boldsymbol{x}^{\mathsf{cent}}, \boldsymbol{x}'^{\mathsf{cent}} \rangle = \langle \boldsymbol{x} - \bar{\boldsymbol{x}}, \boldsymbol{x}' - \bar{\boldsymbol{x}} \rangle$.

After centering, the inner product between any object and the data centroid (which is a zero vector because $\bar{\boldsymbol{x}}^{\mathsf{cent}} = \bar{\boldsymbol{x}} - \bar{\boldsymbol{x}} = \mathbf{0}$) is uniformly 0; in other words, all objects in the dataset have an equal similarity to the centroid. According to the observation that the objects similar to the centroid become hubs (Radovanović et al., 2010a), we can expect hubs to be reduced after centering.

Intuitively, centering reduces hubs because it makes the length of the feature vector $\boldsymbol{x}^{\mathsf{cent}}$ short for (hub) objects $\boldsymbol{x}$ that lie close to the data centroid $\bar{\boldsymbol{x}}$; see Eq. (2). And since we measure object similarity by inner product, shorter vectors tend to produce smaller similarity scores. Hence objects close to the data centroid become less similar to other objects after centering, and no longer be hubs. In Section 4, we analyze the effect of centering on hubness in more detail.

3.1 Centered Gram matrix

Let $\mathbf{I}$ be an $n \times n$ identity matrix and $\mathbb{1}$ be an n-dimensional all-ones vector. The symmetric matrix $\mathbf{H} = \mathbf{I} - (1/n)\mathbb{1}\mathbb{1}^{\mathrm{T}}$ is called *centering matrix*, because the centered data matrix $\mathbf{X}^{\mathsf{cent}} = [\boldsymbol{x}_1^{\mathsf{cent}}, \cdots, \boldsymbol{x}_n^{\mathsf{cent}}]$ can be computed by $\mathbf{X}^{\mathsf{cent}} = \mathbf{XH}$ (Mardia et al., 1979).

The Gram matrix $\mathbf{K}^{\mathsf{cent}}$ of the centered feature vectors, whose (i, j) element holds the inner product $\langle \boldsymbol{x}_i^{\mathsf{cent}}, \boldsymbol{x}_j^{\mathsf{cent}} \rangle$, can be calculated from the original Gram matrix $\mathbf{K}$ by

$$\mathbf{K}^{\mathsf{cent}} = \left(\mathbf{X}^{\mathsf{cent}}\right)^{\mathrm{T}} \left(\mathbf{X}^{\mathsf{cent}}\right) = \mathbf{H}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{H} = \mathbf{HKH}. \tag{3}$$

Eq. (3) implies that the original data matrix $\mathbf{X}$ is not needed to compute the centered Gram matrix $\mathbf{K}^{\mathsf{cent}}$, provided that $\mathbf{K}$ is given. It is hence possible to use the so-called *kernel trick*; i.e., centering can be applied even if data matrix $\mathbf{X}$ is not available but the similarity of objects can be measured by a kernel function in an implicit feature space.

4 Theoretical analysis of the effect of centering on hubness

We now analyze why objects most similar to the centroid tend to be hubs in the dataset, and give an explanation as to why centering may suppress the emergence of hubs.

4.1 Before centering

Consider a dataset of m-dimensional feature vectors, with each vector $\boldsymbol{x} \in \mathbb{R}^m$ generated independently from a distribution with a finite mean vector $\boldsymbol{\mu}$. In other words, objects $\boldsymbol{x}$ in this dataset are drawn from a distribution $P(\boldsymbol{x})$, i.e.,

$$\boldsymbol{x} \sim P(\boldsymbol{x}),$$

and

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}] = \int \boldsymbol{x}\, dP(\boldsymbol{x}) \tag{4}$$

where $\mathbb{E}[\cdot]$ denotes the expectation of a random variable.

We will use the following elementary lemma on the distributions of inner product subsequently.

Lemma 1. *Let $\boldsymbol{a} \in \mathbb{R}^m$ be a fixed vector, and $\boldsymbol{x} \in \mathbb{R}^m$ be an object sampled according to distribution $P(\boldsymbol{x})$. Then the inner product $\langle \boldsymbol{a}, \boldsymbol{x} \rangle$ follows a distribution with mean $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$.*

Proof. From the linearity of the inner product and Eq. (4), we obtain

$$\begin{aligned} \mathbb{E}[\langle \boldsymbol{a}, \boldsymbol{x} \rangle] &= \int \langle \boldsymbol{a}, \boldsymbol{x} \rangle\, dP(\boldsymbol{x}) \\ &= \langle \boldsymbol{a}, \int \boldsymbol{x}\, dP(\boldsymbol{x}) \rangle = \langle \boldsymbol{a}, \boldsymbol{\mu} \rangle. \end{aligned}$$

□

Now, imagine that we have an object $\boldsymbol{x}$ sampled from $P(\boldsymbol{x})$, and we want to compute its nearest neighbor in a dataset. Let $\boldsymbol{h}$ and $\boldsymbol{\ell}$ be two fixed objects in the dataset, such that the inner product to the true mean $\boldsymbol{\mu}$ is higher for $\boldsymbol{h}$ than for $\boldsymbol{\ell}$, i.e.,

$$\langle \boldsymbol{h}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{\ell}, \boldsymbol{\mu} \rangle > 0. \tag{5}$$

We are interested in which of $\boldsymbol{h}$ and $\boldsymbol{\ell}$ is more similar to $\boldsymbol{x}$ (in terms of inner product), or in other words, the difference of two inner products

$$z = \langle \boldsymbol{h}, \boldsymbol{x} \rangle - \langle \boldsymbol{\ell}, \boldsymbol{x} \rangle = \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{x} \rangle. \tag{6}$$

Because $\boldsymbol{x}$ is a random variable, so is z. Let $Q(z)$ be the distribution of z; i.e., $z \sim Q(z)$.

Using Lemma 1 with $\boldsymbol{a} = \boldsymbol{h} - \boldsymbol{\ell}$, together with Eq. (5), we have

$$\mathbb{E}[z] = \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{\mu} \rangle = \langle \boldsymbol{h}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{\ell}, \boldsymbol{\mu} \rangle > 0. \quad (7)$$

Note that the above statement is only concerned about the mean, so it does not in general assure that

$$\langle \boldsymbol{h}, \boldsymbol{x} \rangle > \langle \boldsymbol{\ell}, \boldsymbol{x} \rangle \quad (8)$$

holds with high probability; there is a chance that a small number of outliers are inflating the mean. To assure that inequality (8) holds with probability greater than $1/2$ for instance, the median rather than the mean of the distribution $Q(z)$ must be greater than 0.

If the distribution $Q(z)$ is symmetric, the median occurs at the same point as the mean, and the above claim holds. Indeed, if the components of $\boldsymbol{x}$ are generated independently from (possibly non-identical) normal distributions, we can show that $Q(z)$ also obeys a normal distribution. Because it is a symmetric distribution, we can safely say that in this case, Eq. (8) holds with probability greater than $1/2$.

For a general non-symmetric distribution with a finite variance, the median is known to be within the standard deviation of the mean (Mallows, 1991), so we could still say that Eq. (8) is likely to hold if $\langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{\mu} \rangle$ is sufficiently large compared to the standard deviation.

Now, if we let $\boldsymbol{h}$ be the object in a given dataset with the highest similarity (inner product) to the mean $\boldsymbol{\mu}$, and let $\boldsymbol{\ell}$ be any other object in the set, then we see from the above discussion that $\boldsymbol{h}$ is likely to have higher similarity to $\boldsymbol{x}$, a test sample drawn from distribution $P(\boldsymbol{x})$. Because this holds for any $\boldsymbol{\ell}$ in the dataset, the conclusion is that the objects in the dataset most similar to $\boldsymbol{\mu}$ are likely to become hubs.

4.2 After centering

Next let us investigate what happens if the dataset is centered. Let $\bar{\boldsymbol{x}}$ be the sample (empirical) mean given by Eq. (1). After centering, the similarity of $\boldsymbol{x}$ with each of the two fixed objects $\boldsymbol{h}$ and $\boldsymbol{\ell}$ are evaluated by $\langle \boldsymbol{h} - \bar{\boldsymbol{x}}, \boldsymbol{x} - \bar{\boldsymbol{x}} \rangle$ and $\langle \boldsymbol{\ell} - \bar{\boldsymbol{x}}, \boldsymbol{x} - \bar{\boldsymbol{x}} \rangle$, respectively. Their difference z^{cent} is given by

$$\begin{aligned} z^{\mathsf{cent}} &= \langle \boldsymbol{h} - \bar{\boldsymbol{x}}, \boldsymbol{x} - \bar{\boldsymbol{x}} \rangle - \langle \boldsymbol{\ell} - \bar{\boldsymbol{x}}, \boldsymbol{x} - \bar{\boldsymbol{x}} \rangle \\ &= \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{x} - \bar{\boldsymbol{x}} \rangle \\ &= \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{x} \rangle - \langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle \\ &= z - \langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle. \end{aligned}$$

The last equality follows from Eq. (6). By definition we have $z \sim Q(z)$, and since $\langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle$ is a constant,

$$z^{\mathsf{cent}} = z - \langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle \sim Q(z + \langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle).$$

In other words, the shape of the distribution does not change, but the mean is shifted to

$$\begin{aligned} \mathbb{E}[z^{\mathsf{cent}}] &= \mathbb{E}[z] - \langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle \\ &= \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{h} - \boldsymbol{\ell}, \bar{\boldsymbol{x}} \rangle \\ &= \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{\mu} - \bar{\boldsymbol{x}} \rangle, \end{aligned}$$

where $\mathbb{E}[z]$ is given by Eq. (7). If the sample mean $\bar{\boldsymbol{x}}$ is close enough to the true mean $\boldsymbol{\mu}$, i.e., $\bar{\boldsymbol{x}} \approx \boldsymbol{\mu}$, we have an approximation

$$\mathbb{E}[z^{\mathsf{cent}}] = \langle \boldsymbol{h} - \boldsymbol{\ell}, \boldsymbol{\mu} - \bar{\boldsymbol{x}} \rangle \approx 0. \quad (9)$$

Thus, if the median and the mean of distribution $Q(z)$ are again not far apart, Eq. (9) suggests that $\boldsymbol{h} - \bar{\boldsymbol{x}}$ and $\boldsymbol{\ell} - \bar{\boldsymbol{x}}$ are about equally likely to be more similar to $\boldsymbol{x} - \bar{\boldsymbol{x}}$; i.e., neither has a greater chance to become a hub.

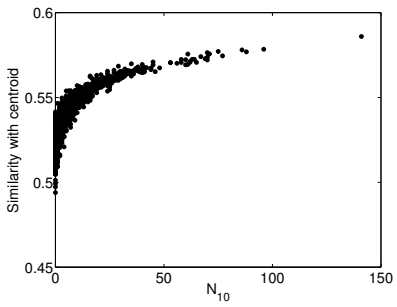
5 Hubs in multi-cluster data

In this section, we discuss emergence of hubs when the data consists of multiple clusters. In fact, the analysis of Section 4 is distribution-free, and thus also applies to the case of multi-modal $P(\boldsymbol{x})$. However, one might still argue that objects similar to the data centroid should hardly occur in that case. Using both synthetic and real datasets, we demonstrate below that even in multi-cluster data, objects that are only slightly more similar to the data mean (centroid) may emerge as hubs.

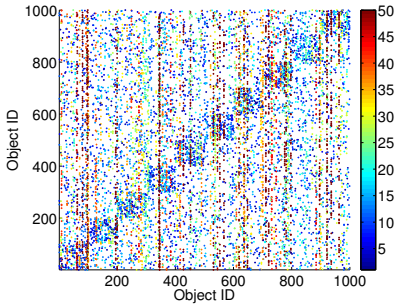
5.1 Synthetic data

5.1.1 Data generation

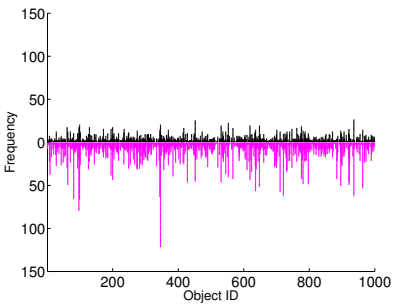
We generated a high-dimensional multi-cluster dataset by modeling it as a mixture of ten von Mises-Fisher distributions (Mardia and Jupp, 2000) in



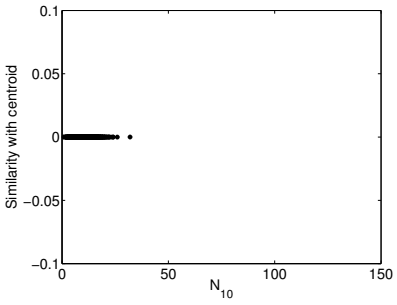
(a) Before centering: N_{10} vs. inner product similarity to the data centroid



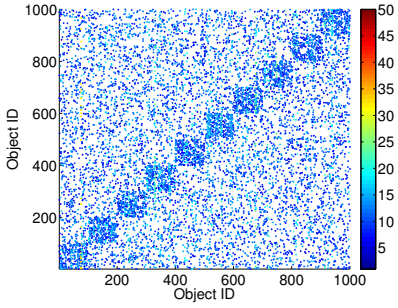
(b) Before centering: kNN matrix



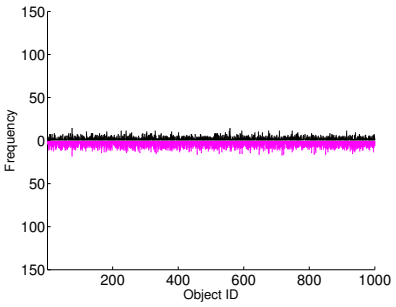
(c) Before centering: Breakdown of N_{10} by cluster match/mismatch between objects and neighbors



(d) After centering: N_{10} vs. inner product similarity to the data centroid



(e) After centering: kNN matrix



(f) After centering: Breakdown of N_{10} by cluster match/mismatch between objects and neighbors

Figure 1: 300-dimensional synthetic data. (a), (d): scatter plot of the N_{10} value of objects and their similarity to centroid. (b), (e): kNN matrices. The points are colored according to the N_{10} value of object x; warmer colors indicate higher N_{10} values. (c), (f): the number of times (y-axis) an object (whose ID is on the x-axis) appears in the 10 nearest neighbors of objects of the same cluster (black bars), and those of different clusters (magenta).

$\mathbb{R}^{300}$. The von Mises-Fisher distribution is a distribution of unit vectors (it can roughly be thought of as a normal distribution on a unit hypersphere), so for objects (feature vectors) sampled from this distribution, inner product reduces to cosine similarity.

We sampled[1] 100 objects from each of the ten distributions (clusters), and made a dataset of 1,000 objects in total.

The von Mises-Fisher distribution has two parameters, the mean direction vector $\boldsymbol{\mu}$, and the concentration parameter κ characterizing how strongly the population is concentrated around the direction $\boldsymbol{\mu}$. We set $\kappa = 500$ for all ten distributions, but the mean directions $\boldsymbol{\mu}$ were made distinct; all mean direction vectors had 30 components set to 0.5 while the remaining 270 components were set to 1, but the 30 components with value 0.5 were chosen to be distinct among the ten clusters. This configuration assures that all ten mean directions have the same angle from the all-ones vector $[1, \ldots, 1]^{\mathrm{T}}$, which is the direction of the mean of the entire data distribution.

Note that even though all sampled objects reside on the surface of the unit hypersphere, the data centroid lies not on the surface but inside the hypersphere. And after centering, the length of the feature vectors may vary from one another, but we do not normalize these vectors; i.e., object similarity is measured by raw inner product, not by cosine.

[1]We used the random sampling code available at http://people.kyb.tuebingen.mpg.de/suvrit/work/progs/movmf.html (Banerjee et al., 2005).

5.1.2 Correlation between hubness and centroid similarity

The scatter plot in Figure 1(a) shows the correlation between the degree of hubness (N_{10}) of an object and its inner product similarity to the data centroid. The N_{10} value of an object is defined as the number of times the object appears in the 10 nearest neighbors of other objects in the dataset. It was used in (Radovanović et al., 2010a) to measure the degree of hubness of individual objects.

The plot clearly shows that the hub objects (i.e., those with high N_{10}) consist of objects that are similar to the centroid. Figure 1(d) shows the scatter plot after the data is centered, created in the same way as Figure 1(a). The similarity to the centroid is uniformly 0 as a result of centering, and no objects have an N_{10} value greater than 33.

5.1.3 Influence of hubs on objects in different clusters

The kNN matrix of Figure 1(b) depicts the kNN relations with $k = 10$ among objects before centering. In this matrix, both the x- and y- axes represent the ID of the objects. If object x is in the 10 nearest neighbors of object y, a point is plotted at coordinates (x, y). As a result, there are exactly $k = 10$ points in each row. The color of points indicates the degree of hubness of object x; warmer color represents higher N_{10} value of the object.

In this matrix, object IDs are sorted by the cluster the objects belong to. Hence in the ideal case in which the k nearest neighbors of every object consist genuinely of objects from the same cluster, only the diagonal blocks would be colored, and off-diagonal areas would be left blank.

As Figure 1(b) shows, the actual situation is far from ideal, even though ten diagonal blocks are still identifiable. The presence of many warm colored vertical lines suggests that many hub objects appear in the 10 nearest neighbors of other objects that are not in the same cluster as the hubs. Thus these hubs may have a strong influence on the kNN prediction of other objects.

Figure 1(e) shows the kNN matrix after centering. The warm colored lines have disappeared, and the diagonal blocks are now more visible.

The bar graphs of Figures 1(c) and (f) plot the N_{10} value of each object (whose ID is on the x-axis). Recall that N_{10} is the number of times an object appears in the 10 nearest neighbors of other objects. The bar for each object is broken down by whether the object and its neighbors belong to the same cluster (black bar) or in different clusters (magenta bar). In terms of kNN classification, having a large number of nearest neighbors with the same class improves the classification performance, so longer black bars and shorter magenta bars are more desirable.

Before centering (Figure 1(c)), hub objects with large N_{10} values are similar not only to objects belonging to the same cluster (as indicated by black bars), but also to objects belonging to different clusters (magenta bars). After centering (Figure 1(f)), the number of tall magenta bars decreases.

Before centering, 22.7% of the 10 nearest neighbors of an object have the same class label as the object (as indicated by the ratio of the total height of black bars relative to that of all bars in Figure 1(c)). After centering, the percentage increases to 31.6%.

5.2 Real dataset

We did the same analysis as Sections 5.1.2–5.1.3 to a real dataset with multiple-cluster structure: the Reuters Transcribed dataset. This multi-class document classification dataset has ten classes, and each class roughly forms a cluster. We will also use this dataset in an experiment in Section 7.2.

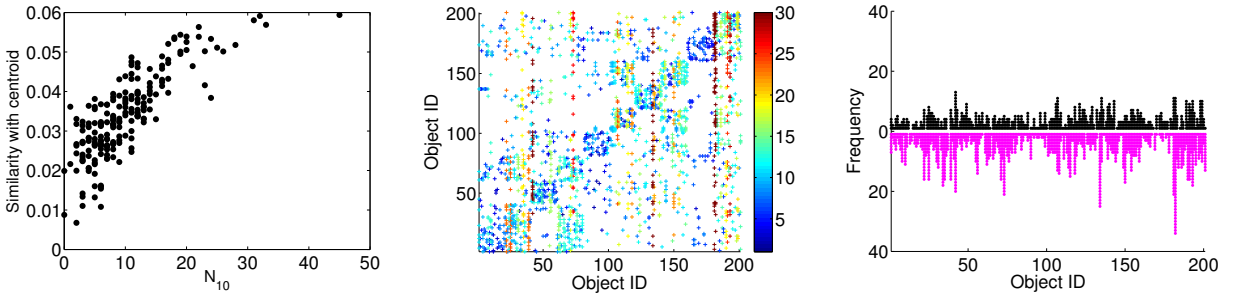
The results are shown in Figure 2. We can observe the same trends as we saw in Figure 1 for the synthetic data: positive correlation between hubness (N_{10}) and inner product with the data centroid before centering; hubs appearing in the nearest neighbors of many objects of different classes; and both are reduced after centering.

The ratio of the height of black bars to that of all bars in Figure 2(c) is 38.4% before centering, whereas it improves to 41.0% after centering (Figure 2(f)).

6 Hubness weighted centering

Centering shifts the origin of the space to the data centroid, and objects similar to the centroid tend to become hubs. Thus in a sense, centering can be interpreted as an operation that shifts the origin towards hubs.

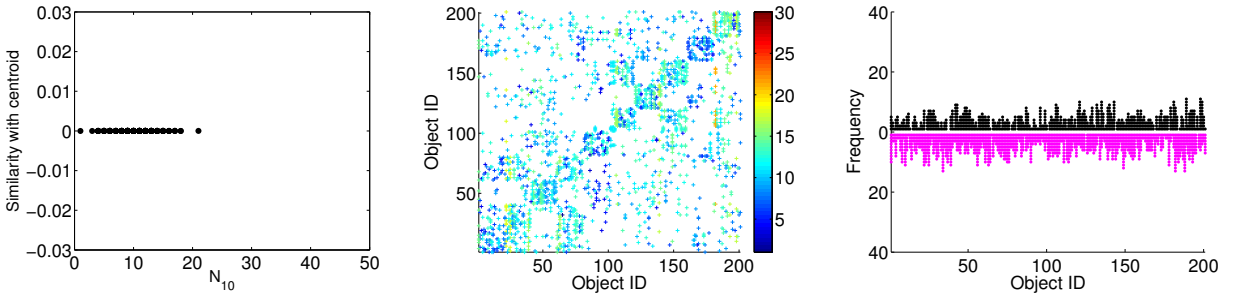
In this section, we extrapolate this interpretation,



(a) Before centering: N_{10} vs. inner product similarity to the data centroid

(b) Before centering: kNN matrix

(c) Before centering: Breakdown of N_{10} by class match/mismatch between objects and neighbors



(d) After centering: N_{10} vs. inner product similarity to the data centroid

(e) After centering: kNN matrix

(f) After centering: Breakdown of N_{10} by class match/mismatch between objects and neighbors

Figure 2: Reuters Transcribed data.

and move the origin more actively towards hub objects in the dataset, rather than towards the data centroid. To this end, we consider *weighted centering*, a variation of centering in which each object is associated with a weight, and the origin is shifted to the weighted mean of the data. Specifically, we define the weight of an object as the sum of the similarities (inner products) between the object and all objects, regarding this sum as the index of how likely the object can be a hub.

6.1 Weighted centering

In weighted centering, we associate weight w_i to each object i in the dataset, and move the origin to the weighted centroid

$$\bar{\boldsymbol{x}}^{\text{weighted}} = \sum_{i=1}^{n} w_i \boldsymbol{x}_i$$

where $\sum_{i=1}^{n} w_i = 1$ and $0 \le w_i \le 1$ for $i = 1, \ldots, n$. Thus, object $\boldsymbol{x}$ is mapped to a new feature vector

$$\boldsymbol{x}^{\text{weighted}} = \boldsymbol{x} - \bar{\boldsymbol{x}}^{\text{weighted}} = \boldsymbol{x} - \sum_{i=1}^{n} w_i \boldsymbol{x}_i.$$

Notice that the original centering formula (2) is recovered by letting $w_i = 1/n$ for all $i = 1, \ldots, n$.

Weighted centering can also be kernelized by using the weighted centering matrix $\mathbf{H}(\boldsymbol{w}) = \mathbf{I} - \mathbb{1}\boldsymbol{w}^{\mathrm{T}}$ in place of $\mathbf{H}$ in Eq. (3). The resulting Gram matrix is

$$\mathbf{K}^{\text{weighted}} = \mathbf{H}(\boldsymbol{w})\mathbf{K}\mathbf{H}(\boldsymbol{w})^{\mathrm{T}}. \tag{10}$$

6.2 Similarity-dependent weighting

To move the origin towards hubs more aggressively, we place more weights on objects that are more likely to become hubs. This likelihood is estimated by the similarity of individual objects to all objects in the data set.

Let d_i be the sum of the similarity between object $\boldsymbol{x}_i$ and all objects in the dataset. So,

$$d_i = \sum_{j=1}^{n} \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = n \, \langle \boldsymbol{x}_i, \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{x}_j \rangle.$$

As seen from the last equation, d_i is proportional to the similarity (inner product) between object $\boldsymbol{x}_i$ and the data centroid.

Now we define $\{w_i\}_{i=1}^n$ from $\{d_i\}_{i=1}^n$ by

$$w_i = \frac{d_i^\gamma}{\sum_{j=1}^n d_j^\gamma},$$

where γ is a parameter controlling how much we emphasize the effect of d_i. Setting $\gamma = 0$ results in $w_i = 1$ for every i, and hence is equivalent to normal centering. When $\gamma > 0$, weighted centering moves the origin closer to the objects with a large d_i than normal centering would.

7 Experiments

We evaluated the effect of centering in two natural language tasks: word sense disambiguation (WSD) and document classification. We are interested in whether hubs are actually reduced after centering, and whether the performance of kNN classification is improved.

Throughout this section, $\mathbf{K}$ denotes cosine similarity matrix; i.e., inner product of feature vectors normalized to unit length; $\mathbf{K}^{\mathsf{cent}}$ denotes the centered similarity matrix computed by Eq. (3) from $\mathbf{K}$; $\mathbf{K}^{\mathsf{weighted}}$ denotes its hubness weighted variant given by Eq. (10). Depending on context, these symbols are also used to denote kNN classifiers using respective similarity measures.

For comparison, we also tested two recently proposed approaches to hub reduction: transformation of the base similarity measure (in our case, $\mathbf{K}$) by Mutual Proximity (Schnitzer et al., 2012)[2], and the one (Suzuki et al., 2012) based on graph Laplacian kernels. Since the Laplacian kernels are defined for graph nodes, we computed them by taking the cosine similarity matrix $\mathbf{K}$ as the weighted adjacency (affinity) matrix of a graph. For Laplacian kernels, we computed both the regularized Laplacian kernel (Chebotarev and Shamis, 1997; Smola and Kondor, 2003) with several parameter values, as well as the commute-time kernel (Saerens et al., 2004), but present only the best results among these kernels.

7.1 Word sense disambiguation

7.1.1 Task and dataset

In the WSD experiment, we used the dataset for the Senseval-3 English Lexical Sample (ELS) task (Mihalcea et al., 2004). It is a collection of sentences containing 57 polysemous words, and each of these sentences is annotated with a gold standard sense of the target word. The goal of the ELS task is to build a classifier for each target word, which, given a context around the word, predicts a sense from the known set of senses.

We used a basic bag-of-words representation for the context surrounding a target word (Mihalcea, 2004; Navigli, 2009). A context is thus represented as a high-dimensional feature vector holding the tf-idf weighted frequency of words[3] in context.

7.1.2 Compared methods

We applied kNN classification using cosine similarity $\mathbf{K}$, and its four transformed similarity measures: centered similarity $\mathbf{K}^{\mathsf{cent}}$, its weighted variant $\mathbf{K}^{\mathsf{weighted}}$, Mutual Proximity and graph Laplacian kernels. The sense of a test object was predicted by voting from the k training objects most similar to the test object, as measured by the respective similarity measures.

We used leave-one-out cross validation within the training data to tune neighborhood size k for the kNN classification and the voting scheme, i.e., either (unweighted) majority vote, or weighted vote in which votes from individual objects are weighted by their similarity score to the test objects. We also selected parameter γ in $\mathbf{K}^{\mathsf{weighted}}$ and the best graph Laplacian kernel among the regularized Laplacian and commute time kernels using the training data.

7.1.3 Evaluation

We computed two indices for each similarity measure: (i) skewness of the N_{10} distribution to evaluate

[2] We used the Matlab script downloaded from http://www.ofai.at/~dominik.schnitzer/mp/.

[3] We removed stop words listed in the on-line appendix of (Lewis et al., 2004).

Method	F1 score	Skewness
$\mathbf{K}$	60.3	4.55
$\mathbf{K}^{\mathsf{cent}}$	64.0	1.19
$\mathbf{K}^{\mathsf{weighted}}$	**64.8**	1.02
Mutual Proximity	63.0	1.00
Graph Laplacian	61.2	4.51
GAMBL (Decadt et al., 2004)	64.5	—

Table 1: WSD results: Macro-averaged F1 score (points) of the compared methods (larger is better) and empirical skewness of the N_{10} distribution for each similarity measure (smaller is better).

the emergence of hubs, and (ii) macro-averaged F1 score to evaluate the classification performance.

Skewness To evaluate the degree of hub emergence for each similarity measure, we followed (Radovanović et al., 2010a) and counted $N_k(\boldsymbol{x})$, the number of times object $\boldsymbol{x}$ occurs in the kNN lists of other objects in the dataset (we fix $k = 10$ below). The emergence of hubs in a dataset can then be quantified with *skewness*, defined as follows:

$$S_{N_k} = \frac{\mathbb{E}\left[(N_k - \mu_{N_k})^3\right]}{\sigma_{N_k}^3}.$$

In this equation, $\mathbb{E}[\cdot]$ denotes expectation, and μ_{N_k} and σ_{N_k} are the mean and the standard deviation of the N_k distribution, respectively.

When hubs exist in a dataset, the distribution of N_k is expected to skew to the right, and yields a large S_{N_k} (Radovanović et al., 2010a). In other words, similarity measures that yield smaller S_{N_k} are more desirable in terms of hub reduction.

Skewness can only be computed for each dataset, and in the WSD task, each target word has its own dataset. Hence we computed the skewness $S_{N_{10}}$ for each word and then took average.

Macro-averaged F1 score Classification performance was measured by the F1 score macro-averaged over all the 57 target words in the Senseval-3 ELS dataset. The standard Senseval-3 ELS scoring method is based on micro average, but we used macro average to make the evaluation consistent with skewness computation, which, as mentioned above, can only be computed for each dataset (i.e., word).

Dataset	#classes	#objects	#features
Reuters Transcribed	10	201	2730
Mini Newsgroups	20	2000	8811

Table 2: Document classification datasets: Number of classes, data size, and number of features.

7.1.4 Result

Table 1 shows the F1 scores and the skewness of the N_{10} distributions, macro averaged over the 57 target words. The table also includes the macro-averaged F1 score[4] of the GAMBL system, the best memory-based system participated in the Senseval-3 ELS task. Note however that GAMBL uses more elaborate features (e.g., part-of-speech of words) than just a plain bag-of-words used by other methods in this comparison. GAMBL also employs complex post-processing of the kNN outputs.

After centering ($\mathbf{K}^{\mathsf{cent}}$ and $\mathbf{K}^{\mathsf{weighted}}$) skewness became markedly smaller than that of the non-centered cosine $\mathbf{K}$. F1 score also improved with the decrease in skewness. In particular, weighted centering ($\mathbf{K}^{\mathsf{weighted}}$) slightly outperformed GAMBL, though the difference was small. Recall however that $\mathbf{K}^{\mathsf{cent}}$ and $\mathbf{K}^{\mathsf{weighted}}$ only use naive bag-of-words features, unlike GAMBL.

7.2 Document classification

7.2.1 Task and dataset

Two multiclass document classification datasets were used: Reuters Transcribed and Mini Newsgroups, distributed at http://archive.ics.uci.edu/ml/. The properties of the datasets are summarized in Table 2.

7.2.2 Evaluation

The performance was evaluated by the F1 score (equivalent to accuracy in this task) of prediction using leave-one-out cross validation, due to the limited number of documents.

7.2.3 Compared methods

We used the cosine similarity as the base similarity matrix ($\mathbf{K}$). The centered similarity matrix ($\mathbf{K}^{\mathsf{cent}}$) and its weighted variant ($\mathbf{K}^{\mathsf{weighted}}$), Mutual

[4]The macro-averaged F1 of GAMBL was calculated from the per-word F1 scores listed in Table 1 of (Decadt et al., 2004).

Method	F1 score	Skewness
$\mathbf{K}$	56.7	1.61
$\mathbf{K}^{\mathsf{cent}}$	**61.2**	0.11
$\mathbf{K}^{\mathsf{weighted}}$	60.2	0.04
Mutual Proximity	60.2	−0.10
Graph Laplacian	57.2	0.37

(a) Reuters Transcribed

Method	F1 score	Skewness
$\mathbf{K}$	76.5	4.37
$\mathbf{K}^{\mathsf{cent}}$	79.0	1.56
$\mathbf{K}^{\mathsf{weighted}}$	**79.4**	1.68
Mutual Proximity	79.0	0.49
Graph Laplacian	77.6	2.13

(b) Mini Newsgroups

Table 3: Document classification results: F1 score (%) (larger is better) and skewness of the N_{10} distribution for each similarity measure (smaller is better).

Proximity, and graph Laplacian based kernels were computed from $\mathbf{K}$.

kNN classification was done in a standard way: The class of object $\boldsymbol{x}$ is predicted by the majority vote from $k = 10$ objects most similar to $\boldsymbol{x}$, measured by a specified similarity measure. The parameter k for the kNN classification, the voting scheme (i.e., either unweighted or weighted majority vote), γ in $\mathbf{K}^{\mathsf{weighted}}$, and the best graph Laplacian kernel were selected by leave-one-out cross validation.

7.2.4 Result

Table 3 shows the F1 score and the skewness of the N_{10} distribution of the respective methods in document classification. Centered cosine ($\mathbf{K}^{\mathsf{cent}}$) outperformed uncentered cosine similarity $\mathbf{K}$, and achieved an F1 score comparable to Mutual Proximity. Weighted centering ($\mathbf{K}^{\mathsf{weighted}}$) further improved F1 on the Mini Newsgroups data.

8 Conclusion

We have shown that centering similarity matrices reduces the emergence of hubs in the data, and consequently improves the accuracy of nearest neighbor classification. We have theoretically analyzed why objects most similar to the mean tend to make hubs, and also proved that centering cancels the bias in the distribution of inner products, and thus is expected to reduce hubs.

In WSD and document classification tasks, kNN classifiers showed much better performance with centered similarity measures than non-centered ones. Weighted centering shifts the origin towards hubs more aggressively, and further improved the classification performance in some cases.

In future work, we plan to exploit the class distribution in the dataset to make more effective similarity measures; notice that the hubness weighted centering of Section 6 is an unsupervised method, in the sense that class information was not used for determining weights. We will investigate if more effective weighting can be done using this information.

Acknowledgments

We thank anonymous reviewers for helpful comments.

References

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. 2005. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382.

P. Yu. Chebotarev and E. V. Shamis. 1997. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9):1505–1514.

Yihua Chen, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. 2009. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10:747–776.

Bart Decadt, Véronique Hoste, Walter Daelemans, and Antal Van den Bosch. 2004. GAMBL, genetic algorithm optimization of memory-based WSD. In Rada Mihalcea and Phil Edmonds, editors, *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 108–112.

L. Eriksson, E. Johansson, N. Kettaneh-Wold, J. Trygg, C. Wikström, and S. Wold. 2006. *Multi- and Megavariate Data Analysis, Part 1, Basic Principles and Applications*. Umetrics, Inc.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 897–906, Honolulu, Hawaii, USA.

Douglas H. Fisher and Hans-Joachim Lenz, editors. 1996. *Learning from Data: Artificial Intelligence and*

Statistics V: Workshop on Artificial Intelligence and Statistics. Lecture Notes in Statistics 112. Springer.

Ruixin Guo and Sounak Chakraborty. 2010. Bayesian adaptive nearest neighbor. *Statistical Analysis and Data Mining*, 3(2):92–105.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing*. Prentice Hall, 2nd edition.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Colin Mallows. 1991. Another comment on O'Cinneide. *The American Statistician*, 45(3):257.

K. V. Mardia and P. Jupp. 2000. *Directional Statistics*. John Wiley and Sons, 2nd edition.

K. V. Mardia, J. T. Kent, and J. M. Bibby. 1979. *Multivariate Analysis*. Academic Press.

Brij M. Masand, Gordon Linoff, and David L. Waltz. 1992. Classifying news stories using memory based reasoning. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92)*, pages 59–65.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English lexical sample task. In Rada Mihalcea and Phil Edmonds, editors, *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 25–28, Barcelona, Spain.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL '04)*, pages 33–40, Boston, Massachusetts, USA.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL '08)*, pages 236–244, Columbus, Ohio, USA.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41:10:1–10:69.

Ali Mustafa Qamar, Éric Gaussier, Jean-Pierre Chevallet, and Joo-Hwee Lim. 2008. Similarity learning for nearest neighbor classification. In *Proceedings of the 8th International Conference on Data Mining (ICDM '08)*, pages 983–988, Pisa, Italy.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010a. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010b. On the existence of obstinate results in vector space models. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pages 186–193, Geneva, Switzerland.

Marco Saerens, François Fouss, Luh Yen, and Pierr Dupont. 2004. The principal components analysis of graph, and its relationships to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning (ECML '04)*, Lecture Notes in Artificial Intelligence 3201, pages 371–383, Pisa, Italy. Springer.

Dominik Schnitzer, Arthur Flexer, Markus Schedl, and Gerhard Widmer. 2012. Local and global scaling reduce hubs in space. *Journal of Machine Learning Research*, 13:2871–2902.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24:97–123.

Alexander J. Smola and Risi Kondor. 2003. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, Proceedings*, Lecture Notes in Artificial Intelligence 2777, pages 144–158. Springer.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11)*, pages 48–52, Portland, Oregon, USA.

Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, Yuji Matsumoto, and Marco Saerens. 2012. Investigating the effectiveness of Laplacian-based kernels in hub reduction. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI-12)*, pages 1112–1118, Toronto, Ontario, Canada.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*, pages 948–957, Uppsala, Sweden.

Jigang Wang, Predrag Neskovic, and Leon N. Cooper. 2006. Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence. *Pattern Recognition*, 39(3):417–423.

Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 42–49, Berkeley, California, USA.

Unsupervised Spectral Learning of WCFG as Low-rank Matrix Completion

Raphaël Bailly[†] **Xavier Carreras**[†] **Franco M. Luque**[‡] **Ariadna Quattoni**[†]

[†] Universitat Politècnica de Catalunya
Barcelona, 08034
rbailly,carreras,aquattoni@lsi.upc.edu

[‡] Universidad Nacional de Córdoba and CONICET
X500HUA Córdoba, Argentina
francolq@famaf.unc.edu.ar

Abstract

We derive a spectral method for unsupervised learning of Weighted Context Free Grammars. We frame WCFG induction as finding a Hankel matrix that has low rank and is linearly constrained to represent a function computed by inside-outside recursions. The proposed algorithm picks the grammar that agrees with a sample and is the simplest with respect to the nuclear norm of the Hankel matrix.

1 Introduction

Weighted Context Free Grammars (WCFG) define an important class of languages. Their expressivity makes them good candidates for modeling a wide range of natural language phenomena. This expressivity comes at a cost: unsupervised learning of WCFG seems to be a particularly hard task. And while it is a well-studied problem, it is still to a great extent unsolved.

Several methods for unsupervised learning of WCFG have been proposed. Some rely on heuristics that are used to build incrementally an approximation of the unknown grammar (Adriaans et al., 2000; Van Zaanen, 2000; Tu and Honavar, 2008). Other methods are based on maximum likelihood estimation, searching for the grammar that has the largest posterior given the training corpus (Baker, 1979; Lari and Young, 1990; Pereira and Schabes, 1992; Klein and Manning, 2002). Several Bayesian inference approaches have also been proposed (Chen, 1995; Kurihara and Sato, 2006; Liang et al., 2007; Cohen et al., 2010). These approaches perform parameter estimation by exploiting Markov sampling techniques.

Recently, for the related problem of unsupervised dependency parsing, Gormley and Eisner (2013) proposed a new way of framing the max-likelihood estimation. In their formulation the problem is expressed as an integer quadratic program subject to non-linear constraints. They exploit techniques from mathematical programming to solve the resulting optimization.

In spirit, the work by Clark (2001; 2007) is probably the most similar to our approach since both approaches share an algebraic view of the problem. In his case the key idea is to work with an algebraic representation of a WCFG. The problem of recovering the constituents of the grammar is reduced to the problem of identifying its syntactic congruence.

In the last years, multiple spectral learning algorithms have been proposed for a wide range of models (Hsu et al., 2009; Bailly et al., 2009; Bailly et al., 2010; Balle et al., 2011; Luque et al., 2012; Cohen et al., 2012). Since the spectral approach provides a good thinking tool to reason about distributions over Σ^*, the question of whether they can be used for unsupervised learning of WCFG seems natural. Still, while spectral algorithms for unsupervised learning of languages can learn regular languages, tree languages and simple dependency grammars, the frontier to WCFG seems hard to reach.

In fact, the most recent theoretical results on spectral learning of WCFG do not seem to be very encouraging. Recently, Hsu et al. (2012) showed that the problem of recovering the joint distribution over PCFG derivations and their yields is not identifiable.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 624–635,
Seattle, Washington, USA, 18-21 October 2013.

Although, for some simple grammar subclasses (e.g. independent left and right children), identification in the weaker sense (over the yields of the grammar) implies strong identification (e.g. over joint distribution of yields and derivations). In their paper, they propose a spectral algorithm based on a generalization of the method of moments for these restricted subclasses.

Thus one open direction for spectral research consists on defining subclasses of context free languages that can be learned (in the strong sense) from observations of yields. Yet, an alternative research direction is to consider learnability in the weaker sense. In this paper we take the second road, and focus on the problem of approximating the distribution over yields generated by a WCFG.

Our main contribution is to present a spectral algorithm for unsupervised learning of WCFG. Following ideas from Balle et al. (2012), the algorithm is framed as a convex optimization where we search for a low-rank matrix satisfying two types of constraints: (1) Constraints derived from observable statistics over yields; and (2) Constraints derived from certain recurrence relations satisfied by a WCFG. Our derivations of the learning algorithm illustrate the main ingredients behind the spectral approach to learning functions over Σ^* which are: (1) to exploit the recurrence relations satisfied by the target family of functions and (2) provide algebraic formulations of these relations.

We alert the reader that although we are able to frame the problem as a convex optimization, the number of variables involved is quite large and prohibits a practical implementation of the method on a realistic scenario. The experiments we present should be regarded as examples designed to illustrate the behavior of the method. More research is needed to make the optimization more efficient, and we are optimistic that such improvements can be achieved by exploiting problem-specific properties of the optimization. Regardless of this, ours is a novel way of framing the grammatical inference problem.

The rest of the paper is organized as follows. Section 2 gives preliminaries on WCFG and the type of functions we will learn. Section 3 establishes that spectral methods can learn a WCFG from a Hankel matrix containing statistics about context-free cuts. Section 4 presents the unsupervised algorithm, where we formulate grammar induction as a low-rank optimization. Section 5 presents experiments, and finally we conclude the paper.

Notation Let Σ be an alphabet. We use σ to denote an arbitrary symbol in Σ. The set of all finite strings over Σ is denoted by $\Sigma^\star$, where we write λ for the empty string. We also use the set $\Sigma^+ = \Sigma^\star \setminus \{\lambda\}$.

We use bold letters to represent column vectors $\boldsymbol{v}$ and matrices $\boldsymbol{M}$. We use $\boldsymbol{I}_n$ to denote the n-dimensional identity matrix. We use $\boldsymbol{M}^+$ to denote the *Moore-Penrose pseudoinverse* of some matrix $\boldsymbol{M}$. $\boldsymbol{M} \otimes \boldsymbol{M}'$ is the Kronecker product between matrices $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{M}' \in \mathbb{R}^{p \times q}$ resulting in a matrix in $\mathbb{R}^{mp \times nq}$. The rest of notation will be given as needed.

2 Weighted Context Free Grammars

In this section we define Weighted Context Free Grammars (WCFG). We start with a classic definition and then describe an algebraic form of WCFG that will be used throughout the paper. We also describe the fundamental recursions in WCFG.

2.1 WCFG in Classic Form

A WCFG over Σ is a tuple $\bar{G} = \langle V, R, T, w_\star, w_T, w_R \rangle$ where

- V is the set of non-terminal symbols. We assume that $V = \{1, \ldots, n\}$ for some natural number n, and that $V \cap \Sigma = \emptyset$.
- R is a set of binary rules of the form $i \to j\ k$ where $i, j, k \in V$.
- T is a set of unary rules of the form $i \to \sigma$ where $i \in V$ and $\sigma \in \Sigma$.
- $w_\star : V \to \mathbb{R}$, with $w_\star(i)$ being the weight of starting a derivation with non-terminal i.
- $w_T : V \times \Sigma \to \mathbb{R}$, with $w_T(i \to \sigma)$ being the weight of rule rewriting i into σ.
- $w_R : V \times V \times V \to \mathbb{R}$, with $w_R(i \to j\ k)$ being the weight of rewriting i into $j\ k$.

A WCFG $\bar{G}$ computes a function $g_{\bar{G}} : \Sigma^+ \to \mathbb{R}$ defined as

$$g_{\bar{G}}(x) = \sum_{i \in V} w_\star(i) \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x) \quad , \qquad (1)$$

where we define the *inside* function $\bar{\beta}_{\bar{G}} : V \times \Sigma^+ \rightarrow \mathbb{R}$ recursively:

$$\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} \sigma) = w_T(i \rightarrow \sigma) \tag{2}$$

$$\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x) = \sum_{\substack{j,k \in V \\ x_1,x_2 \in \Sigma^+ \\ \text{s.t. } x = x_1 x_2}} w_R(i \rightarrow j\ k)\ \bar{\beta}_{\bar{G}}(j \overset{\star}{\Rightarrow} x_1) \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2)\ , \tag{3}$$

where in the second case we assume $|x| > 1$. The inside function $\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x)$ exploits the fundamental inside recursion in WCFG (Baker, 1979; Lari and Young, 1990). We will find useful to define the *outside* function $\bar{\alpha}_{\bar{G}} : \Sigma^\star \times V \times \Sigma^\star \rightarrow \mathbb{R}$ defined recursively as:

$$\bar{\alpha}_{\bar{G}}(\lambda; i; \lambda) = w_\star(i) \tag{4}$$

$$\begin{aligned} \bar{\alpha}_{\bar{G}}(x; i; y) = &\sum_{\substack{j,k \in V \\ x_1 \in \Sigma^\star, x_2 \in \Sigma^+ \\ \text{s.t. } x = x_1 x_2}} w_R(j \rightarrow k\ i) \cdot \\ &\bar{\alpha}_{\bar{G}}(x_1; j; y) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2) \\ &+ \sum_{\substack{j,k \in V \\ y_1 \in \Sigma^+, y_2 \in \Sigma^\star \\ \text{s.t. } y = y_1 y_2}} w_R(j \rightarrow i\ k) \cdot \\ &\bar{\alpha}_{\bar{G}}(x; j; y_2) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} y_1)\ , \end{aligned} \tag{5}$$

where in the second case we assume that either $x \neq \lambda$ or $y \neq \lambda$.

For $x, z \in \Sigma^\star$ and $y \in \Sigma^+$ we have that

$$\sum_{i \in V} \bar{\alpha}_{\bar{G}}(x; i; z) \cdot \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} y) \tag{6}$$

is the weight that the grammar $\bar{G}$ assigns to a string xyz that has a *cut* or bracketing around y. Technically, it corresponds to the sum of the weights of all derivations that have a constituent spanning y. In particular we have that

$$g_{\bar{G}}(x) = \sum_i \bar{\alpha}_{\bar{G}}(\lambda; i; \lambda) \cdot \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x)\ .$$

If x is a string of length m, and $x_{[t:t']}$ is the substring of x from positions t to t', it also happens that

$$g_{\bar{G}}(x) = \sum_i \bar{\alpha}_{\bar{G}}(x_{[1:t-1]}; i; x_{[t+1:m]}) \cdot \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x_{[t]}))$$

for any t between 1 and m.

In this paper we will make frequent use of inside and outside quantities. Notationally, for outsides the semi-colon between two strings, i.e. $x; z$, will simbolize a cut where we can insert an inside string y.

Finally, we note that Probabilistic Context Free Grammars (PCFG) are a special case of WCFG where: $w_\star(i)$ is the probability to start a derivation with non-terminal i; $w_R(i \rightarrow j\ k)$ is the conditional probability of rewriting nonterminal i into j and k; $w_T(i \rightarrow \sigma)$ is the probability of rewriting i into symbol σ; $\sum_i w_\star(i) = 1$; and for each $i \in V$, $\sum_{j,k} w_R(i \rightarrow j\ k) + \sum_\sigma w_T(i \rightarrow \sigma) = 1$. Under these conditions the function $g_{\bar{G}}$ is a probability distibution over Σ^+.

2.2 WCFG in Algebraic Form

We now define a WCFG in algebraic form. A Weighted Context Free Grammar (WCFG) over Σ with n states is a tuple $G = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \boldsymbol{A} \rangle$ with:

- An initial vector $\boldsymbol{\alpha}_\star \in \mathbb{R}^n$.
- Terminal vectors $\boldsymbol{\beta}_\sigma \in \mathbb{R}^n$ for $\sigma \in \Sigma$.
- A bilinear operator $\boldsymbol{A} \in \mathbb{R}^{n \times n^2}$.

A WCFG G computes a function $g_G : \Sigma^\star \rightarrow \mathbb{R}$ defined as

$$g_G(x) = \boldsymbol{\alpha}_\star^\top \beta_G(x) \tag{7}$$

where the *inside* function $\beta_G : \Sigma^+ \rightarrow \mathbb{R}^n$ is

$$\beta_G(\sigma) = \boldsymbol{\beta}_\sigma \tag{8}$$

$$\beta_G(x) = \sum_{\substack{x_1,x_2 \in \Sigma^+ \\ x = x_1 x_2}} \boldsymbol{A}(\beta_G(x_1) \otimes \beta_G(x_2)) \tag{9}$$

We will define the *outside* function $\alpha_G : \Sigma^\star \times \Sigma^\star \rightarrow \mathbb{R}^n$ as:

$$\alpha_G(\lambda; \lambda) = \boldsymbol{\alpha}_\star \tag{10}$$

$$\begin{aligned} \alpha_G(x; z)^\top = &\sum_{\substack{x_1 \in \Sigma^\star, x_2 \in \Sigma^+ \\ x = x_1 x_2}} \alpha_G(x_1; z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{I}_n) \\ &+ \sum_{\substack{z_1 \in \Sigma^+, z_2 \in \Sigma^\star \\ z = z_1 z_2}} \alpha_G(x; z_2)^\top \boldsymbol{A}(\boldsymbol{I}_n \otimes \beta_G(z_1)) \end{aligned} \tag{11}$$

For $x, z \in \Sigma^\star$ and $y \in \Sigma^+$ we have that

$$\alpha_G(x; z)^\top \beta_G(y) \tag{12}$$

is the weight that the grammar assigns to the string xyz with a cut around y. In particular, $g_G(x) = \alpha_G(\lambda; \lambda)^\top \beta_G(x)$.

Let us make clear that a WCFG is the same device in classic or algebraic forms. If $\bar{G} = \langle V, R, T, w_\star, w_T, w_R \rangle$ and $G = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \boldsymbol{A} \rangle$, the mapping is:

$$w_\star(i) = \boldsymbol{\alpha}_\star(i) \tag{13}$$
$$w_T(i \to \sigma) = \boldsymbol{\beta}_\sigma[i] \tag{14}$$
$$w_R(i \to j\ k) = \boldsymbol{A}[i, j, k] \tag{15}$$
$$\bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x) = \beta_G(x)[i] \tag{16}$$
$$\bar{\alpha}_{\bar{G}}(x; i; z) = \alpha_G(x; z)[i] \tag{17}$$

See Section A.1 for a proof of Eq. 16 and 17.

3 WCFG and Hankel Matrices

In this section we describe Hankel matrices for WCFG. These matrices explicitly capture inside-outside recursions employed by WCFG functions, and are key to a derivation of a spectral learning algorithm that learns a grammar G using statistics of a training sample.

Let us define some sets. We say that $\mathcal{I}^1 = \Sigma^+$ is the set of *inside strings*. The set of *composed inside strings* $\mathcal{I}^2$ is the set of elements (x, x'), where $x, x' \in \Sigma^+$. Intuitively (x, x') represents two adjacent spans with an operation, i.e., it keeps the trace of the operation that composes x with x' and yields xx'. We will use the set $\mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$.

The set of *outside contexts* $\mathcal{O}$ is the set containing elements $\langle x; z \rangle$, where $x, z \in \Sigma^\star$. Intuitively, $\langle x; z \rangle$ represents a context where we can insert an inside element y in between x and z, yielding xyz.

Consider a function $f : \mathcal{O} \times \mathcal{I} \to \mathbb{R}$. The Hankel matrix of f is the bi-infinite matrix $\boldsymbol{H}_f \in \mathbb{R}^{\mathcal{O} \times \mathcal{I}}$ such that $\boldsymbol{H}_f(o, i) = f(o, i)$.

In practice we will work with finite sub-blocks of $\boldsymbol{H}_f$. To this end we will employ the notion of basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, where $\{\langle \lambda, \lambda \rangle\} \subseteq \mathcal{P} \subseteq \mathcal{O}$ is a set of outside contexts and $\Sigma \subseteq \mathcal{S} \subseteq \mathcal{I}^1$ is a set of inside strings. We will use $p = |\mathcal{P}|$ and $s = |\mathcal{S}|$. Furthermore, we define the *inside completion* of $\mathcal{S}$ as the set $\mathcal{S}^\dagger = \{(x, x') \mid x, x' \in \mathcal{S}\}$. Note that $\mathcal{S}^\dagger \subseteq \mathcal{I}^2$. We say that $\mathcal{B}^\dagger = (\mathcal{P}, \mathcal{S}^\dagger)$ is the inside completion of $\mathcal{B}$.

The sub-block of $\boldsymbol{H}_f$ defined by $\mathcal{B}$ is the $p \times s$ matrix $\boldsymbol{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ with $\boldsymbol{H}_\mathcal{B}(o, i) = \boldsymbol{H}_f(o, i) = f(o, i)$. In addition to $\boldsymbol{H}_\mathcal{B}$, we are interested in these additional finite vectors and matrices:

- $\boldsymbol{h}_\star \in \mathbb{R}^\mathcal{S}$ is the s-dimensional vector with coordinates $\boldsymbol{h}_\star(x) = f(\langle \lambda, \lambda \rangle, x)$.
- $\boldsymbol{h}_\sigma \in \mathbb{R}^\mathcal{P}$ is the p-dimensional vector with coordinates $\boldsymbol{h}_\sigma(o) = f(o, \sigma)$.
- $\boldsymbol{H}_A \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}^\dagger}$ with $\boldsymbol{H}_A(o, (x_1, x_2)) = f(o, (x_1, x_2))$.

3.1 Hankel Factorizations

If f is computed by a WCFG G, then $\boldsymbol{H}_f$ has rank n factorization. To see this, consider the following matrices. First a matrix $\boldsymbol{S} \in \mathbb{R}^{n \times \mathcal{I}^1}$ of *inside* vectors for all strings, with column x taking value $\boldsymbol{S}_x = \beta_G(x)$. Then a matrix $\boldsymbol{P} \in \mathbb{R}^{\mathcal{O} \times n}$ of outside vectors for all contexts, with row $\langle x; z \rangle$ taking value $\boldsymbol{P}_{\langle x; z \rangle} = \alpha_G(x; z)$. It is easy to see that $\boldsymbol{H}_f = \boldsymbol{P}\boldsymbol{S}$, since $\boldsymbol{H}_f(\langle x; z \rangle, y) = \boldsymbol{P}_{\langle x; z \rangle}\boldsymbol{S}_y = \alpha_G(x; z)^\top \beta_G(y)$. Therefore $\boldsymbol{H}_f$ has rank n.

The same happens for sub-blocks. If $\boldsymbol{H}_\mathcal{B}$ is the sub-block associated with basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, then the sub-blocks $\boldsymbol{P}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times n}$ and $\boldsymbol{S}_\mathcal{B} \in \mathbb{R}^{n \times \mathcal{S}}$ of $\boldsymbol{P}$ and $\boldsymbol{S}$ also accomplish that $\boldsymbol{H}_\mathcal{B} = \boldsymbol{P}_\mathcal{B}\boldsymbol{S}_B$. It also happens that

$$\boldsymbol{h}_\star^\top = \boldsymbol{\alpha}_\star^\top \boldsymbol{S}_\mathcal{B} \tag{18}$$
$$\boldsymbol{h}_\sigma = \boldsymbol{P}_\mathcal{B}\boldsymbol{\beta}_\sigma \tag{19}$$
$$\boldsymbol{H}_A = \boldsymbol{P}_\mathcal{B}\boldsymbol{A}(\boldsymbol{S}_\mathcal{B} \otimes \boldsymbol{S}_\mathcal{B}) \quad . \tag{20}$$

We say that a basis $\mathcal{B}$ is complete for f if $\text{rank}(\boldsymbol{H}_\mathcal{B}) = \text{rank}(\boldsymbol{H}_f)$. The following is a key result for spectral methods.

Lemma 1. *Let $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ be a complete basis of dimension n for a function f and let $\boldsymbol{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ be the Hankel sub-block of f for $\mathcal{B}$. Let $\boldsymbol{h}_\star$, $\boldsymbol{h}_\sigma$ and $\boldsymbol{H}_A$ be the additional matrices for $\mathcal{B}$. If $\boldsymbol{H}_\mathcal{B} = \boldsymbol{P}\boldsymbol{S}$ is a rank n factorization, then the WCFG $G = \langle \boldsymbol{\alpha}_\star, \{\boldsymbol{\beta}_\sigma\}, \boldsymbol{A} \rangle$ with*

$$\boldsymbol{\alpha}_\star^\top = \boldsymbol{h}_\star^\top \boldsymbol{S}^+ \tag{21}$$
$$\boldsymbol{\beta}_\sigma = \boldsymbol{P}^+ \boldsymbol{h}_\sigma \tag{22}$$
$$\boldsymbol{A} = \boldsymbol{P}^+ \boldsymbol{H}_A(\boldsymbol{S} \otimes \boldsymbol{S})^+ \tag{23}$$

computes f.

See proof in Section A.2.

3.2 Supervised Spectral Learning of WCFG

The spectral learning method directly exploits Lemma 1. In a nutshell, the spectral method is:

1. Choose a complete basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ and a dimension n.
2. Use training data to compute estimates of the necessary Hankel matrices: $\boldsymbol{H}_{\mathcal{B}}$, $\boldsymbol{h}_{\star}$, $\boldsymbol{h}_{\sigma}$, $\boldsymbol{H}_{A}$.
3. Compute the SVD of $\boldsymbol{H}_{\mathcal{B}}$, $\boldsymbol{H}_{\mathcal{B}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^{\top}$.
4. Create a truncated rank n factorization of $\boldsymbol{H}_{\mathcal{B}}$ as $\boldsymbol{P}_n\boldsymbol{S}_n$, having $\boldsymbol{P}_n = \boldsymbol{U}_n\boldsymbol{\Lambda}_n$ and $\boldsymbol{S}_n = \boldsymbol{V}_n^{\top}$, where we only consider the top n singular values/vectors of $\boldsymbol{\Lambda}, \boldsymbol{U}, \boldsymbol{V}$.
5. Use Lemma 1 to compute G, using $\boldsymbol{P}_n$ and $\boldsymbol{S}_n$.

Because of Lemma 1, if $\mathcal{B}$ is complete and we have access to the true $\boldsymbol{H}_{\mathcal{B}}$, $\boldsymbol{h}_{\star}$, $\boldsymbol{h}_{\sigma}$, $\boldsymbol{H}_{A}$ of a WCFG target function g^*, then the algorithm will compute a G that exactly computes g^*. In practice, we only have access to empirical estimates of the Hankel matrices. In this case, there exist PAC-style sample complexity bounds that state that g_G will be a close approximation to g^* (Hsu et al., 2009; Bailly et al., 2009; Bailly et al., 2010).

The parameters of the algorithm are the basis and the dimension of the grammar n. One typically employs some validation strategy using held-out data. Empirically, the performance of these methods has been shown to be good, and similar to that of EM (Luque et al., 2012; Cohen et al., 2013). It is also important to mention that in the case that the target g^* is a probability distribution, the function g_G will be close to g^*, but it will only define a distribution *in the limit*: in practice it will not sum to one, and for some inputs it might return negative values. This is a practical difficulty of spectral methods, for example to apply evaluation metrics like perplexity which are only defined for distributions.

4 Unsupervised Learning of WCFG

In the previous section we have exposed that if we have access to estimates of a Hankel matrix of a WCFG G, we can recover G. However, the statistics in the Hankel require access to strings that have information about context-free cuts. We will assume that we only have access to statistics about plain strings of a distribution, i.e. $p(x)$, which we call *observations*. In this scenario, one natural idea is to search for a Hankel matrix that agrees with the observations. The method we present in this section frames this problem as a low-rank matrix optimization problem. We first characterize the space of solutions to our problem, i.e. Hankel matrices associated with WCFG that agree with observable statistics. Then we present the method.

4.1 Characterization of a WCFG Hankel

In this section we describe valid WCFG Hankel matrices using linear constraints.

We first describe an inside-outside basis that is an extension of the one in the previous section. Inside elements are the same, namely $\mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$, where $\mathcal{I}^1$ are strings (x) and $\mathcal{I}^2$ are composed strings (x, x'). The set of *outside contexts* $\mathcal{O}^1$ is the set containing elements $\langle x; z\rangle$, defined as before. The set of *composed outside contexts* has elements $\langle x, x'; z\rangle$, and $\langle x; z', z\rangle$, where $x, z \in \Sigma^{\star}$ and $x', z' \in \Sigma^{+}$. These outside contexts keep an operation open in one of the sides. For example, if we consider $\langle x; z', z\rangle$ and insert a string y, we obtain $x(y, z')z$, where we use (y, z') to explicitly denote a composed inside string. We will use $\mathcal{O} = \mathcal{O}^1 \cup \mathcal{O}^2$.

In this section, we will assume that $\mathcal{I}$ and $\mathcal{O}$ are finite and closed. By closed, we mean that:

- $(x) \in \mathcal{I} \Rightarrow (x_1, x_2) \in \mathcal{I}$ for $x = x_1x_2$
- $(x_1, x_2) \in \mathcal{I} \Rightarrow x_1 \in \mathcal{I}, x_2 \in \mathcal{I}$
- $\langle x; z\rangle \in \mathcal{O} \Rightarrow \langle x_1, x_2; z\rangle \in \mathcal{O}$ for $x = x_1x_2$
- $\langle x; z\rangle \in \mathcal{O} \Rightarrow \langle x; z_1, z_2\rangle \in \mathcal{O}$ for $z = z_1z_2$
- $\langle x_1, x_2; z\rangle \in \mathcal{O} \Rightarrow (x_2) \in \mathcal{I}$
- $\langle x; z_1, z_2\rangle \in \mathcal{O} \Rightarrow (z_1) \in \mathcal{I}$

We will consider a Hankel matrix $\boldsymbol{H} \in \mathbb{R}^{\mathcal{O}\times\mathcal{I}}$. Some entries of this matrix will correspond to *observable* quantities. Specifically, for any string $x \in \mathcal{I}^1$ for which we know $p(x)$ we can define the following *observable constraint*:

$$p(x) = \boldsymbol{H}(\langle\lambda;\lambda\rangle, (x)) \tag{24}$$

The rest of entries of $\boldsymbol{H}$ correspond to a string with an inside-outside cut, and these are not observable. Our method will infer the values of these entries. The following constraints will ensure that the matrix $\boldsymbol{H}$ is a well defined Hankel matrix for WCFG:

- Hankel constraints: $\forall \langle x; z\rangle \in \mathcal{O}, (y_1, y_2) \in \mathcal{I}$

$$\begin{aligned}\boldsymbol{H}(\langle x; z\rangle, (y_1, y_2)) &= \boldsymbol{H}(\langle x, y_1; z\rangle, (y_2)) \\ &= \boldsymbol{H}(\langle x; y_2, z\rangle, (y_1)) \quad (25)\end{aligned}$$

- Inside constraints: $\forall\, o \in \mathcal{O}, (x) \in \mathcal{I}$

$$\boldsymbol{H}(o, (x)) = \sum_{x=x_1x_2} \boldsymbol{H}(o, (x_1, x_2)) \quad (26)$$

- Outside constraints: $\forall \langle x; z\rangle \in \mathcal{O}, i \in \mathcal{I}$

$$\begin{aligned}\boldsymbol{H}(\langle x; z\rangle, i) &= \sum_{x=x_1x_2} \boldsymbol{H}(\langle x_1, x_2; z\rangle, i) \\ &+ \sum_{z=z_1z_2} \boldsymbol{H}(\langle x; z_1, z_2\rangle, i) \quad (27)\end{aligned}$$

Constraint (25) states that composition operations that result in the same structure should have the same value. Constraints (26) and (27) ensure that the values in the Hankel follow the inside-outside recursions that define the computations of a WCFG function. The following lemma formalizes this concept. Let $\boldsymbol{H}_\varepsilon$ be the sub-block of $\boldsymbol{H}$ restricted to $\mathcal{O}^1 \times \mathcal{I}^1$, i.e. without compositions.

Lemma 2. *If $\boldsymbol{H}$ satisfies constraints (25),(26) and (27), and if $\mathrm{rank}(\boldsymbol{H}) = \mathrm{rank}(\boldsymbol{H}_\varepsilon)$ then there exists a WCFG that generates $\boldsymbol{H}_\varepsilon$.*

See proof in Section A.3.

4.2 Convex Optimization

We now present the core optimization program behind our method. Let $\mathbf{vec}(\boldsymbol{H})$ be a vector in $\mathbb{R}^{|\mathcal{O}|\cdot|\mathcal{I}|}$ corresponding to all coefficients of $\boldsymbol{H}$ in column vector form. Let $\boldsymbol{O}$ be a matrix such that $\boldsymbol{O} \cdot \mathbf{vec}(\boldsymbol{H}) = \boldsymbol{z}$ represents the observation constraints. For example, if i-th row of $\boldsymbol{O}$ corresponds to the Hankel coefficient $\boldsymbol{H}(\langle\lambda;\lambda\rangle, (x))$ then $\boldsymbol{z}(i) = p(x)$. Let $\boldsymbol{K}$ be a matrix such that $\boldsymbol{K} \cdot \mathbf{vec}(\boldsymbol{H}) = \mathbf{0}$ represents the constraints (25), (26) and (27).

The optimization problem is:

$$\begin{aligned}\underset{\boldsymbol{H}}{\text{minimize}} \quad & \mathrm{rank}(\boldsymbol{H}) \\ \text{subject to} \quad & \|\boldsymbol{O} \cdot \mathbf{vec}(\boldsymbol{H}) - \boldsymbol{z}\|_2 \leq \mu \\ & \boldsymbol{K} \cdot \mathbf{vec}(\boldsymbol{H}) = \mathbf{0} \\ & \|\boldsymbol{H}\|_2 \leq 1.\end{aligned} \quad (28)$$

Intuitively, we look for $\boldsymbol{H}$ that agrees with the observable statistics and satisfies the inside-outside constraints. μ is a parameter of the method that controls the degree of error in fitting the observables $\boldsymbol{z}$. The $\|\boldsymbol{H}\|_2 \leq 1$ is satisfied by any Hankel matrix derived from a true distribution, and is used to avoid incoherent solutions.

The above optimization problem, however, is computationally hard because of the rank objective. We employ a common relaxation of the rank objective, based on the nuclear norm as in (Balle et al., 2012). The optimization is:

$$\begin{aligned}\underset{\boldsymbol{H}}{\text{minimize}} \quad & \|\boldsymbol{H}\|_* \\ \text{subject to} \quad & \|\boldsymbol{O} \cdot \mathbf{vec}(\boldsymbol{H}) - \boldsymbol{z}\|_2 \leq \mu \\ & \boldsymbol{K} \cdot \mathbf{vec}(\boldsymbol{H}) = \mathbf{0} \\ & \|\boldsymbol{H}\|_2 \leq 1.\end{aligned} \quad (29)$$

To optimize (29) we employ a projected gradient strategy, similar to the FISTA scheme proposed by Beck and Teboulle (2009). The method alternates between separate projections for the observable constraints, the ℓ_2 norm, the inside-outside constraints, and the nuclear norm. Of these, the latter two are the most expensive.

Elsewhere, we develop theoretical properties of the optimization (28) applied to finite-state transductions (Bailly et al., 2013). One can prove that there is theoretical identifiability of the rank and the parameters of an FST distribution, using a rank minimization formulation. However, this problem is NP-hard, and it remains open whether there exists a polynomial method with identifiability results. These results should generalize to WCFG.

5 Experiments

In this section we describe some experiments with the learning algorithms for WCFG. Our goal is to verify that the algorithms can learn some basic context-free languages, and to study the possibility of using them on real data.

5.1 Synthetic Experiments

We performed experiments on synthetic data, obtained by choosing a PCFG with random parameters ($\in [0, 1]$), with a normalization step in order to get a probability distribution. We built the Hankel matrix from the inside basis $\{(x)\}_{x\in\Sigma}$ and outside basis

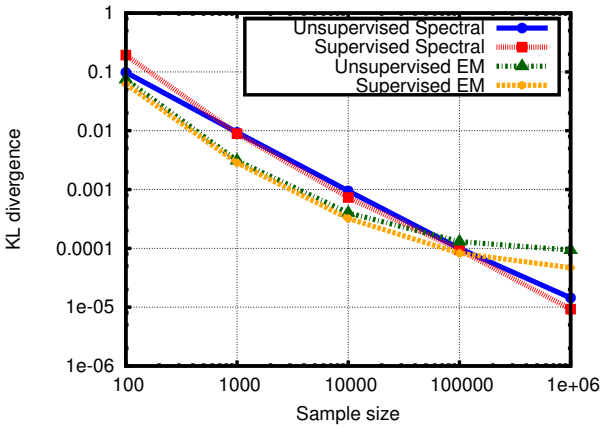


Figure 1: KL divergence for spectral and EM methods, unsupervised and supervised, for different sizes of learning sample, on log-log scales. Results are averages over 50 random target PCFG with 2 states and 2 symbols.

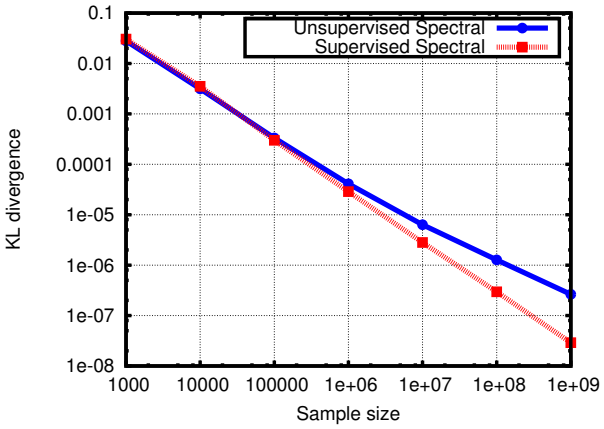


Figure 2: KL divergence for unsupervised and supervised spectral methods, for different sizes of learning sample, on log-log scales. Results are averages over 50 random target PCFG with 3 states and 6 symbols.

$\{\langle\lambda;\lambda\rangle\} \cup \{\langle x;\lambda\rangle, \langle\lambda;x\rangle\}_{x\in\Sigma}$. The composed insides for the operator matrix are thus $\{(x,y)\}_{x,y\in\Sigma}$. The matrix in the optimizer has the following structure

$$H = \begin{array}{c} \\ \langle\lambda;\lambda\rangle \\ \langle x;\lambda\rangle \\ \langle\lambda;x\rangle \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} (y) & \cdots & (y,z) \end{array} \\ \begin{pmatrix} (\lambda;y;\lambda) & \cdots & (\lambda;y,z;\lambda) \\ (x;y;\lambda) & \cdots & (x;y,z;\lambda) \\ (\lambda;y;x) & \cdots & (\lambda;y,z;x) \\ \cdots & \cdots & \cdots \end{pmatrix} \end{array}$$

The constraints we use are:

$$\begin{aligned} K = &\{\boldsymbol{H}((x;y;\lambda)) = \boldsymbol{H}((\lambda;x;y))\}_{x,y\in\Sigma} \cup \\ &\{\boldsymbol{H}((\lambda;x;y)) = \boldsymbol{H}((\lambda;x,y;\lambda))\}_{x,y\in\Sigma} \cup \\ &\{\boldsymbol{H}((x;y;\lambda)) = \boldsymbol{H}((\lambda;x,y;\lambda))\}_{x,y\in\Sigma} \end{aligned}$$

and

$$\begin{aligned} O = &\{\boldsymbol{H}((\lambda;x;\lambda)) = p_S(x)\}_{x\in\Sigma} \cup \\ &\{\boldsymbol{H}((\lambda;x;y)) = p_S(xy)\}_{x,y\in\Sigma} \cup \\ &\{\boldsymbol{H}((x;y,z;\lambda)) + \boldsymbol{H}((\lambda;x,y;z)) = p_S(xyz)\}_{x,y,z\in\Sigma} \end{aligned}$$

We use p_S to denote the empirical distribution. Those are simplified versions of the Hankel, inside, outside and observation constraints. The set O is built from the following remarks: (1) $(xy) = (x,y)$ and (2) $(xyz) = (xy,z)+(x,yz)$. The method uses statistics for sequences up to length 3.

The algorithm we use for the unsupervised spectral method is a simplified version: we use alternatively a hard projection on the constraints (by projecting iteratively on each constraint), and a thresholding-shrinkage operation for the target dimension. We use the same trick as FISTA for the update. We finally use the regular spectral method on this matrix to get our model.

We compare this method with an unsupervised EM, and also with supervised versions of spectral method and EM. We compare the accuracy of the different models in terms of KL-divergence for sequences up to length 10. We run 50 optimization steps for the unsupervised spectral method, and 200 iterations for the EM methods. Figure 1 shows the results, corresponding the the geometric mean over 50 experiments on random targets of 2 symbols and 2 states.

For sample size greater than 10^5, the unsupervised spectral method seems to provide better solutions than both EM and supervised EM. The solution, in terms of KL-divergence, is comparable to the one obtained with the supervised spectral method. The computation time of unsupervised spectral method is almost constant w.r.t. the sample size, around $1.67s$, while computation time of unsupervised EM (resp. supervised EM) is $6.10^3 s$ (resp. $2.10^4 s$) for sample size 10^6.

Figure 2 presents learnings curve for random targets with 3 states and 6 symbols. One can see that, for big sample sizes (10^9), the unsupervised spectral method is losing accuracy compared to the supervised method. This is due to a lack of information,

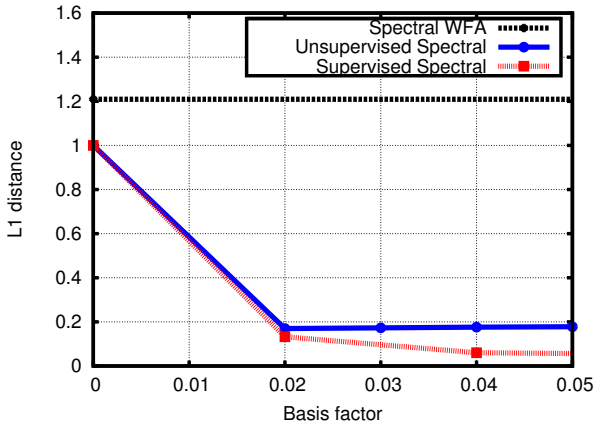


Figure 3: Learning errors for different models in terms of the size of the basis.

and could be overcome by considering a greater basis (e.g. inside sequences up to length 2 or 3).

5.2 Dyck Languages

We now present experiments using the following PCFG:

$$S \rightarrow \; S\,S\;(0.2) \mid a\,S\,b\;(0.4) \mid a\,b\;(0.4)$$

This PCFG generates a probabilistic version of the well-known Dyck language or balanced parenthesis language, an archetypical context-free language.

We do experiments with the following models and algorithms:

- WFA: a Weighted Finite Automata learned using spectral methods as described in (Luque et al., 2012). Parameters: number of states and size of basis.

- Supervised Spectral: a WCFG learned from structured strings using the algorithm of section 3.2. We choose as basis the most frequent insides and outsides observed in the training data. The size of the basis is determined by a parameter f called the basis factor, that determines the proportion of total insides and outsides that will be in the basis.

- Unsupervised Spectral: a WCFG learned from strings using the algorithm of Section 4. The basis is like in the supervised case, but since context-free cuts in the strings are not observed, all possible inside and outsides of the sample (i.e. all possible substrings and contexts) are considered.

basis	size of $\boldsymbol{H}$	obs.	i/o ctr.
1×11	39×159	34	162
6×14	$1{,}163 \times 764$	146	6,360
12×18	$4{,}462 \times 2{,}239$	322	25,374
18×22	$9{,}124 \times 4{,}149$	479	52,524
24×26	$15{,}755 \times 6{,}858$	657	89,718
30×29	$19{,}801 \times 8{,}545$	769	112,374
36×34	$27{,}989 \times 11{,}682$	916	156,690
42×37	$3{,}638 \times 15{,}026$	1,035	200,346
48×41	$45{,}192 \times 18{,}235$	1,157	244,398
54×45	$53{,}741 \times 21{,}196$	1,281	284,466
60×48	$60{,}844 \times 23{,}890$	1,382	318,354

Table 1: Problem sizes for the WSJ10 training corpus.

basis / n	5	10	15	20
1×11	$1.265\,10^{-3}$			
6×14	$7.06\,10^{-4}$	$6.92\,10^{-4}$		
12×18	$7.30\,10^{-4}$	$6.28\,10^{-4}$	$6.01\,10^{-4}$	
18×22	$7.31\,10^{-4}$	$6.29\,10^{-4}$	$5.84\,10^{-4}$	$5.59\,10^{-4}$
24×26	$7.35\,10^{-4}$	$6.39\,10^{-4}$	$5.88\,10^{-4}$	$5.31\,10^{-4}$
30×29	$7.34\,10^{-4}$	$6.41\,10^{-4}$	$5.86\,10^{-4}$	$5.30\,10^{-4}$

Table 2: Experiments with the unsupervised spectral method on the WSJ10 corpus. Results are in terms of expected L_1 on the training set, for different basis and numbers of states.

We generate a training set by sampling 4,000 strings from the target PCFG and counting the relative frequency of each. For the supervised model, we generate strings paired with their context-free derivation. To measure the quality of the learned models, we use the L_1 distance to the target distribution over a fixed set of strings $\Sigma^{\leq n}$, for $n = 7$.[1]

Figure 3 shows the results for the different models and for different basis sizes (in terms of the basis factor f). Here we can clearly see that the WCFG models, even the unsupervised one, outperform the WFA in reproducing the target distribution.

5.3 Natural Language Experiments

Now we present some preliminar tests using natural language data. For these tests, we used the WSJ10 subset of the Penn Treebank, as Klein and Manning (2002). This dataset consists of the sentences of length ≤ 10 after filtering punctuation and currency. We removed lexical items and mapped the POS tags

[1] Given two functions f_1 and f_2 over strings, the L_1 distance is the sum of the absolute difference over all strings in a set: $\sum_x |f_1(x) - f_2(x)|$.

to the Universal Part-of-Speech Tagset (Petrov et al., 2012), reducing the alphabet to a set of 11 symbols.

Table 1 shows the size of the problem for different basis sizes. As described in the previous subsection for the unsupervised case, we obtain the basis by taking the most frequent observed substrings and contexts. We then compute all yields that can be generated with this basis, and close the basis to include all possible insides and outsides with operations completions, such that we create a Hankel as described in Section 4.1. Table 1 shows, for each base, the size of $\boldsymbol{H}$ we induce, the number of observable constraints (i.e. sentences we train from), and the number of inside-outside constraints.

With the current implementation of the optimizer we were only able to run the unsupervised learning for small basis sizes. Table 2 shows the expected L_1 on training data. For a fixed basis, as we increase the number of states we see that the error decreases, showing that the method is inducing a Hankel matrix that explains the observable statistics.

6 Conclusions

We have presented a novel approach for unsupervised learning of WCFG. Our method combines ingredients of spectral learning with low-rank convex optimization methods.

Our method optimizes over a matrix that, even if it grows polynomially with respect to the size of training, results in a large problem. To scale the method to learn languages of the complexity of natural languages we would need to identify optimization algorithms specially suited for this problem.

A Proofs

A.1 Proof of Inside-Outside Eq. 16 and 17

For the inside function, the base case is trivial. By induction:

$$\begin{aligned}
\beta_G(x)[i] &= \sum_{x=x_1x_2} \boldsymbol{A}(\beta_G(x_1) \otimes \beta_G(x_2))[i] \\
&= \sum_{\substack{j,k\in V \\ x=x_1x_2}} \boldsymbol{A}[i,j,k] \cdot \beta_G(x_1)[j] \cdot \beta_G(x_2)[k] \\
&= \sum_{\substack{j,k\in V \\ x=x_1x_2}} w_R(i \to j\ k) \cdot \bar{\beta}_{\bar{G}}(j \overset{\star}{\Rightarrow} x_1) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2) \\
&= \bar{\beta}_{\bar{G}}(i \overset{\star}{\Rightarrow} x)
\end{aligned}$$

For the outside function, let $\boldsymbol{e}_i$ be an n-dimensional vector with coordinate i to 1 and the rest to 0. We reformulate the mapping as:

$$\alpha_G(x;z)^\top \boldsymbol{e}_i = \bar{\alpha}_{\bar{G}}(x;i;z) \tag{30}$$

The base case is trivial by definitions. We use the property of Kronecker products that $(\boldsymbol{v} \otimes \boldsymbol{I}_n)\boldsymbol{v}' = (\boldsymbol{v} \otimes \boldsymbol{v}')$ and $(\boldsymbol{I}_n \otimes \boldsymbol{v})\boldsymbol{v}' = (\boldsymbol{v}' \otimes \boldsymbol{v})$ for $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^n$. We first look at one of the terms of $\alpha_G(x;z)^\top \boldsymbol{e}_i$:

$$\begin{aligned}
&\alpha_G(x_1;z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{I}_n)\boldsymbol{e}_i \\
&= \alpha_G(x_1;z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{e}_i) \\
&= \sum_{j,k\in V} (\alpha_G(x_1;z)^\top \boldsymbol{e}_j) \cdot \boldsymbol{A}[j,k,i] \cdot \beta_G(x_2)[k] \\
&= \sum_{j,k\in V} \bar{\alpha}_{\bar{G}}(x_1;j;z) \cdot w_R(j \to k\ i) \cdot \bar{\beta}_{\bar{G}}(k \overset{\star}{\Rightarrow} x_2)
\end{aligned}$$

Applying the distributive property in $\alpha_G(x;z)^\top \boldsymbol{e}_i$ it is easy to see that all terms are mapped to the corresponding term in $\bar{\alpha}_{\bar{G}}(x;i;z)$.

A.2 Proof of Lemma 1

Let $G' = \langle \boldsymbol{\alpha}'_\star, \{\boldsymbol{\beta}'_\sigma\}, \boldsymbol{A}' \rangle$ be a WCFG for f that induces a rank factorization $\boldsymbol{H} = \boldsymbol{P}'\boldsymbol{S}'$. We first show that there exists an invertible matrix $\boldsymbol{M}$ that changes the basis of the operators of G into those of G'. Define $\boldsymbol{M} = \boldsymbol{S}'\boldsymbol{S}^+$ and note that $\boldsymbol{P}^+\boldsymbol{P}'\boldsymbol{S}'\boldsymbol{S}^+ = \boldsymbol{P}^+\boldsymbol{H}\boldsymbol{S}^+ = \boldsymbol{I}$ implies that $\boldsymbol{M}$ is invertible with $\boldsymbol{M}^{-1} = \boldsymbol{P}^+\boldsymbol{P}'$. We now check that the operators of G correspond to the operators of G' under this change of basis. First we see that

$$\begin{aligned}
\boldsymbol{A} &= \boldsymbol{P}^+\boldsymbol{H}_A(\boldsymbol{S} \otimes \boldsymbol{S})^+ \\
&= \boldsymbol{P}^+\boldsymbol{P}'\boldsymbol{A}'(\boldsymbol{S}' \otimes \boldsymbol{S}')(\boldsymbol{S} \otimes \boldsymbol{S})^+ \\
&= \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{S}'\boldsymbol{S}^+ \otimes \boldsymbol{S}'\boldsymbol{S}^+) \\
&= \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{M} \otimes \boldsymbol{M}) \quad .
\end{aligned}$$

Now, since $\boldsymbol{h}_\star = \boldsymbol{\alpha}'^\top_\star \boldsymbol{S}'$ and $\boldsymbol{h}_\sigma = \boldsymbol{P}'\boldsymbol{\beta}'_\sigma$, it follows that $\boldsymbol{\alpha}^\top_\star = \boldsymbol{\alpha}'^\top_\star \boldsymbol{M}$ and $\boldsymbol{\beta}_\sigma = \boldsymbol{M}^{-1}\boldsymbol{\beta}'_\sigma$.

Finally we check that G and G' compute the same function, namely $f(o,i) = \alpha_G(o)^\top \beta_G(i) = \alpha_{G'}(o)^\top \beta_{G'}(i)$. We first see that $\beta_G(x) =$

$\boldsymbol{M}^{-1}\beta_{G'}(x)$:

$$\beta_G(\sigma) = \boldsymbol{\beta}_\sigma = \boldsymbol{M}^{-1}\boldsymbol{\beta}'_\sigma \tag{31}$$

$$\beta_G(x) = \sum_{x=x_1x_2} \boldsymbol{A}(\beta_G(x_1) \otimes \beta_G(x_2)) \tag{32}$$

$$= \sum_{x=x_1x_2} \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{M} \otimes \boldsymbol{M})(\beta_G(x_1) \otimes \beta_G(x_2))$$

$$= \boldsymbol{M}^{-1} \sum_{x=x_1x_2} \boldsymbol{A}'(\boldsymbol{M}\beta_G(x_1) \otimes \boldsymbol{M}\beta_G(x_2))$$

$$= \boldsymbol{M}^{-1} \sum_{x=x_1x_2} \boldsymbol{A}'(\beta_{G'}(x_1) \otimes \beta_{G'}(x_2))$$

It can also be shown that $\alpha_G(x;z)^\top = \alpha_{G'}(x;z)^\top \boldsymbol{M}$. One must see that in any term:

$$\alpha_G(x_1;z)^\top \boldsymbol{A}(\beta_G(x_2) \otimes \boldsymbol{I}_n) \tag{33}$$
$$= \alpha_G(x_1;z)^\top \boldsymbol{M}^{-1}\boldsymbol{A}'(\boldsymbol{M} \otimes \boldsymbol{M})(\beta_G(x_2) \otimes \boldsymbol{I}_n)$$
$$= \alpha_{G'}(x_1;z)^\top \boldsymbol{A}'(\boldsymbol{M}\beta_G(x_2) \otimes \boldsymbol{M}\boldsymbol{I}_n)$$
$$= \alpha_{G'}(x_1;z)^\top \boldsymbol{A}'(\beta_{G'}(x_2) \otimes \boldsymbol{I}_n)\boldsymbol{M}$$

and the relation follows. Finally:

$$\alpha_G(x;z)^\top \beta_G(y) \tag{34}$$
$$= \alpha_{G'}(x;z)^\top \boldsymbol{M}\boldsymbol{M}^{-1}\beta_{G'}(y)$$
$$= \alpha_{G'}(x;z)^\top \beta_{G'}(y)$$

□

A.3 Proof of Lemma 2

We will use the following sub-blocks of $\boldsymbol{H}$:

- $\boldsymbol{H}_\varepsilon$ is the sub-block restricted to $\mathcal{O}^1 \times \mathcal{I}^1$, i.e. without compositions.
- $\boldsymbol{H}_A$ is the sub-block restricted to $\mathcal{O}^1 \times \mathcal{I}^2$, i.e. inside compositions.
- $\boldsymbol{H}'_A$ is the sub-block restricted to $\mathcal{O}^2 \times \mathcal{I}^1$, i.e. outside compositions.
- $\boldsymbol{h}_\star^\top \in \mathbb{R}^{\mathcal{I}^1}$ is the row of $\boldsymbol{H}_\varepsilon$ for $\langle\lambda;\lambda\rangle$.
- $\boldsymbol{h}_{(x)} \in \mathbb{R}^{\mathcal{O}^1}$ is the column of $\boldsymbol{H}_\varepsilon$ for (x).
- $\boldsymbol{h}_{(x_1,x_2)} \in \mathbb{R}^{\mathcal{O}^1}$ is the column of $\boldsymbol{H}_A$ for (x_1,x_2).
- $\boldsymbol{h}'_{\langle x;z\rangle} \in \mathbb{R}^{\mathcal{I}^1}$ is the row of $\boldsymbol{h}_\varepsilon$ for $\langle x;z\rangle$.
- $\boldsymbol{h}'_{\langle x_1,x_2;z\rangle}$ and $\boldsymbol{h}'_{\langle x;z_1,z_2\rangle}$ be the rows in $\mathbb{R}^{\mathcal{I}^1}$ of $\boldsymbol{h}'{}_A$ for $\langle x_1,x_2;z\rangle$ and $\langle x;z_1,z_2\rangle$).

One supposes that $\mathrm{rank}(\boldsymbol{H}_\varepsilon) = \mathrm{rank}(\boldsymbol{H})$. We define G as

$$\alpha_\star^\top = \boldsymbol{h}_\star^\top \boldsymbol{H}_\varepsilon^+, \beta_a = \boldsymbol{h}_{(a)}, \boldsymbol{A} = \boldsymbol{H}_A(\boldsymbol{H}_\varepsilon^+ \otimes \boldsymbol{H}_\varepsilon^+)$$

Lemma 3. *One has that* $\beta_G(x) = \boldsymbol{h}_{(x)}$, *and* $\beta_G(x_1,x_2) = \boldsymbol{h}_{(x_1,x_2)}$.

Proof. By induction. For sequences of size 1, one has $\beta_G(x) = \beta_x = \boldsymbol{h}_{(x)}$. For the recursive case, let $\boldsymbol{e}_{(x)}$ be a vector in $\mathbb{R}^{\mathcal{I}^1}$ with 1 in the coordinate of (x) in $\boldsymbol{H}_\varepsilon$. Let $\boldsymbol{e}_{(x,y)}$ be a vector in $\mathbb{R}^{\mathcal{I}^2}$ with 1 in the coordinate of (x,y) in $\boldsymbol{H}_A$. For $\beta_G(x,y)$, one has $\boldsymbol{H}_\varepsilon^+\beta_G(x) = \boldsymbol{e}_{(x)}$, and $\boldsymbol{H}_\varepsilon^+\beta_G(y) = \boldsymbol{e}_{(y)}$, thus $\boldsymbol{H}_\varepsilon^+\beta_G(x) \otimes \boldsymbol{H}_\varepsilon^+\beta_G(y) = \boldsymbol{e}_{(x,y)}$ and $\boldsymbol{H}_A(\boldsymbol{H}_\varepsilon^+\beta_G(x) \otimes \boldsymbol{H}_\varepsilon^+\beta_G(y)) = \boldsymbol{h}_{(x,y)}$. Finally, one has that $\beta_G(x) = \sum_{x=x_1x_2}\beta_G(x_1,x_2) = \sum_{x=x_1x_2}\boldsymbol{h}_{(x_1,x_2)} = \boldsymbol{h}_{(x_1x_2x_3)}$ by the equation (26). □

One has a symmetric result for outside vectors. We define G' as

$$\alpha_\star^\top = \boldsymbol{h}_\star^\top, \beta_a = \boldsymbol{H}_\varepsilon^+\boldsymbol{h}_{(a)}, \boldsymbol{A} = \boldsymbol{H}_\varepsilon^+\boldsymbol{H}_A$$

Lemma 4. *One has that* $\alpha_{G'}(\langle x;z\rangle)^\top = \boldsymbol{h}'_{\langle x;z\rangle}$, $\alpha_{G'}(\langle x_1,x_2;z\rangle)^\top = \boldsymbol{h}'_{\langle x_1,x_2;z\rangle}$ *and* $\alpha_{G'}(\langle x;z_1,z_2\rangle)^\top = \boldsymbol{h}'_{\langle x;z_1,z_2\rangle}$.

Proof. (Sketch) Equation (31) is used in the same way than (27) before. Equation (25) is used to ensure a link between $\boldsymbol{H}'_A$ and $\boldsymbol{H}_A$. □

Let g be the mapping computed by G and G'. One has that $g(o,i) = \alpha_{G'}(o)^\top\beta_{G'}(i) = \alpha_G(o)^\top\beta_G(i) = \alpha_{G'}(o)^\top\boldsymbol{H}_\varepsilon^+\beta_G(i) = \boldsymbol{H}_\varepsilon(o,i)$.

Acknowledgements We are grateful to Borja Balle and the anonymous reviewers for providing us with helpful comments. This work was supported by a Google Research Award, and by projects XLike (FP7-288342), ERA-Net CHISTERA VISEN, TACARDI (TIN2012-38523-C02-02), BASMATI (TIN2011-27479-C04-03), SGR-GPLN (2009-SGR-1082) and SGR-LARCA (2009-SGR-1428). Xavier Carreras was supported by the Ramón y Cajal program of the Spanish Government (RYC-2008-02223). Franco M. Luque was supported by the National University of Córdoba and by a Postdoctoral fellowship of CONICET, Argentinian Ministry of Science, Technology and Productive Innovation.

References

Pieter Adriaans, Marten Trautwein, and Marco Vervoort. 2000. Towards high speed grammar induction on large text corpora. In *SOFSEM 2000: Theory and Practice of Informatics*, pages 173–186. Springer.

Raphaël Bailly, Franois Denis, and Liva Ralaivola. 2009. Grammatical inference as a principal component analysis problem. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 33–40, Montreal, June. Omnipress.

Raphaël Bailly, Amaury Habrard, and François Denis. 2010. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of the 21st International Conference Algorithmic Learning Theory*, Lecture Notes in Computer Science, pages 74–88. Springer.

Raphaël Bailly, Xavier Carreras, and Ariadna Quattoni. 2013. Unsupervised spectral learning of finite-state transducers. In *Advances in Neural Information Processing Systems 26*.

James K. Baker. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Borja Balle, Ariadna Quattoni, and Xavier Carreras. 2011. A spectral learning algorithm for finite state transducers. In *Proceedings of ECML PKDD*, pages 156–171.

Borja Balle, Ariadna Quattoni, and Xavier Carreras. 2012. Local loss optimization in operator models: A new insight into spectral learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-2012)*, ICML '12, pages 1879–1886, New York, NY, USA, July. Omnipress.

Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March.

Stanley F Chen. 1995. Bayesian grammar induction for language modeling. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 228–235. Association for Computational Linguistics.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 13. Association for Computational Linguistics.

Alexander Clark. 2007. Learning deterministic context free grammars: The omphalos competition. *Machine Learning*, 66(1):93–110.

Shay B. Cohen, David M. Blei, and Noah A. Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572, Los Angeles, California, June. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable pcfgs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–231, Jeju Island, Korea, July. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia, June. Association for Computational Linguistics.

Matthew Gormley and Jason Eisner. 2013. Nonconvex global optimization for latent-variable models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria, August. 11 pages.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2009. A spectral algorithm for learning hidden markov models. In *Proceedings of the Annual Conference on Computational Learning Theory (COLT)*.

Daniel Hsu, Sham Kakade, and Percy Liang. 2012. Identifiability and unmixing of latent parse trees. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1520–1528.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Kenichi Kurihara and Taisuke Sato. 2006. Variational bayesian grammar induction for natural language. In *Grammatical Inference: Algorithms and Applications*, pages 84–96. Springer.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical dirichlet processes. In *EMNLP-CoNLL*, pages 688–697.

Franco M. Luque, Ariadna Quattoni, Borja Balle, and Xavier Carreras. 2012. Spectral learning for non-deterministic dependency parsing. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 409–419, Avignon, France, April. Association for Computational Linguistics.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware, USA, June. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC*, May.

Kewei Tu and Vasant Honavar. 2008. Unsupervised learning of probabilistic context-free grammar using iterative biclustering. In *Grammatical Inference: Algorithms and Applications*, pages 224–237. Springer.

Menno Van Zaanen. 2000. Abl: Alignment-based learning. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 961–967. Association for Computational Linguistics.

Identifying Phrasal Verbs Using Many Bilingual Corpora

Karl Pichotta*
Department of Computer Science
University of Texas at Austin
pichotta@cs.utexas.edu

John DeNero
Google, Inc.
denero@google.com

Abstract

We address the problem of identifying multiword expressions in a language, focusing on English phrasal verbs. Our *polyglot ranking* approach integrates frequency statistics from translated corpora in 50 different languages. Our experimental evaluation demonstrates that combining statistical evidence from many parallel corpora using a novel ranking-oriented boosting algorithm produces a comprehensive set of English phrasal verbs, achieving performance comparable to a human-curated set.

1 Introduction

A *multiword expression* (MWE), or *noncompositional compound*, is a sequence of words whose meaning cannot be composed directly from the meanings of its constituent words. These idiosyncratic phrases are prevalent in the lexicon of a language; Jackendoff (1993) estimates that their number is on the same order of magnitude as that of single words, and Sag et al. (2002) suggest that they are much more common, though quantifying them is challenging (Church, 2011). The task of identifying MWEs is relevant not only to lexical semantics applications, but also machine translation (Koehn et al., 2003; Ren et al., 2009; Pal et al., 2010), information retrieval (Xu et al., 2010; Acosta et al., 2011), and syntactic parsing (Sag et al., 2002). Awareness of MWEs has empirically proven useful in a number of domains: Finlayson and Kulkarni (2011), for example, use MWEs to attain a significant performance improvement in word sense disambiguation; Venkatapathy and Joshi (2006) use features associated with MWEs to improve word alignment.

We focus on a particular subset of MWEs, English *phrasal verbs*. A phrasal verb consists of a head verb followed by one or more particles, such that the meaning of the phrase cannot be determined by combining the simplex meanings of its constituent words (Baldwin and Villavicencio, 2002; Dixon, 1982; Bannard et al., 2003).[1] Examples of phrasal verbs include *count on* [*rely*], *look after* [*tend*], or *take off* [*remove*], the meanings of which do not involve counting, looking, or taking. In contrast, there are verbs followed by particles that are not phrasal verbs, because their meaning is compositional, such as *walk towards*, *sit behind*, or *paint on*.

We identify phrasal verbs by using frequency statistics calculated from parallel corpora, consisting of bilingual pairs of documents such that one is a translation of the other, with one document in English. We leverage the observation that a verb will translate in an atypical way when occurring as the head of a phrasal verb. For example, the word *look* in the context of *look after* will tend to translate differently from how *look* translates generally. In order to characterize this difference, we calculate a frequency distribution over translations of *look*, then compare it to the distribution of translations of *look* when followed by the word *after*. We expect that idiomatic phrasal verbs will tend to have unexpected translation of their head verbs, measured by the Kullback-Leibler divergence between those distributions.

Our *polyglot ranking* approach is motivated by the hypothesis that using many parallel corpora of different languages will help determine the degree of semantic idiomaticity of a phrase. In order to com-

*Research conducted during an internship at Google.

[1] Nomenclature varies: the term *verb-particle construction* is also used to denote what we call phrasal verbs; further, the term *phrasal verb* is sometimes used to denote a broader class of constructions.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 636–646,
Seattle, Washington, USA, 18-21 October 2013.

bine evidence from multiple languages, we develop a novel boosting algorithm tailored to the task of ranking multiword expressions by their degree of idiomaticity. We train and evaluate on disjoint subsets of the phrasal verbs in English Wiktionary[2]. In our experiments, the set of phrasal verbs identified automatically by our method achieves held-out recall that nears the performance of the phrasal verbs in WordNet 3.0, a human-curated set. Our approach strongly outperforms a monolingual system, and continues to improve when incrementally adding translation statistics for 50 different languages.

2 Identifying Phrasal Verbs

The task of identifying phrasal verbs using corpus information raises several issues of experimental design. We consider four central issues below in motivating our approach.

Types vs. Tokens. When a phrase is used in context, it takes a particular meaning among its possible senses. Many phrasal verbs admit compositional senses in addition to idiomatic ones—contrast idiomatic "*look down on him for his politics*" with compositional "*look down on him from the balcony*." In this paper, we focus on the task of determining whether a phrase type is a phrasal verb, meaning that it frequently expresses an idiomatic meaning across its many token usages in a corpus. We do not attempt to distinguish which individual phrase tokens in the corpus have idiomatic senses.

Ranking vs. Classification. Identifying phrasal verbs involves relative, rather than categorical, judgments: some phrasal verbs are more compositional than others, but retain a degree of noncompositionality (McCarthy et al., 2003). Moreover, a polysemous phrasal verb may express an idiosyncratic sense more or less often than a compositional sense in a particular corpus. Therefore, we should expect a corpus-driven system not to classify phrases as strictly idiomatic or compositional, but instead assign a ranking or relative scoring to a set of candidates.

Candidate Phrases. We distinguish between the task of identifying candidate multiword expressions and the task of ranking those candidates by their semantic idiosyncracy. With English phrasal verbs, it is straightforward to enumerate all desired verbs followed by one or more particles, and rank the entire set.

[2] http://en.wiktionary.org

Feature	*Description*
φ_L ($\times 50$)	KL Divergence for each language L
μ_1	frequency of phrase given verb
μ_2	PMI of verb and particles
μ_3	μ_1 with interposed pronouns

Table 1: Features used by the polyglot ranking system.

Using Parallel Corpora. There have been a number of approaches proposed for the use of multilingual resources for MWE identification (Melamed, 1997; Villada Moirón and Tiedemann, 2006; Caseli et al., 2010; Tsvetkov and Wintner, 2012; Salehi and Cook, 2013). Our approach differs from previous work in that we identify MWEs using translation distributions of verbs, as opposed to 1–1, 1–m, or m–n word alignments, most-likely translations, bilingual dictionaries, or distributional entropy. To the best of our knowledge, ours is the first approach to use translational distributions to leverage the observation that a verb typically translates differently when it heads a phrasal verb.

3 The Polyglot Ranking Approach

Our approach uses bilingual and monolingual statistics as features, computed over unlabeled corpora. Each statistic characterizes the degree of idiosyncrasy of a candidate phrasal verb, using a single monolingual or bilingual corpus. We combine features for many language pairs using a boosting algorithm that optimizes a ranking objective using a supervised training set of English phrasal verbs. Each of these aspects of our approach is described in detail below; for reference, Table 1 provides a list of the features used.

3.1 Bilingual Statistics

One of the intuitive properties of an MWE is that its individual words likely do not translate literally when the whole expression is translated into another language (Melamed, 1997). We capture this effect

by measuring the divergence between how a verb translates generally and how it translates when heading a candidate phrasal verb.

A *parallel corpus* is a collection of document pairs $\langle D_E, D_F \rangle$, where D_E is in English, D_F is in another language, one document is a translation of the other, and all documents D_F are in the same language. A *phrase-aligned parallel corpus* aligns those documents at a sentence, phrase, and word level. A phrase e aligns to another phrase f if some word in e aligns to some word in f and no word in e or f aligns outside of f or e, respectively. As a result of this definition, the words within an aligned phrase pair are themselves connected by word-level alignments.

Given an English phrase e, define $F(e)$ to be the set of all foreign phrases observed aligned to e in a parallel corpus. For any $f \in F(e)$, let $P(f|e)$ be the conditional probability of the phrase e translating to the phrase f. This probability is estimated as the relative frequency of observing f and e as an aligned phrase pair, conditioned on observing e aligned to any phrase in the corpus:

$$P(f|e) = \frac{N(e,f)}{\sum_{f'} N(e,f')}$$

with $N(e,f)$ the number of times e and f are observed occurring as an aligned phrase pair.

Next, we assign statistics to individual verbs within phrases. The first word of a candidate phrasal verb e is a verb. For a candidate phrasal verb e and a foreign phrase f, let $\pi_1(e,f)$ be the subphrase of f that is most commonly word-aligned to the first word of e. As an example, consider the phrase pair $e =$ *talk down to* and $f =$ *hablar con menosprecio*. Suppose that when e is aligned to f, the word *talk* is most frequently aligned to *hablar*. Then $\pi_1(e,f) =$ *hablar*.

For a phrase e and its set $F(e)$ of aligned translations, we define the *constituent translation probability* of a foreign subphrase x as:

$$P_e(x) = \sum_{f \in F(e)} P(f|e) \cdot \delta\left(\pi_1(e,f), x\right) \quad (1)$$

where δ is the Kronecker delta function, taking value 1 if its arguments are equal and 0 otherwise. Intuitively, P_e assigns the probability mass for every f to its subphrase most commonly aligned to the verb in e. It expresses how this verb is translated in the context of a phrasal verb construction.[3] Equation (1) defines a distribution over all phrases x of a foreign language.

We also assign statistics to verbs as they are translated outside of the context of a phrase. Let $v(e)$ be the verb of a phrasal verb candidate e, which is always its first word. For a single-word verb phrase $v(e)$, we can compute the constituent translation probability $P_{v(e)}(x)$, again using Equation (1). The difference between $P_e(x)$ and $P_{v(e)}(x)$ is that the latter sums over all translations of the verb $v(e)$, regardless of whether it appears in the context of e:

$$P_{v(e)}(x) = \sum_{f \in F(v(e))} P(f|v(e)) \cdot \delta\left(\pi_1(v(e),f), x\right)$$

For a one-word phrase such as $v(e)$, $\pi_1(v(e),f)$ is the subphrase of f that most commonly directly word-aligns to the one word of $v(e)$.

Finally, for a phrase e and its verb $v(e)$, we calculate the Kullback-Leibler (KL) divergence between the translation distribution of $v(e)$ and e:

$$D_{KL}\left(P_{v(e)} \| P_e\right) = \sum_x P_{v(e)}(x) \ln \frac{P_{v(e)}(x)}{P_e(x)} \quad (2)$$

where the sum ranges over all x such that $P_{v(e)}(x) > 0$. This quantifies the difference between the translations of e's verb when it occurs in e, and when it occurs in general. Figure 1 illustrates this computation on a toy corpus.

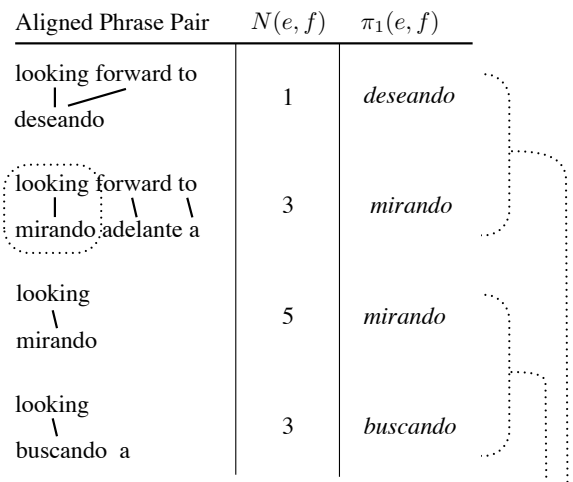
Smoothing. Equation (2) is defined only if, for every x such that $P_{v(e)}(x) > 0$, it is also the case that $P_e(x) > 0$. In order to ensure that this condition holds, we smooth the translation distributions toward uniform. Let D be the set of phrases with non-zero probability under either distribution:

$$D = \{x : P_{v(e)}(x) > 0 \text{ or } P_e(x) > 0\}$$

Then, let U_D be the uniform distribution over D:

$$U_D(x) = \begin{cases} 1/|D| & \text{if } x \in D \\ 0 & \text{if } x \notin D \end{cases}$$

[3] To extend this statistic to other types of multiword expressions, one could compute a similar distribution for other content words in the phrase.



Aligned Phrase Pair	$N(e,f)$	$\pi_1(e,f)$
looking forward to / deseando	1	*deseando*
looking forward to / mirando adelante a	3	*mirando*
looking / mirando	5	*mirando*
looking / buscando a	3	*buscando*

	mirando	*deseando*	*buscando*
$P_{v(e)}(x)$	$\frac{5}{8} = 0.625$	0	$\frac{3}{8} = 0.375$
$P'_{v(e)}(x)$	0.610	0.02	0.373
$P_e(x)$	$\frac{3}{4} = 0.75$	$\frac{1}{4} = 0.25$	0
$P'_e(x)$	0.729	0.254	0.02

$$D_{KL}(P'_{v(e_i)} \| P'_{e_i}) = -0.109 + -0.045 + 1.159 = 1.005$$

Figure 1: The computation of $D_{KL}(P'_{v(e_i)} \| P'_{e_i})$ using a toy corpus, for $e =$ *looking forward to*. Note that the second aligned phrase pair contains the third, so the second's count of 3 must be included in the third's count of 5.

When computing divergence in Equation (2), we use the smoothed distributions P'_e and $P'_{v(e)}$:

$$P'_e(x) = \alpha P_e(x) + (1-\alpha)U_D(x)$$
$$P'_{v(e)}(x) = \alpha P_{v(e)}(x) + (1-\alpha)U_D(x).$$

We use $\alpha = 0.95$, which distributes 5% of the total probability mass evenly among all events in D.

Morphology. We calculate statistics for morphological variants of an English phrase. For a candidate English phrasal verb e (for example, *look up*), let E denote the set of inflections of that phrasal verb (for *look up*, this will be [*look*|*looks*|*looked*|*looking*] *up*). We extract the variants in E from the verb entries in English Wiktionary. The final score computed from a phrase-aligned parallel corpus translating English sentences into a language L is the average KL divergence of smoothed constituent translation distributions for any inflected form $e_i \in E$:

$$\varphi_L(e) = \frac{1}{|E|} \sum_{e_i \in E} D_{KL}\left(P'_{v(e_i)} \| P'_{e_i}\right)$$

3.2 Monolingual Statistics

We also collect a number of monolingual statistics for each phrasal verb candidate, motivated by the considerable body of previous work on the topic (Church and Hanks, 1990; Lin, 1999; McCarthy et al., 2003). The monolingual statistics are designed to identify frequent collocations in a language. This set of monolingual features is not comprehensive, as we focus our attention primarily on bilingual features in this paper.

As above, define E to be the set of morphologically inflected variants of a candidate e, and let V be the set of inflected variants of the head verb $v(e)$ of e. We define three statistics calculated from the phrase counts of a monolingual English corpus. First, we define $\mu_1(e)$ to be the relative frequency of the candidate e, given e's head verb, summed over morphological variants:

$$\begin{aligned}\mu_1(e) &= \ln P(E|V) \\ &= \ln \frac{\sum_{e_i \in E} N(e_i)}{\sum_{v_i \in V} N(v_i)}\end{aligned}$$

where $N(x)$ is the number of times phrase x was observed in the monolingual corpus.

Second, define $\mu_2(e)$ to be the pointwise mutual information (PMI) between V (the event that one of the inflections of the verb in e is observed) and R, the event of observing the rest of the phrase:

$$\begin{aligned}&\mu_2(e) \\ &= \text{PMI}(V, R) \\ &= \lg P(V,R) - \lg\left(P(V)P(R)\right) \\ &= \lg P(E) - \lg\left(P(V)P(R)\right) \\ &= \lg \sum_{e_i \in E} N(e_i) - \lg \sum_{v_i \in V} N(v_i) - \lg N(r) + \lg N\end{aligned}$$

where N is the total number of tokens in the corpus, and logarithms are base-2. This statistic characterizes the degree of association between a verb and its phrasal extension. We only calculate μ_2 for two-word phrases, as it did not prove helpful for longer phrases.

Finally, define $\mu_3(e)$ to be the relative frequency of the phrasal verb e augmented by an accusative pronoun, conditioned on the verb. Let A be the set of phrases in E with an accusative pronoun (*it, them, him, her, me, you*) optionally inserted either at the end of the phrase or directly after the verb. For $e =$ *look up*, $A = \{$*look up, look X up, look up X, looks up, looks X up, looks up X,* $\dots\}$, with X an accusative pronoun. The μ_3 statistic is similar to μ_1, but allows for an intervening or following pronoun:

$$\begin{aligned}\mu_3(e) &= \ln P(A|V) \\ &= \ln \frac{\sum_{e_i \in A} N(e_i)}{\sum_{v_i \in V} N(v_i)}.\end{aligned}$$

This statistic is designed to exploit the intuition that phrasal verbs frequently have accusative pronouns either inserted into the middle (e.g. *look it up*) or at the end (e.g. *look down on him*).

3.3 Ranking Phrasal Verb Candidates

Our goal is to assign a single real-valued score to each candidate e, by which we can rank candidates according to semantic idiosyncrasy. For each language L for which we have a parallel corpus, we defined, in section 3.1, a function $\varphi_L(e)$ assigning real values to candidate phrasal verbs e, which we hypothesize is higher on average for more idiomatic compounds. Further, in section 3.2, we defined real-valued monolingual functions μ_1, μ_2, and μ_3 for which we hypothesize the same trend holds. Because each score individually ranks all candidates, it is natural to view each φ_L and μ_i as a weak ranking function that we can combine with a supervised boosting objective. We use a modified version of AdaBoost (Freund and Schapire, 1995) that optimizes for recall.

For each φ_L and μ_i, we compute a ranked list of candidate phrasal verbs, ordered from highest to lowest value. To simplify learning, we consider only the top 5000 candidate phrasal verbs according to μ_1, μ_2, and μ_3. This pruning procedure excludes candidates that do not appear in our monolingual corpus.

We optimize the ranker using an unranked, incomplete training set of phrasal verbs. We can evaluate the quality of the ranker by outputting the top N ranked candidates and measuring recall relative to this gold-standard training set. We choose this recall-at-N metric so as to not directly penalize precision errors, as our training set is incomplete.

Algorithm 1 Recall-Oriented Ranking AdaBoost

for $i = 1 : |X|$ **do**
  $w[i] \leftarrow 1/|X|$
end for
for $t = 1 : T$ **do**
  for all $h \in \mathcal{H}$ **do**
    $\epsilon_h \leftarrow 0$
    for $i = 1 : |X|$ **do**
      if $x_i \notin h$ **then**
        $\epsilon_h \leftarrow \epsilon_h + w[i]$
      end if
    end for
  end for
  $h_t \leftarrow \text{argmax}_{h \in \mathcal{H}} |\epsilon_B - \epsilon_h|$
  $\alpha_t \leftarrow \ln(\epsilon_B / \epsilon_{h_t})$
  for $i = 1 : |X|$ **do**
    if $x_i \in h_t$ **then**
      $w[i] \leftarrow \frac{1}{Z} w[i] \exp(-\alpha_t)$
    else
      $w[i] \leftarrow \frac{1}{Z} w[i] \exp(\alpha_t)$
    end if
  end for
end for

Define $\mathcal{H}$ to be the set of N-element sets containing the top proposals for each weak ranker (we use $N = 2000$). That is, each element of $\mathcal{H}$ is a set containing the 2000 highest values for some φ_L or μ_i. We define the baseline error ϵ_B to be $1 - \mathbb{E}[R]$, with R the recall-at-N of a ranker ordering the candidate phrases in the set $\cup\mathcal{H}$ at random. The value $\mathbb{E}[R]$ is estimated by averaging the recall-at-N of 1000 random orderings of $\cup\mathcal{H}$.

Algorithm 1 gives the formulation of the AdaBoost training algorithm that we use to combine weak rankers. The algorithm maintains a weight vector w (summing to 1) which contains a positive real number for each gold standard phrasal verb in the training set X. Initially, w is uniformly set to $1/|X|$. At each iteration of the algorithm, w is modified to take higher values for recently misclassified examples. We repeatedly choose weak rankers $h_t \in \mathcal{H}$ (and corresponding real-valued coefficients α_t) that correctly rank examples with high w values.

Lines 5–12 of Algorithm 1 calculate the weighted error values ϵ_h for every weak ranker set $h \in \mathcal{H}$. The error ϵ_h will be 1 if h contains none of X and 0 if h contains all of X, as w always sums to 1. Line 13 picks the ranker $h_t \in \mathcal{H}$ whose weighted error is as far as possible from the random baseline error ϵ_B. Line 14 calculates a coefficient α_t for h_t, which will be positive if $\epsilon_{h_t} < \epsilon_B$ and negative if $\epsilon_{h_t} > \epsilon_B$. Intuitively, α_t encodes the importance of h_t—it will be high if h_t performs well, and low if it performs poorly. The Z in lines 17 and 19 is the normalizing constant ensuring the vector w sums to 1.

After termination of Algorithm 1, we have weights $\alpha_1, \ldots, \alpha_T$ and lists $h_1, \ldots, h_T$. Define f_t as the function that generated the list h_t (each f_t will be some φ_L or μ_i). Now, we define a final combined function φ, taking a phrase e and returning a real number:

$$\varphi(e) = \sum_{t=1}^{T} \alpha_t f_t(e).$$

We standardize the scores of individual weak rankers to have mean 0 and variance 1, so that their scores are comparable.

The final learned ranker outputs a real value, instead of the class labels frequently found in AdaBoost. This follows previous work using boosting for learning to rank (Freund et al., 2003; Xu and Li, 2007). Our algorithm differs from previous methods because we are seeking to optimize for Recall-at-N, rather than a ranking loss.

4 Experimental Evaluation

4.1 Training and Test Set

In order to train and evaluate our system, we construct a gold-standard list of phrasal verbs from the freely available English Wiktionary. We gather phrasal verbs from three sources within Wiktionary:

1. Entries labeled as *English phrasal verbs*[4],
2. Entries labeled as *English idioms*[5], and
3. The *derived terms*[6] of English verb entries.

[4] http://en.wiktionary.org/wiki/Category:English_phrasal_verbs

[5] http://en.wiktionary.org/wiki/Category:English_idioms

[6] For example, see http://en.wiktionary.org/wiki/take#Derived_terms

about	*across*	*after*	*against*	*along*
among	*around*	*at*	*before*	*behind*
between	*beyond*	*by*	*down*	*for*
from	*in*	*into*	*like*	*off*
on	*onto*	*outside*	*over*	*past*
round	*through*	*to*	*towards*	*under*
up	*upon*	*with*	*within*	*without*

Table 2: Particles and prepositions allowed in phrasal verbs gathered from Wiktionary.

Many of the idioms and derived terms are not phrasal verbs (e.g. *kick the bucket*, *make-or-break*). We filter out any phrases not of the form VP^+, with V a verb, and P^+ denoting one or more occurrences of particles and prepositions from the list in Table 2. We omit prepositions that do not productively form English phrasal verbs, such as *amid* and *as*. This process also omits some compounds that are sometimes called phrasal verbs, such as light verb constructions, e.g. *have a go* (Butt, 2003), and noncompositional verb-adverb collocations, e.g. *look forward*.

There are a number of extant phrasal verb corpora. For example, McCarthy et al. (2003) present graded human compositionality judgments for 116 phrasal verbs, and Baldwin (2008) presents a large set of candidates produced by an automated system, with false positives manually removed. We use Wiktionary instead, in an attempt to construct a maximally comprehensive data set that is free from any possible biases introduced by automatic extraction processes.

4.2 Filtering and Data Partition

The merged list of phrasal verbs extracted from Wiktionary included some common collocations that have compositional semantics (e.g. *know about*), as well as some very rare constructions (e.g. *cheese down*). We removed these spurious results systematically by filtering out very frequent and very infrequent entries. First, we calculated the log probability of each phrase, according to a language model built from a large monolingual corpus of news documents and web documents, smoothed with stupid backoff (Brants et al., 2007). We sorted all Wiktionary phrasal verbs according to this value. Then, we selected the contiguous 75% of the sorted phrases that minimize the variance of this statistic. This method

		Recall-at-1220	
		Dev	Test
Baseline	Frequent Candidates	17.0	19.3
	WordNet 3.0 Frequent	41.6	43.7
	WordNet 3.0 Filtered	49.4	**48.8**
Boosted	Monolingual Only	30.1	30.2
	Bilingual Only	47.1	43.9
	Monolingual+Bilingual	**50.8**	47.9

Table 3: Our boosted ranker combining monolingual and bilingual features (bottom) compared to three baselines (top) gives comparable performance to the human-curated upper bound.

removed a few very frequent phrases and a large number of rare phrases. The remaining phrases were split randomly into a development set of 694 items and a held-out test set of 695 items.

4.3 Corpora

Our monolingual English corpus consists of news articles and documents collected from the web. Our parallel corpora from English to each of 50 languages also consist of documents collected from the web via distributed data mining of parallel documents based on the text content of web pages (Uszkoreit et al., 2010).

The parallel corpora were segmented into aligned sentence pairs and word-aligned using two iterations of IBM Model 1 (Brown et al., 1993) and two iterations of the HMM-based alignment model (Vogel et al., 1996) with posterior symmetrization (Liang et al., 2006). This training recipe is common in large-scale machine translation systems.

4.4 Generating Candidates

To generate the set of candidate phrasal verbs considered during evaluation, we exhaustively enumerated the Cartesian product of all verbs present in the previously described Wiktionary set ($\mathcal{V}$), all particles in Table 2 ($\mathcal{P}$) and a small set of second particles $\mathcal{T} = \{$*with, to, on,* $\epsilon\}$, with ϵ the empty string. The set of candidate phrasal verbs we consider during evaluation is the product $\mathcal{V} \times \mathcal{P} \times \mathcal{T}$, which contains 96,880 items.

4.5 Results

We optimize a ranker using the boosting algorithm described in section 3.3, using the features from Table 1, optimizing performance on the Wiktionary development set described in section 4.2. Monolingual and bilingual statistics are calculated using the corpora described in section 4.3, with candidate phrasal verbs being drawn from the set described in section 4.4.

We evaluate our method of identifying phrasal verbs by computing *recall-at-*N. This statistic is the fraction of the Wiktionary test set that appears in the top N proposed phrasal verbs by the method, where N is an arbitrary number of top-ranked candidates held constant when comparing different approaches (we use $N = 1220$). We do not compute precision, because the test set to which we compare is not an exhaustive list of phrasal verbs, due to the development/test split, frequency filtering, and omissions in the original lexical resource. Proposing a phrasal verb not in the test set is not necessarily an error, but identifying many phrasal verbs from the test set is an indication of an effective method. Recall-at-N is a natural way to evaluate a ranking system where the gold-standard data is an incomplete, unranked set.

Table 3 compares our approach to three baselines using the Recall-at-1220 metric evaluated on both the development and test sets. As a lower bound, we evaluated the 1220 most frequent candidates in our Monolingual corpus (*Frequent Candidates*).

As a competitive baseline, we evaluated the set of phrasal verbs in WordNet 3.0 (Fellbaum, 1998). We selected the most frequent 1220 out of 1781 verb-particle constructions in WordNet (*WordNet 3.0 Frequent*). A stronger baseline resulted from applying the same filtering procedure to WordNet that we did to Wiktionary: sorting all verb-particle entries by their language model score and retaining the 1220 consecutive entries that minimized language model variance (*WordNet 3.0 Filtered*). WordNet is a human-curated resource, and yet its recall-at-N compared to our Wiktionary test set is only 48.8%, indicating substantial divergence between the two resources. Such divergence is typical: lexical resources often disagree about what multiword expressions to include (Lin, 1999).

The three final lines in Table 3 evaluate our

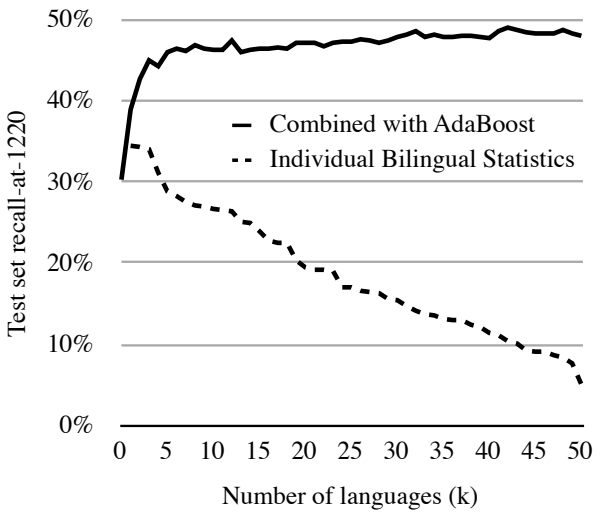


Figure 2: The solid line shows recall-at-1220 when combining the k best-performing bilingual statistics and three monolingual statistics. The dotted line shows the individual performance of the kth best-performing bilingual statistic, when applied in isolation to rank candidates.

boosted ranker. Automatically detecting phrasal verbs using monolingual features alone strongly outperformed the frequency-based lower bound, but underperformed the WordNet baseline. Bilingual features, using features from 50 languages, proved substantially more effective. The combination of both types of features yielded the best performance, outperforming the human-curated WordNet baseline on the development set (on which our ranker was optimized) and approaching its performance on the held-out test set.

4.6 Feature Analysis

The solid line in Figure 2 shows the recall-at-1220 for a boosted ranker using all monolingual statistics and k bilingual statistics, for increasing k. Bilingual statistics are added according to their individual recall, from best-performing to worst. That is, the point at $k = 0$ uses only μ_1, μ_2, and μ_3, the point at $k = 1$ adds the best individually-performing bilingual statistic (Spanish) as a weak ranker, the next point adds the second-best bilingual statistic (German), etc. Boosting maximizes performance on the development set, and evaluation is performed on the test set. We use $T = 53$ (equal to the total number of weak rankers).

	Recall-at-1220	
	Dev	Test
Bilingual only	47.1	43.9
Bilingual+μ_1	48.1	46.9
Bilingual+μ_2	50.1	48.3
Bilingual+μ_3	48.4	46.3
Bilingual+$\mu_1 + \mu_2$	50.2	47.9
Bilingual+$\mu_1 + \mu_3$	49.0	47.4
Bilingual+$\mu_2 + \mu_3$	50.4	**49.4**
Bilingual+$\mu_1 + \mu_2 + \mu_3$	**50.8**	47.9

Table 4: An ablation of monolingual statistics shows that they are useful in addition to the 50 bilingual statistics combined, and no single statistic provides maximal performance.

The dotted line in Figure 2 shows that individual bilingual statistics have recall-at-1220 ranging from 34.4% to 5.0%. This difference reflects the different sizes of parallel corpora and usefulness of different languages in identifying English semantic idiosyncrasy. Combining together the signal of multiple languages is clearly beneficial, and including many low-performing languages still offers overall improvements.

Table 4 shows the effect of adding different subsets of the monolingual statistics to the set of all 50 bilingual statistics. Monolingual statistics give a performance improvement of up to 5.5% recall on the test set, but the comparative behavior of the various combinations of the μ_i is somewhat unpredictable when training on the development set and evaluating on the test set. The pointwise mutual information of a verb and its particles (μ_2) appears to be the most useful feature. In fact, the test set performance of using μ_2 alone outperforms the combination of all three. The best combination even outperforms the WordNet 3.0 baseline on the test set, though optimizing on the development set would not select this model.

4.7 Error Analysis

Table 5 shows the 100 highest ranked phrasal verb candidates by our system that do not appear in either the development or test sets. Most of these candidates are in fact English phrasal verbs that happened to be missing from Wiktionary; some are present in Wiktionary but were removed from the reference

pick up	*pat on*	*tap into*	*fit for*	*charge with*	*suit against*
catch up	*burst into*	*muck up*	*haul up*	*give up*	*get off*
get through	*get up*	*get in*	*tack on*	*buzz about*	*do like*
plump for	*haul in*	*keep up with*	*strap on*	*catch up with*	*suck into*
get round	*chop off*	*slap on*	*pitch into*	*get into*	*inquire into*
drop behind	*get on*	*catch up on*	*pass on*	*cue from*	*carry around*
get around	*get over*	*shoot at*	*pick over*	*shoot by*	*shoot in*
make up to	*get past*	*cast down*	*set up with*	*rule off*	*hand round*
piss on	*hit by*	*break down*	*move for*	*lead off*	*pluck off*
flip through	*edge over*	*strike off*	*plug into*	*keep up*	*go past*
set off	*pull round*	*see about*	*stay on*	*put up*	*sidle up to*
buzz around	*take off*	*set up*	*slap in*	*head towards*	*shoot past*
inquire for	*tuck up*	*lie with*	*well before*	*go on with*	*reel from*
drive along	*snap off*	*barge into*	*whip on*	*put down*	*instance through*
bar from	*cut down on*	*let in*	*tune in to*	*move off*	*suit in*
lean against	*well beyond*	*get down to*	*go across*	*sail into*	*lie over*
hit with	*chow down on*	*look after*	*catch at*		

Table 5: The highest ranked phrasal verb candidates from our full system that do not appear in either Wiktionary set. Candidates are presented in decreasing rank; "pat on" is the second highest ranked candidate.

sets during filtering, and the remainder are in fact not phrasal verbs (true precision errors).

These errors fall largely into two categories. Some candidates are compositional, but contain polysemous verbs, such as *hit by*, *drive along*, and *head towards*. In these cases, prepositions disambiguate the verb, which naturally affects translation distributions. Other candidates are not phrasal verbs, but instead phrases that tend to have a different syntactic role, such as *suit against*, *instance through*, *fit for*, and *lie over* (conjugated as *lay over*). A careful treatment of part-of-speech tags when computing corpus statistics might address this issue.

5 Related Work

The idea of using word-aligned parallel corpora to identify idiomatic expressions has been pursued in a number of different ways. Melamed (1997) tests candidate MWEs by collapsing them into single tokens, training a new translation model with these tokens, and using the performance of the new model to judge candidates' noncompositionality. Villada Moirón and Tiedemann (2006) use word-aligned parallel corpora to identify Dutch MWEs, testing the assumption that the distributions of alignments of MWEs will generally have higher entropies than those of fully compositional compounds. Caseli et al. (2010) generate candidate multiword expressions by picking out sufficiently common phrases that align to single target-side tokens. Tsvetkov and Wintner (2012) generate candidate MWEs by finding one-to-one alignments in parallel corpora which are not in a bilingual dictionary, and ranking them based on monolingual statistics. The system of Salehi and Cook (2013) is perhaps the closest to the current work, judging noncompositionality using string edit distance between a candidate phrase's automatic translation and its components' individual translations. Unlike the current work, their method does not use distributions over translations or combine individual bilingual values with boosting; however, they find, as we do, that incorporating many languages is beneficial to MWE identification.

A large body of work has investigated the identification of noncompositional compounds from monolingual sources (Lin, 1999; Schone and Jurafsky, 2001; Fazly and Stevenson, 2006; McCarthy et al., 2003; Baldwin et al., 2003; Villavicencio, 2003). Many of these monolingual statistics could be viewed as weak rankers and fruitfully incorporated into our framework.

There has also been a substantial amount of work addressing the problem of differentiating between literal and idiomatic instances of phrases in context (Katz and Giesbrecht, 2006; Li et al., 2010;

Sporleder and Li, 2009; Birke and Sarkar, 2006; Diab and Bhutada, 2009). We do not attempt this task; however, techniques for token identification could be used to improve type identification (Baldwin, 2005).

6 Conclusion

We have presented the polyglot ranking approach to phrasal verb identification, using parallel corpora from many languages to identify phrasal verbs. We proposed an evaluation metric that acknowledges the inherent incompleteness of reference sets, but distinguishes among competing systems in a manner aligned to the goals of the task. We developed a recall-oriented learning method that integrates multiple weak ranking signals, and demonstrated experimentally that combining statistical evidence from a large number of bilingual corpora, as well as from monolingual corpora, produces the most effective system overall. We look forward to generalizing our approach to other types of noncompositional phrases.

Acknowledgments

Special thanks to Ivan Sag, who argued for the importance of handling multi-word expressions in natural language processing applications, and who taught the authors about natural language syntax once upon a time. We would also like to thank the anonymous reviewers for their helpful suggestions.

References

Otavio Acosta, Aline Villavicencio, and Viviane Moreira. 2011. Identification and treatment of multiword expressions applied to information retrieval. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Timothy Baldwin and Aline Villavicencio. 2002. Extracting the unextractable: A case study on verb-particles. In *Proceedings of the Sixth Conference on Natural Language Learning*.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Timothy Baldwin. 2005. Deep lexical acquisition of verb-particle constructions. *Computer Speech & Language, Special Issue on Multiword Expressions*.

Timothy Baldwin. 2008. A resource for evaluating the deep lexical acquisition of english verb-particle constructions. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions*.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *Proceedings of European Chapter of the Association for Computational Linguistics*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*.

Miriam Butt. 2003. The light verb jungle. In *Proceedings of the Workshop on Multi-Verb Constructions*.

Helena de Medeiros Caseli, Carlos Ramisch, Maria das Graças Volpe Nunes, and Aline Villavicencio. 2010. Alignment-based extraction of multiword expressions. *Language Resources and Evaluation*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1).

Kenneth Church. 2011. How many multiword expressions do people know? In *Proceedings of the ACL Workshop on Multiword Expressions*.

Mona T. Diab and Pravin Bhutada. 2009. Verb noun construction MWE token supervised classification. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Robert Dixon. 1982. The grammar of english phrasal verbs. *Australian Journal of Linguistics*.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

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

Mark Finlayson and Nidhi Kulkarni. 2011. Detecting multiword expressions improves word sense disambiguation. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Yoav Freund and Robert E. Schapire. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Conference on Computational Learning Theory*.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*.

Ray Jackendoff. 1993. *The Architecture of the Language Faculty*. MIT Press.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the Association for Computational Linguistics*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the Association for Computational Linguistics*.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL Workshop on Multiword Expressions*.

I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Santanu Pal, Sudip Kumar Naskar, Pavel Pecina, Sivaji Bandyopadhyay, and Andy Way. 2010. Handling named entities and compound verbs in phrase-based statistical machine translation. In *Proceedings of the COLING 2010 Workshop on Multiword Expressions*.

Zhixiang Ren, Yajuan Lu, Jie Cao, Qun Liu, and Yun Huang. 2009. Improving statistical machine translation using domain bilingual multiword expressions. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the CICLING Conference on Intelligent Text Processing and Computational Linguistics*.

Bahar Salehi and Paul Cook. 2013. Predicting the compositionality of multiword expressions using translations in multiple languages. In *Second Joint Conference on Lexical and Computational Semantics*.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Yulia Tsvetkov and Shuly Wintner. 2012. Extraction of multi-word expressions from small parallel corpora. In *Natural Language Engineering*.

Jakob Uszkoreit, Jay Ponte, Ashok Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the Conference on Computational Linguistics*.

Sriram Venkatapathy and Aravind K Joshi. 2006. Using information about multi-word expressions for the word-alignment task. In *Proceedings of the ACL Workshop on Multiword Expressions*.

Begoña Villada Moirón and Jörg Tiedemann. 2006. Identifying idiomatic expressions using automatic word-alignment. In *Proceedings of the EACL Workshop on Multiword Expressions in a Multilingual Context*.

Aline Villavicencio. 2003. Verb-particle constructions and lexical resources. In *Proceedings of the ACL workshop on Multiword expressions*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the Conference on Computational linguistics*.

Jun Xu and Hang Li. 2007. AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the SIGIR Conference on Research and Development in Information Retrieval*.

Ying Xu, Randy Goebel, Christoph Ringlstetter, and Grzegorz Kondrak. 2010. Application of the tightness continuum measure to chinese information retrieval. In *Proceedings of the COLING Workshop on Multiword Expressions*.

Deep Learning for Chinese Word Segmentation and POS Tagging

Xiaoqing Zheng
Fudan University
220 Handan Road
Shanghai, 200433, China
zhengxq@fudan.edu.cn

Hanyang Chen
Fudan University
220 Handan Road
Shanghai, 200433, China
chenhy12345@gmail.com

Tianyu Xu
Fudan University
220 Handan Road
Shanghai, 200433, China
xty213@gmail.com

Abstract

This study explores the feasibility of performing Chinese word segmentation (CWS) and POS tagging by deep learning. We try to avoid task-specific feature engineering, and use deep layers of neural networks to discover relevant features to the tasks. We leverage large-scale unlabeled data to improve internal representation of Chinese characters, and use these improved representations to enhance supervised word segmentation and POS tagging models. Our networks achieved close to state-of-the-art performance with minimal computational cost. We also describe a perceptron-style algorithm for training the neural networks, as an alternative to maximum-likelihood method, to speed up the training process and make the learning algorithm easier to be implemented.

1 Introduction

Word segmentation has been a long-standing challenge for the Chinese NLP community. It has received steady attention over the past two decades. Previous studies show that joint solutions usually lead to the improvement in accuracy over pipelined systems by exploiting POS information to help word segmentation and avoiding error propagation. However, traditional joint approaches usually involve a great number of features, which arises four limitations. First, the size of the result models is too large for practical use due to the storage and computing constraints of certain real-world applications. Second, the number of parameters is so large that the trained model is apt to overfit on training corpus. Third, a longer training time is required. Last but not the least, the decoding by dynamic programming technique might be intractable since a large search space is faced by the decoder.

The choice of features, therefore, is a critical success factor for these systems. Most of the state-of-the-art systems address their tasks by applying linear statistical models to the features carefully optimized for the tasks. This approach is effective because researchers can incorporate a large body of linguistic knowledge into the models. However, the approach does not scale well when it is used to perform more complex joint tasks, for example, the task of joint word segmentation, POS tagging, parsing, and semantic role labeling. A challenge for such a joint model is the large combined search space, which makes engineering effective task-specific features and structured learning of parameters very hard. Instead, we use multilayer neural networks to discover the useful features from the input sentences.

There are two main contributions in this paper. (1) We describe a perceptron-style algorithm for training the neural networks, which not only speeds up the training of the networks with negligible loss in performance, but also can be implemented more easily; (2) We show that the tasks of Chinese word segmentation and POS tagging can be effectively performed by the deep learning. Our networks achieved close to state-of-the-art performance by transferring the unsupervised internal representations of Chinese characters into the supervised models.

Section 2 presents the general architecture of neural networks, and our perceptron-style training algorithm for tagging. Section 3 describes how to

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 647–657,
Seattle, Washington, USA, 18-21 October 2013.

leverage large unlabeled data to obtain more useful character embeddings, and reports the experimental results of our systems. Section 4 presents a brief overview of related work. The conclusions are given in section 5.

2 The Neural Network Architecture

Chinese word segmentation and part-of-speech tagging tasks can be formulated as assigning labels to characters of an input sentence. The performance of the traditional tagging approaches is heavily dependent on the choice of features, for example, conditional random fields (CRFs), often with a set of feature templates. For that reason, much of the effort in designing such systems goes into the feature engineering, which is important but labor-intensive, mainly based on human ingenuity and linguistic intuition.

In order to make learning algorithms less dependent on the feature engineering, we chose to use a variant of the neural network architecture first proposed by (Bengio et al., 2003) for probabilistic language model, and reintroduced later by (Collobert et al., 2011) for multiple NLP tasks. The network takes the input sentence and discovers multiple levels of feature extraction from the inputs, with higher levels representing more abstract aspects of the inputs. The architecture is shown in Figure 1. The first layer extracts features for each Chinese character. The next layer extracts features from a window of characters. The following layers are classical neural network layers. The output of the network is a graph over which tag inference is achieved with a Viterbi algorithm.

2.1 Mapping Characters into Feature Vectors

The characters are fed into the network as indices that are used by a lookup operation to transform characters into their feature vectors. We consider a fixed-sized character dictionary $\mathcal{D}$[1]. The vector representations are stored in a character embedding matrix $\mathcal{M} \in \mathbb{R}^{d \times |\mathcal{D}|}$, where d is the dimensionality of the vector space (a hyper-parameter to be chosen) and $|\mathcal{D}|$ is the size of the dictionary.

[1]Unless otherwise specified, the character dictionary is extracted from the training set. Unknown characters are mapped to a special symbol that is not used elsewhere.

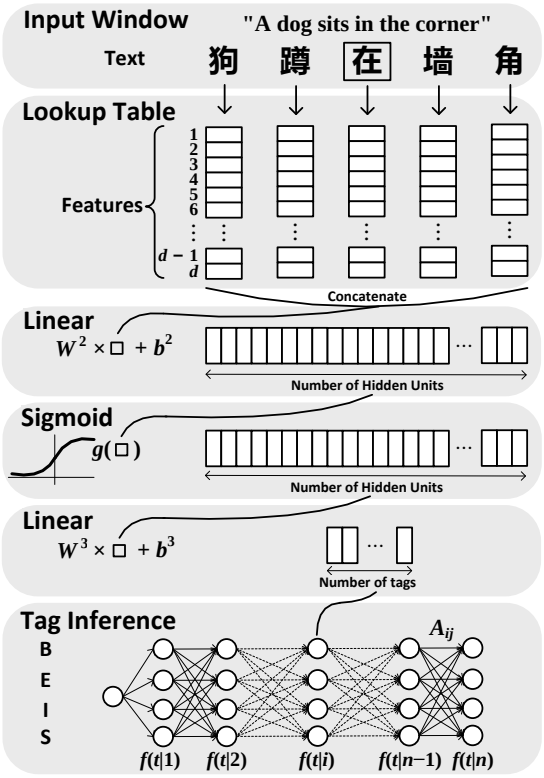


Figure 1: The neural network architecture.

Formally, assume we are given a Chinese sentence $c_{[1:n]}$ that is a sequence of n characters $c_i, 1 \leq i \leq n$. For each character $c_i \in \mathcal{D}$ that has an associated index k_i into the column of the embedding matrix, an d-dimensional feature vector representation is retrieved by the lookup table layer $\mathcal{Z}_{\mathcal{D}}(\cdot) \in \mathbb{R}^d$:

$$\mathcal{Z}_{\mathcal{D}}(c_i) = \mathcal{M} e_{k_i} \quad (1)$$

where we use a binary vector $e_{k_i} \in \mathbb{R}^{|\mathcal{D}| \times 1}$ which is zero in all positions except at the k_i-th index. The lookup operation can be seen as a simple projection layer. The feature vector of each character, starting from a random initialization, can be automatically trained by back propagation to be relevant to the task of interest.

In practice, it is common that one might want to provide other additional features that is thought to be helpful for the task. For example, for the name entity recognition task, one could provide a feature which says if a character is in a list of the common Chinese surnames or not. Another common practice is to introduce some statistics-based measures, such

as *boundary entropy* (Jin and Tanaka-Ishii, 2006) and *accessor variety* (Feng et al., 2004), which are commonly used in unsupervised CWS models. We associate a lookup table to each additional feature, and the character feature vector becomes the concatenation of the outputs of all these lookup tables.

2.2 Tag scoring

A neural network can be considered as a function $f_\theta(\cdot)$ with parameters θ. Any feed-forward neural network with L layers can be seen as a composition of functions $f_\theta^l(\cdot)$ defined for each layer l:

$$f_\theta(\cdot) = f_\theta^L(f_\theta^{L-1}(\dots f_\theta^1(\cdot)\dots)) \tag{2}$$

For each character in a sentence, a score is produced for every tag by applying several layers of the neural network over the feature vectors produced by the lookup table layer. We use a window approach to handle the sequences of variable sentence length. The window approach assumes that the tag of a character depends mainly on its neighboring characters. More precisely, given an input sentence $c_{[1:n]}$, we consider all successive windows of size w (a hyper-parameter), siding over the sentence, from character c_1 to c_n. At position c_i, the character feature window produced by the first lookup table layer can be written as:

$$f_\theta^1(c_i) = \begin{pmatrix} \mathcal{Z}_\mathcal{D}(c_{i-w/2}) \\ \vdots \\ \mathcal{Z}_\mathcal{D}(c_i) \\ \vdots \\ \mathcal{Z}_\mathcal{D}(c_{i+w/2}) \end{pmatrix} \tag{3}$$

The characters with indices exceeding the sentence boundaries are mapped to one of two special symbols, namely *"start"* and *"stop"* symbols.

The fixed-sized vector f_θ^1 is fed to two standard *Linear Layers* that successively perform affine transformations over f_θ^1, interleaved with some non-linearity function $g(\cdot)$, to extract highly non-linear features. Given a set of tags $\mathcal{T}$ for the task of interest, the network outputs a vector of size $|\mathcal{T}|$ for each character at position i, interpreted as a score for each tag in $\mathcal{T}$ and each character c_i in the sentence:

$$\begin{aligned} f_\theta(c_i) &= f_\theta^3(g(f_\theta^2(f_\theta^1(c_i)))) \\ &= W^3 g(W^2 f_\theta^1(c_i) + b^2) + b^3 \end{aligned} \tag{4}$$

where the matrices $W^2 \in \mathbb{R}^{H\times(wd)}, b^2 \in \mathbb{R}^H$, $W^3 \in \mathbb{R}^{|\mathcal{T}|\times H}$ and $b^3 \in \mathbb{R}^{|\mathcal{T}|}$ are the parameters to be trained. The hyper-parameter H is usually called the *number of hidden units*. As non-linear function, we chose a sigmoidal function[2]:

$$g(x) = 1/(1+e^{-x}) \tag{5}$$

2.3 Tag Inference

There are strong dependencies between character tags in a sentence for the tasks like word segmentation and POS tagging. The tags are organized in chunks, and it is impossible for some tags to follow other tags. We introduce a transition score A_{ij} for jumping from $i \in \mathcal{T}$ to $j \in \mathcal{T}$ tags in successive characters, and an initial scores A_{0i} for starting from the i-th tag for taking into account the sentence structure. We want the valid paths of tags to be encouraged, while discouraging all other paths.

Given an input sentence $c_{[1:n]}$, the network outputs the matrix of scores $f_\theta(c_{[1:n]})$. We use a notation $f_\theta(t|i)$ to indicate the score output by the network with parameters θ, for the sentence $c_{[1:n]}$ and for the t-th tag, at the i-th character. The score of a sentence $c_{[1:n]}$ along a path of tags $t_{[1:n]}$ is then given by the sum of transition and network scores:

$$s(c_{[1:n]}, t_{[1:n]}, \theta) = \sum_{i=1}^{n} (A_{t_{i-1}t_i} + f_\theta(t_i|i)) \tag{6}$$

Given a sentence $c_{[1:n]}$, we can find the best tag path $t^*_{[1:n]}$ by maximizing the sentence score:

$$t^*_{[1:n]} = \arg\max_{\forall t'_{[1:n]}} s(c_{[1:n]}, t'_{[1:n]}, \theta) \tag{7}$$

The Viterbi algorithm can be used for this inference. Now we are prepared to show how to train the parameters of the network in an end-to-end fashion.

2.4 Training

The training problem is to determine all the parameters of the network $\theta = (\mathcal{M}, W^2, b^2, W^3, b^3, A)$ from training data. The network generally is trained

[2]In our experiments, the sigmoidal function performs slightly better than the "hard" version of the hyperbolic tangent used by (Collobert, 2011).

by maximizing a likelihood over all the sentences in the training set $\mathcal{R}$ with respect to θ:

$$\theta \mapsto \sum_{\forall (c,t) \in \mathcal{R}} \log p(t|c,\theta) \tag{8}$$

where c represents a sentence and its associated features, and t denotes the corresponding tag sequence. We drop the subscript $[1:n]$ from now for notation simplification. The probability $p(\cdot)$ is calculated from the outputs of the neural network. We will present in the following section how to interpret neural network outputs as probabilities.

Maximizing the log-likelihood (8) with the gradient ascent algorithm[3] is achieved by iteratively selecting a example (c,t) and applying the following gradient update rule:

$$\theta \leftarrow \theta + \lambda \frac{\partial \log p(t|c,\theta)}{\partial \theta} \tag{9}$$

where λ is the learning rate (a hyper-parameter). The gradient in (9) can be computed by a classical back propagation: the differentiation chain rule is applied through the network, until the character embedding layer.

2.4.1 Sentence-Level Log-Likelihood

The score of a sentence (6) is interpreted as a conditional tag path probability by taking it to the exponential (making the score positive) and normalizing it over all possible tag paths (summing to 1 over all paths). Taking the log, the conditional probability of the true path t is given by[4]:

$$\log p(t|c,\theta) = s(c,t,\theta) - \log \sum_{\forall t'} \exp\{s(c,t',\theta)\} \tag{10}$$

[3]We did not use the stochastic gradient ascent algorithm (Bottou, 1991) to train the network as (Collobert et al., 2011). The gradient ascent algorithm was used instead for fairly comparing our algorithm with the sentence-level maximum-likelihood method (see Section 2.4.1). The gradient ascent algorithm requires a loop over all the examples to compute the gradient of the cost function, which will not cause a problem since all the training sets used in this article are finite.

[4]The cost functions are differentiable everywhere thanks to the differentiability of sigmoidal function chosen as non-linearity instead of a "hard" version of the hyperbolic tangent. For details about gradient computations, see Appendix A of (Collobert et al., 2011).

The number of terms in (10) grows exponentially with the length of the input sentence. Although one can compute it in linear time with the Viterbi algorithm, it is quite computationally expensive to compute the conditional probability of the true path, and its derivatives with respect to $f_\theta(t|i)$ and A_{ij}. The gradients with respect to the trainable parameters other than $f_\theta(t|i)$ and A_{ij} can all be computed using the derivatives with respect to $f_\theta(t|i)$ by applying the differentiation chain rule. We will see in the next section our training algorithm that has the advantage of being much cheaper to compute the gradients.

2.5 A New Training Method

The log-likelihood (10) can be seen as the difference between the forward score constrained over the valid path and the sum of the scores of all possible paths. While this training criterion is used, the neural networks are trained by maximizing the likelihood of training data. In fact, a CRF maximizes the same log-likelihood (Lafferty et al., 2001) by using a linear model in stead of a nonlinear neural network.

As an alternative to maximum-likelihood method, we propose the following training algorithm inspired by the work of (Collins, 2002). Given a training example (c,t), the network outputs the matrix of scores $f_\theta(c)$ under the current parameter settings. The highest scoring sequence of tags for the input sentence c then can be found using the Viterbi algorithm: this tagged sequence is denoted by t'. For every character c_i where $t_i \neq t'_i$, we simply set

$$\frac{\partial L_\theta(t,t'|c)}{\partial f_\theta(t_i|i)} {+}{+}, \ \frac{\partial L_\theta(t,t'|c)}{\partial f_\theta(t'_i|i)} {-}{-} \tag{11}$$

and for every transition where $t_{i-1} \neq t'_{i-1}$ or $t_i \neq t'_i$, we set

$$\frac{\partial L_\theta(t,t'|c)}{\partial A_{t_{i-1}t_i}} {+}{+}, \ \frac{\partial L_\theta(t,t'|c)}{\partial A_{t'_{i-1}t'_i}} {-}{-} \tag{12}$$

where "$+\,+$" (which increases a value by one) and "$-\,-$" (which decreases a value by one) are two unary operators, and $L_\theta(t,t'|c)$ is a new function which we now want to maximize over all the training pairs (c,t). The function $L_\theta(t,t'|c)$ can be viewed as the difference between the score of the correct path and that of the incorrect one (which is the highest scoring sequence produced by the network under the current parameters θ).

As an example, say the correct tag sequence of the sentence 狗蹲在墙角 'A dog sits in the corner' is

狗/S 蹲/S 在/S 墙/B 角/E

and under the current parameter settings the highest scoring tag sequence is

狗/S 蹲/B 在/E 墙/B 角/E

Then the derivatives with respect to $f_\theta(S|蹲)$, and $f_\theta(S|在)$ will be set to 1, that with respect to A_{SB}, A_{BE}, A_{EB}, $f_\theta(B|蹲)$, and $f_\theta(E|在)$ to -1, and that with respect to A_{SS} to 2 respectively[5]. Intuitively these assignments have the effect of updating the parameter values in a way that increases the score of the correct tag sequence and decreases the score of the incorrect one output by the network with the current parameter settings. If the tag sequence produced by the network is correct, no changes are made to the values of parameters.

Inputs:
$\mathcal{R}$: a training set.
N: a specified maximum number of iterations.
E: a desired tagging precision.

Initialization: set the initial parameters of the network with small random values.

Output: the trained parameters $\tilde{\theta}$

Algorithm:

do
  for each example $(c, t) \in \mathcal{R}$
    get the matrix $f_\theta(c)$ by the neural network under the current parameters θ
    find the highest scoring sequence of tags t' for c with $f_\theta(c)$ and A_{ij} by using the Viterbi algorithm
    if $(t \neq t')$
      compute the gradients with respect to $f_\theta(c)$ and A_{ij} as (11) and (12)
      compute the gradients with respect to the weights from output layer to character embedding layer
      update the parameters of the network by (9)
until the desired precision E achieved or maximum number of iterations N reached
return $\tilde{\theta}$

Figure 2: The training algorithm for tagging.

We propose the training algorithm in Figure 2. Note that the perceptron algorithm of (Collins, 2002) was designed for discriminatively training an HMM-style tagger, while our algorithm is used to calculate the "direction" in which the parameters are updated (i.e. the gradient of the function we want to maximize). Due to space limitations, we do not give convergence theorems justifying the training algorithm in this paper. Intuitively it can be achieved by combining the theorems of convergence for the perceptron applied to tagging problem from (Collins, 2002) with the convergence results of backpropagation algorithm from (Rumelhart et al., 1986).

3 Experiments

We conducted three sets of experiments. The goal of the first one is to test several variants for each training algorithm on the development set, to gain some understanding of how the choice of hyperparameters impacts upon the performance. We applied the network both with the sentence-level log-likelihood (SLL) and our perceptron-style training algorithm (PSA) to the two Chinese NLP problems: word segmentation, and joint CWS and POS tagging. We ran this set of experiments on the part of Chinese Treebank 4 (CTB-4)[6]. Ninety percent of the sentences (1529) were randomly chosen for training and the rest (168) were used as development set.

The second set of experiments was run on the Chinese Treebank (CTB) data sets from Bakeoff-3 (Levow, 2006), which contains a training and a test corpus for supervised word segmentation and POS tagging tasks. The results were obtained without using any extra knowledge (i.e. the closed test), and are comparable with other models in the literature.

In the third experiment, we study to see how well large unlabeled texts can be used to enhance the supervised learning. Following (Collobert et al., 2011), we first use large unlabeled data set to obtain character embeddings carrying more syntactic and semantic information, and then use these improved embeddings to initialize the character lookup tables of the networks instead of previous random values. Our corpus is the Sina news[7] that contains about 325MB data.

[5]The derivatives with respect to A_{SB} will be set to 0, because it is increased first and decreased afterwards.

[6]The data set was sections 1–43, 144–169, and 900–931 of the treebank, containing 78,023 characters, 45,135 words and 1,697 sentences. These files are double-annotated and can be regarded as golden standard files.

[7]Available at http://www.sina.com.cn/

We implemented two versions of the network: one for the sentence-level log-likelihood and one for our perceptron-style training algorithm. Both are written in Java language. All experiments were run on a computer equipped with an Intel Core i3 processor working at 2.13GHz, with 2GB RAM, running Linux and Java Development Kit 1.6. The standard F-score was used to evaluate the performance of both word segmentation and joint word segmentation and POS tagging tasks. F-score is the harmonic mean of precision p and recall r, which is defined as $2pr/(p+r)$.

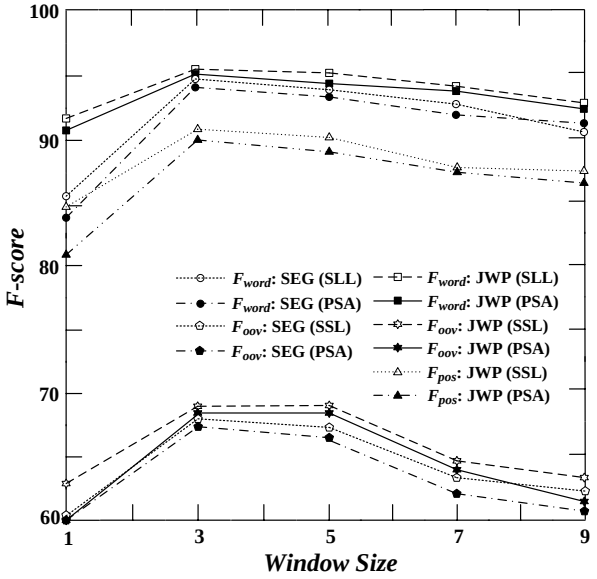


Figure 3: Average F-score versus window size.

3.1 Tagging Schemes

The network will output the scores for all the possible tags for the task of interest. For word segmentation, each character will be assigned one of four possible boundary tags: "*B*" for a character located at the beginning of a word, "*I*" for that inside of a word, "*E*" for that at the end of a word, and "*S*" for a character that is a word by itself.

Following Ng and Lou (2004) we perform joint word segmentation and POS tagging task in a labeling fashion by expanding boundary labels to include POS tags. For instance, we describe verb phases using four different tags. Tag "*S_VP*" is used to mark a verb phase containing a single character. Other tags "*B_VP*", "*I_VP*", and "*E_VP*" are used to mark the first, in-between and last characters of the verb phrase. In fact, we used the "*IOBES*" tagging scheme, and tag "*O*" is not applicable to Chinese word segmentation and POS tagging tasks.

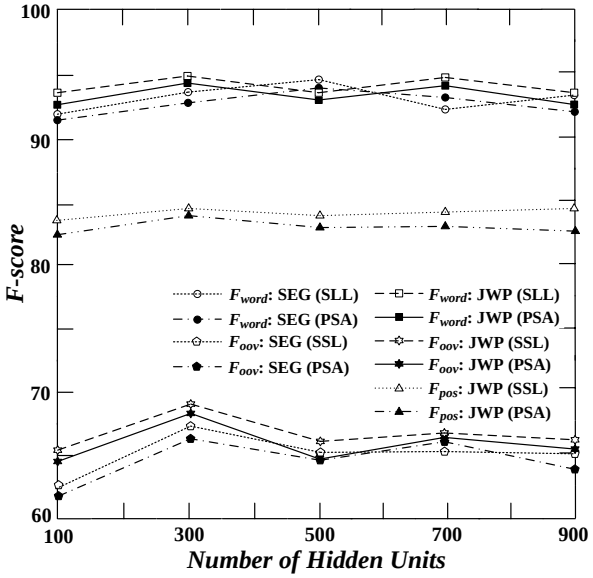


Figure 4: Average F-score versus number of hidden units.

3.2 The Choice of Hyper-parameters

We tuned the hyper-parameters by trying only a few different networks. We report in Figure 3 the F-scores on the development set versus window size for word segmentation (SEG) and joint word segmentation and POS tagging (JWP) tasks with the sentence-level log-likelihood (SLL) and our perceptron-style training algorithm (PSA), and report in Figure 4 the F-scores on the same data set versus number of hidden units. The average F-scores were obtained over 5 runs with different random initialization for each setting of the network. The F-scores of the word segmentation, out-of-vocabulary, and POS tagging are denoted by F_{word}, F_{oov} and F_{pos} respectively.

Generally, the number of hidden units has a limited impact on the performance if it is large enough, which is consistent with the findings of (Collobert et al., 2011) for English. It can be seen from Figure 3 that the performance drops smoothly when the window size is larger than 3. In particularly, the F-score of out-of-vocabulary identification decreases relatively fast beyond window size 5, which shows

that the size of window (and the number of parameters) is too large that the trained network has overfitted on training data. An explanation for this result is that most Chinese words are less than 3 characters, and the neighboring characters outside of the window (size 5) become "noise" when we perform word segmentation.

The hyper-parameters of the network used in all the following experiments are shown in Table 1. Although the top performance was obtained by the network with window size 3, we chose the architecture with window size 5 because a larger training corpus will be used in the following experiments, and the sparseness problem would be alleviated. Furthermore, in order to obtain character embeddings by using large unlabeled data, we prefer to "observe" a character within a slightly larger window to better discover its syntactic and semantic information.

Hyper-parameter	Value
Window size	5
Number of hidden units	300
Character feature dimension	50
Learning rate	0.02

Table 1: Hyper-parameters of the network.

We report in Table 2 the F-score of the first five iterations on the development set for word segmentation with SLL and PSA. The data in the fourth and fifth rows of the table shows the convergence of PSA. The difference of the F-scores between the networks with SLL and PSA can be negligible after the number of iteration is greater than 5. In our implementation, for each iteration the training time is reduced at least 10% by using PSA, compared with SLL. The training time can be reduced further for more complex tasks like POS tagging and semantic role labeling in which a larger tag set is used.

Iteration		1	2	3	4	5
SSL	F_{word}	49.89	69.56	88.91	90.19	91.24
	F_{oov}	15.92	27.54	54.37	55.89	59.74
	Time (s)	209	398	586	737	886
PSA	F_{word}	49.04	68.54	87.79	89.07	91.19
	F_{oov}	13.61	25.79	52.30	55.15	60.49
	Time (s)	184	343	497	610	754

Table 2: Word segmentation results with SLL and PSA for the first five iterations.

Many exponential sums ($\sum_i \exp(x_i)$) are required for training the networks with SLL, and in most cases the values of exponential sums will exceed the range of double-precision floating-point arithmetic defined in popular programming languages. These sums need to be estimated by analytic number theory. In comparison, a lot of the computation-intensive exponential sums are avoided in our training algorithm, which not only speed up the training of the networks but also make it easier to be implemented.

3.3 Closed Test on the SIGHAN Bakeoff

We trained the networks with PSA on the Chinese Treebank (CTB) data set from Bakeoff-3 for both SEG and JWP tasks. The results are reported in Table 3. The hyper-parameters of our networks are reported in Table 1. Although results show that our networks with PSA are behind the state-of-the-art systems, the networks perform comparatively well, considering we did not use any extra information. Many other systems used some extra heuristics or resources to improve their performance. For example, a key parameter in the system of (Wang et al., 2006) was optimized in advance by using an external segmented corpus, and a manually prepared list of characters as well as their types was used in (Zhao et al., 2006; Zhu et al., 2006; Kruengkrai et al., 2009).

It is worth noting that the comparison for joint word segmentation and POS tagging task is indirect because the different versions of CTB were used. We reported the results on CTB-3 from SIGHAN Bakeoff-3, while both (Jiang et al., 2008) and (Kruengkrai et al., 2009) used CTB-5. Both (Ng and Lou, 2004) and (Zhang and Clark, 2008) evenly partitioned the sentences in CTB3 into ten groups, and used nine groups for training and the rest for testing.

Following (Bengio et al., 2003; Collobert et al., 2011), we want semantically and syntactically similar characters to be close in the embedding space. If we knew that 狗 'dog' and 猫 'cat' were similar semantically, and similarly for 蹲 'sit' and 躺 'lie', we could generalize from 狗蹲在墙角 'A dog sits in the corner' to 猫蹲在墙角 'A cat sits in the corner', and to 猫躺在墙角 'A cat lies in the corner' in the same way. We describe the way to obtain these character embeddings by using large unlabeled data in the next section.

Approach		F_{word}	R_{oov}	F_{pos}
SEG	(Zhao et al., 2006)	93.30	70.70	–
	(Wang et al., 2006)	93.00	68.30	–
	(Zhu et al., 2006)	92.70	63.40	–
	(Zhang et al., 2006)	92.60	61.70	–
	(Feng et al., 2006)	91.70	68.00	–
	PSA	92.59	64.24	–
	PSA + LM	94.57	70.12	–
JWP	(Ng and Lou, 2004)	95.20	–	–
	(Zhang and Clark, 2008)	95.90	–	91.34
	(Jiang et al., 2008)	97.30	–	92.50
	(Kruengkrai et al., 2009)	96.11	–	90.85
	PSA	93.83	68.21	90.79
	PSA + LM	95.23	72.38	91.82

Table 3: Comparison of the F-scores on the Penn Chinese Treebank

3.4 Combined Approach

We used the corpus of Sina news to obtain character embeddings carrying more semantic and syntactic information by training a language model that evaluates the acceptability of a piece of text. This language model is again the neural network, and we also use PSA to train the language model. Following (Collobert et al., 2011), we minimize the following criterion with respect to the parameters θ:

$$\theta \mapsto \sum_{\forall h \in \mathcal{H}} \sum_{\forall c' \in \mathcal{D}} \max\{0, 1 - f_\theta(c|h) + f_\theta(c'|h)\} \tag{13}$$

where the score $f_\theta(c|h)$ is the output of the network with parameters θ for a character c at the center of a window h, $\mathcal{D}$ is the dictionary of characters, $\mathcal{H}$ is the set of all possible text windows (i.e. character sequences) from the training data, and $c'|h$ denotes the window obtained by replacing the central character of the window h by the character c'.

We used a dictionary consisting of the characters extracted from all the data sets in Bakeoff-3, which contains about eight thousand characters. The total unsupervised training time was about two weeks. Our combined approach works as initializing the lookup tables of the supervised networks with the character embeddings obtained by unsupervised learning, and then performing supervised training on CTB-3. The lookup tables will not be modified at the supervised training stage.

We reported the results in Table 3, in which our combined approach is indicated by "PSA + LM". It can be seen from Table 3 that this approach results in a performance boost for both SEG and JWP tasks. The POS tagging F-score of our approach was comparable to but still less than the model of (Jiang et al., 2008). They achieved the best score by first separately training multiple word-, character-, and POS n-gram based models, and then integrating them by cascading method. In comparison, our networks achieve the performance by automatically discovering useful features by itself and avoiding the task-specific engineering.

Table 4 compares the decoding speeds on the test data from CTB-3 for our system and for two CRFs-based word segmentation systems. Regardless of the differences in implementation, the neural networks clearly run considerably faster than the systems based on the CRFs model. They also require much more memory than our neural networks.

System	Number of parameters	Time (s)
(Tsai et al., 2006)	3.1×10^6	1669
(Zhao et al., 2006)	3.8×10^6	2382
Neural network	4.7×10^5	138

Table 4: Comparison of computational cost.

4 Related Work

Word segmentation has been pursued with considerable efforts in the Chinese NLP community, and statistical approaches are clearly dominant in the last decade. A popular statistical approach is the character-based tagging solution that treats word segmentation as a sequence tagging problem, assigning labels to the characters indicating whether a character locates at the beginning of, inside, or at the end of a word. The character-based tagging solution was first proposed in (Xue, 2003). This work caused quite a number of character position tagging based CWS studies because known and unknown words can be treated in the same way. Peng, Feng and McCallum (2004) first introduced a linear-chain CRFs model to the character tagging based word segmentation. Zhang and Clark (2007) proposed a word-based CWS approach using a discriminative perceptron learning algorithm, which allows word-level information to be added as features.

Recent years have seen a rise of joint word segmentation and POS tagging approach that improves

the accuracies of both tasks and does not suffer from the error propagation. Ng and Lou (2004) perform such joint task in a labeling fashion by expanding boundary labels to include POS tags. Zhang and Clark (2008) proposed a linear model for the same joint task, which overcomed the disadvantage of (Ng and Lou, 2004), in which it was unable to incorporate "whole word + POS tag" features. Sun (2011) described a sub-word model using stacked learning technique for the joint task, which explored the complementary strength of different character- and word-based segmenters with different views.

The majority of the state-of-the-art systems address their tasks by applying linear statistical models to ad-hoc features. The researchers first chose task-specific features which are then fed to a classification algorithm. The selected features may vary greatly because they are usually chosen in a empirical process, mainly based first on linguistic intuition, and then trial and error. It seems reasonable to assume that the number and effectiveness of features constitutes a major factor in the performance of the various systems, and might even more important than the particular statistical models they used. In comparison, we try to avoid task-specific feature engineering, and use the neural network to learn several layers of feature extraction from the inputs. To the best of our knowledge, this study is among the first ones to perform Chinese word segmentation and POS tagging by deep learning.

It was reported that supervised and unsupervised approaches can be integrated to improve on the overall performance of word segmentation by combining the strengths of both. Zhao and Kit (2011) explored the feasibility of enhancing supervised segmentation by informing the supervised learner of goodness scores obtained from large unlabeled corpus. Sun and Xu (2011) investigated how to improve on the accuracy of supervised word segmentation by leveraging the statistics-based features derived from large unlabeled in-domain corpus and the document to be segmented. The basic idea of these integration solutions is to incorporate a set of statistics-based measures into a CRFs model after these measures are derived from unlabeled data and discretized into feature values. In comparison, we use large unlabeled data to obtain the character embeddings with more syntactic and semantic information.

Several works have investigated how to use deep learning for NLP applications (Bengio et al., 2003; Collobert et al., 2011; Collobert, 2011; Socher et al., 2011). In most cases, words are fed to the neural networks as inputs, and the lookup tables map each word to a vector representation. Our network is different in that the inputs to the network are characters, more raw units than words. In many Asian languages, such as Chinese and Japanese, they are written without using whitespace to delimit words. For these languages, the character becomes a more natural form of input. Furthermore, a perceptron-style algorithm for tagging is proposed for training the networks.

5 Conclusion

We have described a perceptron-style algorithm for training the neural networks, which is much easier to be implemented, and has speed advantage over the maximum-likelihood scheme, while the loss in performance is negligible. The neural networks trained with PSA have been applied to Chinese word segmentation and POS tagging tasks, and the networks achieved close to state-of-the-art performance by using the character representations learned from large unlabeled corpus.

Although we focus on the question of how far we can go for Chinese word segmentation and POS tagging without using the extra task-specific features in this study, there are at least three ways to further improve the performance of the networks, which are worthy to be explored in the future: (1) introduce specific linguistic features (e.g. gazetteer features) that are helpful for the tasks; (2) incorporate some common techniques, such as cascading, voting, and ensemble; and (3) use the special network architecture tailored for the tasks of interest.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. The work was supported by a grant from the National Natural Science Foundation of China (No. 60903078), a grant from Shanghai Leading Academic Discipline Project (No. B114), a grant from Shanghai Municipal Natural Science Foundation (No. 13ZR1403800), and a grant from FDUROP.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3: 1137–1155.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. *In Proceedings of the Neuro-Nîmes*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. *In Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS'11)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12: 2493–2537.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for Chinese word extraction. *Computational Linguistics*, 30(1): 75–93.

Yuanyong Feng, Sun Le, and Yuanhua Lv. 2006. Chinese word segmentation and name entity recognition based on conditional random fields models. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Wenbin Jiang, Liang Huang, Qun Liu, Yajuan Lu. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*.

Zhihui Jin, and Kumiko Tanaka-Ishii. 2006. Unsupervised segmentation of Chinese text by use of branching entropy. *In Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06)*.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and pos tagging. *In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL'09)*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *In Proceedings of the International Conference on Machine learning (ICML'01)*.

Gina A. Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. *In Proceedings of the SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Hwee T. Ng and Jin K. Lou. 2004. Chinese part-of-speech tagging: one-at-atime or all-at-once? word-based or character-based? *In Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *In Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning internal representations by backpropagating errors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1: 318–362. MIT Press.

Richard Socher, Cliff C-Y. Lin, Andrew Y. Ng, and Christopher D. Manning 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. *In Proceedings of the International Conference on Machine learning (ICML'11)*.

Weiwei Sun. 2011. A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. *In Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*.

Richard T.-H. Tsai, Hsieh C. Hung, Chenglung Sung, Hongjie Dai, and Wenlian Hsu. 2006. On closed task of Chinese word segmentation: An improved CRF model coupled with character clustering and automatically generated template matching. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. 2006. Chinese word segmentation with maximum entropy and n-gram language model. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1): 29–48.

Min Zhang, GuoDong Zhou, LingPeng Yang, and DongHong Ji. 2006. Chinese word segmentation and named entity recognition based on a context-dependent mutual information independence Model. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-base perceptron algorithm. *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*.

Hai Zhao, Chang N. Huang, and Mu Li. 2006. An improved Chinese word segmentation system with conditional random field. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Hai Zhao and Chunyu Kit. 2011. Integrating unsupervised and supervised word segmentation: The role of goodness measures. *Information Sciences*, 181(1): 163–183.

Muhua Zhu, Yilin Wang, Zhenxing Wang, Huizhen Wang, and Jingbo Zhu. 2006. Designing special post-processing rules for SVM-based Chinese word segmentation. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Joint Chinese Word Segmentation and POS Tagging on Heterogeneous Annotated Corpora with Multiple Task Learning

Xipeng Qiu, Jiayi Zhao, Xuanjing Huang
Fudan University, 825 Zhangheng Road, Shanghai, China
xpqiu@fudan.edu.cn, zjy.fudan@gmail.com, xjhuang@fudan.edu.cn

Abstract

Chinese word segmentation and part-of-speech tagging (S&T) are fundamental steps for more advanced Chinese language processing tasks. Recently, it has attracted more and more research interests to exploit heterogeneous annotation corpora for Chinese S&T. In this paper, we propose a unified model for Chinese S&T with heterogeneous annotation corpora. We first automatically construct a loose and uncertain mapping between two representative heterogeneous corpora, Penn Chinese Treebank (CTB) and PKU's People's Daily (PPD). Then we regard the Chinese S&T with heterogeneous corpora as two "related" tasks and train our model on two heterogeneous corpora simultaneously. Experiments show that our method can boost the performances of both of the heterogeneous corpora by using the shared information, and achieves significant improvements over the state-of-the-art methods.

1 Introduction

Currently, most of statistical natural language processing (NLP) systems rely heavily on manually annotated resources to train their statistical models. The more of the data scale, the better the performance will be. However, the costs are extremely expensive to build the large scale resources for some NLP tasks. Even worse, the existing resources are often incompatible even for a same task and the annotation guidelines are usually different for different projects, since there are many underlying linguistic theories which explain the same language with different perspectives. As a result, there often exist multiple heterogeneous annotated corpora for a same task with vastly different and incompatible annotation philosophies. These heterogeneous resources are waste on some level if we cannot fully exploit them.

However, though most of statistical NLP methods are not bound to specific annotation standards, almost all of them cannot deal simultaneously with the training data with different and incompatible annotation. The co-existence of heterogeneous annotation data therefore presents a new challenge to utilize these resources.

The problem of incompatible annotation standards is very serious for many tasks in NLP, especially for Chinese word segmentation and part-of-speech (POS) tagging (Chinese S&T). In Chinese S&T, the annotation standards are often incompatible for two main reasons. One is that there is no widely accepted segmentation standard due to the lack of a clear definition of Chinese words. Another is that there are no morphology for Chinese word so that there are many ambiguities to tag the parts-of-speech for Chinese word. For example, the two commonly-used corpora, PKU's People's Daily (PPD) (Yu et al., 2001) and Penn Chinese Treebank (CTB) (Xia, 2000), use very different segmentation and POS tagging standards.

For example, in Table 1, it is very different to annotate the sentence "刘翔进入中国区总决赛 (Liu Xiang reaches the national final in China)" with guidelines of CTB and PDD. PDD breaks some phrases, which are single words in

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 658–668,
Seattle, Washington, USA, 18-21 October 2013.

	Liu	Xiang	reachs	China		final	
CTB	刘翔/NR		进入/VV	中国区/NN		总决赛/NN	
PDD	刘/nrf	翔/nrg	进入/v	中国/ns	区/n	总/b	决赛/vn

Table 1: Incompatible word segmentation and POS tagging standards between CTB and PDD

CTB, into two words. The POS tagsets are also significantly different. For example, PDD gives diverse tags "n" and "vn" for the noun, while CTB just gives "NN". For proper names, they may be tagged as "nr", "ns", etc in PDD, while they are just tagged as "NR" in CTB.

Recently, it has attracted more and more research interests to exploit heterogeneous annotation data for Chinese word segmentation and POS tagging. (Jiang et al., 2009) presented a preliminary study for the annotation adaptation topic. (Sun and Wan, 2012) proposed a structure-based stacking model to fully utilize heterogeneous word structures. They also reported that there is no one-to-one mapping between the heterogeneous word classification and the mapping between heterogeneous tags is very uncertain.

These methods usually have a two-step process. The first step is to train the preliminary taggers on heterogeneous annotations. The second step is to train the final taggers by using the outputs of the preliminary taggers as features. We call these methods as "**pipeline-based**" methods.

In this paper, we propose a method for joint Chinese word segmentation and POS tagging with heterogeneous annotation corpora. We regard the Chinese S&T with heterogeneous corpora as two "related" tasks which can improve the performance of each other. Since it is impossible to establish an exact mapping between two annotations, we first automatically construct a loose and uncertain mapping the heterogeneous tagsets of CTB and PPD. Thus we can tag a sentence in one style with the help of the "related" information in another heterogeneous style. The proposed method can improve the performances of joint Chinese S&T on both corpora by using the shared information of each other, which is proven effective by experiments.

There are three main contributions of our model:

- First, we regard these two joint S&T tasks on different corpora as two related tasks which have interdependent and peer relationship.
- Second, different to the pipeline-based methods, our model can be trained simultaneously on the heterogeneous corpora. Thus, it can also produce two different styles of POS tags.
- Third, our model do not depend on the exactly correct mappings between the two heterogeneous tagsets. The correct mapping relations can be automatically built in training phase.

The rest of the paper is organized as follows: We first introduce the related works in section 2 and describe the background of character-based method for joint Chinese S&T in section 3. Section 4 presents an automatic method to build the loose mapping function. Then we propose our method on heterogeneous corpora in 5 and 6. The experimental results are given in section 7. Finally, we conclude our work in section 8.

2 Related Works

There are some works to exploit heterogeneous annotation data for Chinese S&T.

(Gao et al., 2004) described a transformation-based converter to transfer a certain annotation-style word segmentation result to another style. However, this converter need human designed transformation templates, and is hard to be generalized to POS tagging.

(Jiang et al., 2009) proposed an automatic adaptation method of heterogeneous annotation standards, which depicts a general pipeline to integrate the knowledge of corpora with different

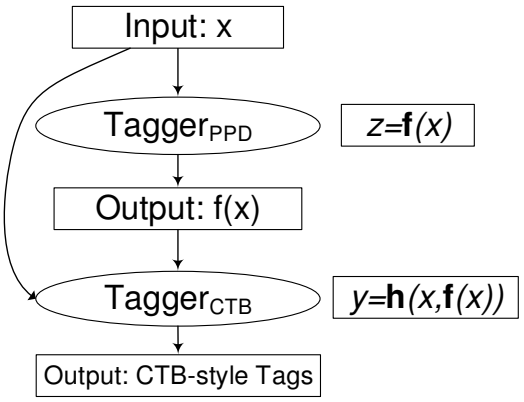


Figure 1: Traditional Pipeline-based Strategy for Heterogeneous POS Tagging

underling annotation guidelines. They further proposed two optimization strategies, iterative training and predict-self re-estimation, to further improve the accuracy of annotation guideline transformation (Jiang et al., 2012).

(Sun and Wan, 2012) proposed a structure-based stacking model to fully utilize heterogeneous word structures.

These methods regard one annotation as the main target and another annotation as the complementary/auxiliary purposes. For example, in their solution, an auxiliary tagger $\mathbf{Tagger_{PPD}}$ is trained on a complementary corpus PPD, to assist the target CTB-style $\mathbf{Tagger_{CTB}}$. To refine the character-based tagger, PPD-style character labels are directly incorporated as new features. The brief sketch of these methods is shown in Figure 1.

The related work in machine learning literature is multiple task learning (Ben-David and Schuller, 2003), which learns a problem together with other related problems at the same time, using a shared representation. This often leads to a better model for the main task, because it allows the learner to use the commonality among the tasks. Multiple task learning has been proven quite successful in practice and has been also applied to NLP (Ando and Zhang, 2005). We also preliminarily verified that multiple task learning can improve the performance on this problem in our previous work (Zhao et al., 2013), which is a simplified case of the work in this paper and has a relative low complexity.

Different with the multiple task learning, whose tasks are actually different labels in the same classification task, our model utilizes the shared information between the real different tasks and can produce the corresponding different styles of outputs.

3 Joint Chinese Word Segmentation and POS Tagging

Currently, the mainstream method of Chinese POS tagging is joint segmentation & tagging with character-based sequence labeling models(Lafferty et al., 2001), which can avoid the problem of segmentation error propagation and achieve higher performance on both subtasks(Ng and Low, 2004; Jiang et al., 2008; Sun, 2011; Qiu et al., 2012).

The label of each character is the cross-product of a segmentation label and a tagging label. If we employ the commonly used label set {B, I, E, S} for the segmentation part of cross-labels ({B, I, E} represent *Begin*, *Inside*, *End* of a multi-node segmentation respectively, and S represents a *Single* node segmentation), the label of character can be in the form of {B-T}(T represents POS tag). For example, *B-NN* indicates that the character is the begin of a noun.

4 Automatically Establishing the Loose Mapping Function for the Labels of Characters

To combine two human-annotated corpora, the relationship of their guidelines should be found. A **mapping function** should be established to represent the relationship between two different annotation guidelines. However, the exact mapping relations are hard to establish. As reported in (Sun and Wan, 2012), there is no one-to-one mapping between their heterogeneous word classification, and the mapping between heterogeneous tags is very uncertain.

Fortunately, there is a loose mapping can be found in CTB annotation guideline[1] (Xia, 2000). Table 2 shows some

[1]Available at http://www.cis.upenn.edu/ ~chi-

	CTB's Tag	PDD' Tag[1]
Total tags	33	26
verbal noun	NN	v[+nom]
proper noun	NR	n
是 (shi4)	VC	v
有 (you3)	VE, VV	v
conjunctions	CC, CS	c
other verb	VV, VA	v, a, z
number	CD, OD	m

[1] The tag set of PDD just includes the 26 broad categories in the mapping table. The whole tag set of PDD has 103 sub categories.

Table 2: Examples of mapping between CTB and PDD's tagset

mapping relations in CTB annotation guideline. These loose mapping relations are many-to-many mapping. For example, the mapping may be "NN/CTB↔{n,nt,nz}/PDD", "NR/CTB↔{nr,ns}/PDD", "v/PDD↔{VV, VA}/CTB" and so on.

We define $\mathcal{T}_1$ and $\mathcal{T}_2$ as the tag sets for two different annotations, and $t_1 \in \mathcal{T}_1$ and $t_2 \in \mathcal{T}_2$ are the corresponding tags in two tag sets respectively.

We first establish a loose mapping function $\mathbf{m} : \mathcal{T}_1 \times \mathcal{T}_2 \rightarrow \{0, 1\}$ between the tags of CTB and PDD.

$$\mathbf{m}(t_1, t_2) = \begin{cases} 1 & \text{if } t_1 \text{ and } t_2 \text{ have mapping relation} \\ 0 & \text{else} \end{cases} \quad (1)$$

The mapping relations are automatically build from the CTB guideline (Xia, 2000). Due to the fact that the tag set of PPD used in the CTB guideline is just broad categories, we expand the mapping relations to include the sub categories. If a PPD's tag is involved in the mapping, all its sub categories should be involved. For example, for the mapping "NR/CTB↔nr/PDD", the relation of *NR* and *nrf/nrg* should be added in the mapping relations too (*nrf/nrg* belong to *nr*).

Since we use the character-based joint S&T model, we also need to find the mapping function between the labels of characters.

nese/posguide.3rd.ch.pdf

In this paper, we employ the commonly used label set {B, I, E, S} for the segmentation part of cross-labels and the label of character can be in the form of {B-T}(T represents POS tag). Thus, each mapping relation $t_1 \leftrightarrow t_2$ can be automatically transformed to four forms: B-t_1 ↔B-t_2, I-t_1 ↔I-t_2, E-t_1 ↔E-t_2 and S-t_1 ↔S-t_2. ("B-NR/CTB↔{B-nr,B-ns}/PPD" for example).

Beside the above transformation, we also give a slight modification to adapt the different segmentation guidelines. For instance, the person name "莫言 (Mo Yan)" is tagged as "B-NR, E-NR" in CTB but "S-nrf, S-nrg" in PPD. So, some special mappings may need to be added like "B-NR/CTB↔S-nrf/PPD", "E-NR/CTB↔{S-nrg, E-nrg}/PPD", "M-NR/CTB↔{B-nrg, M-nrg}/PPD" and so on. Although these special mappings are also established automatically with an exhaustive solution. In fact, we give segmentation alignment only to proper names due to the limitation of computing ability.

Thus, we can easily build the **loose bidirectional mapping function $\tilde{\mathbf{m}}$** for the labels of characters. An illustration of our construction flowchart is shown in Figure 2.

Finally, total 524 mappings relationships are established.

5 Joint Chinese S&T with Heterogeneous Data with Multiple Task Learning

Inspired by the multiple task learning (Ben-David and Schuller, 2003), we can regard the joint Chinese S&T with heterogeneous data as two "related" tasks, which can improve the performance of each other simultaneously with shared information.

5.1 Sequence Labeling Model

We first introduce the commonly used sequence labeling model in character-based joint Chinese S&T.

Sequence labeling is the task of assigning labels $\mathbf{y} = y_1, \ldots, y_n (y_i \in \mathcal{Y})$ to an input sequence $\mathbf{x} = x_1, \ldots, x_n$. $\mathcal{Y}$ is the set of labels.

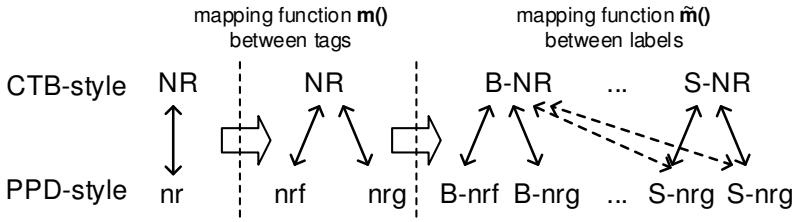


Figure 2: An Illustration of Automatically Establishing the Loose Mapping Function

Given a sample $\mathbf{x}$, we define the feature $\Phi(\mathbf{x}, \mathbf{y})$. Thus, we can label $\mathbf{x}$ with a score function,

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} S(\mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})), \tag{2}$$

where $\mathbf{w}$ is the parameter of score function $S(\cdot)$. The feature vector $\Phi(\mathbf{x}, \mathbf{y})$ consists of lots of overlapping features, which is the chief benefit of discriminative model. Different algorithms vary in the definition of $S(\cdot)$ and the corresponding objective function. $S(\cdot)$ is usually defined as linear or exponential family function.

For first-order sequence labeling, the feature can be denoted as $\phi_k(\mathbf{x}, y_{i-1:i})$, where i stands for the position in the sequence and k stands for the number of feature templates. For the linear classifier, the score function can be rewritten in detail as

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_{i=1}^{L} \left(\langle \mathbf{u}, \mathbf{f}(\mathbf{x}, y_i) \rangle + \langle \mathbf{v}, \mathbf{g}(\mathbf{x}, y_{i-1:i}) \rangle \right), \tag{3}$$

where $y_{i:j}$ denotes label subsequence $y_i y_{i+1} \cdots y_j$; $\mathbf{f}$ and $\mathbf{g}$ denote the state and transition feature vectors respectively, $\mathbf{u}$ and $\mathbf{v}$ are their corresponding weight vectors; L is the length of $\mathbf{x}$.

5.2 The Proposed Model

Different to the single task learning, the heterogeneous data have two sets of labels $\mathcal{Y}$ and $\mathcal{Z}$.

The heterogeneous datasets $\mathbb{D}_s$ and $\mathbb{D}_s$ consist of $\{\mathbf{x}_i, \mathbf{y}_i\}(i = 0, \cdots, m)$ and $\{\mathbf{x}_i, \mathbf{z}_i\}(i = 0, \cdots, n)$ respectively.

For a sequence $\mathbf{x} = x_1, \ldots, x_L$ with length L. , there may have two output sequence labels $\mathbf{y} = y_1, \ldots, y_L$ and $\mathbf{z} = z_1, \ldots, z_L$, where $y_i \in \mathcal{Y}$ and $z_i \in \mathcal{Z}$.

We rewrite the loose mapping function $\tilde{\mathbf{m}}$ between two label sets into the following forms,

$$\varphi(y) = \{z | \tilde{\mathbf{m}}(y, z) = 1\}, \tag{4}$$
$$\varphi(z) = \{y | \tilde{\mathbf{m}}(y, z) = 1\}, \tag{5}$$

where $\varphi(z) \subset \mathcal{Y}$ and $\varphi(y) \subset \mathcal{Z}$ are the subsets of $\mathcal{Y}$ and $\mathcal{Z}$. Give a label y(or z) in an annotation, the loose mapping function φ returns the corresponding mapping label set in another heterogeneous annotation.

Our model for heterogeneous sequence labeling can be write as

$$\begin{aligned}\hat{\mathbf{y}} = \arg\max_{\mathbf{y}, y_i \in \mathcal{Y}} \sum_{i=1}^{L} \Big(& \langle \mathbf{u}, \mathbf{f}(\mathbf{x}, y_i) \rangle \\ & + \langle \mathbf{s}, \sum_{z \in \varphi(y_i)} \mathbf{h}(\mathbf{x}, z) \rangle \\ & + \langle \mathbf{v}_1, \mathbf{g}_1(\mathbf{x}, y_{i-1:i}) \rangle \\ & + \langle \mathbf{v}_2, \sum_{\substack{z_{i-1} \in \varphi(y_{i-1}) \\ z_i \in \varphi(y_i)}} \mathbf{g}_2(\mathbf{x}, z_{i-1:i}) \rangle \Big),\end{aligned} \tag{6}$$

and

$$\begin{aligned}\hat{\mathbf{z}} = \arg\max_{\mathbf{z}, z_i \in \mathcal{Z}} \sum_{i=1}^{L} \Big(& \langle \mathbf{u}, \sum_{y \in \varphi(z_i)} \mathbf{f}(\mathbf{x}, y) \rangle + \\ & \langle \mathbf{s}, \mathbf{h}(\mathbf{x}, z_i) \rangle \\ & + \langle \mathbf{v}_1, \sum_{\substack{y_{i-1} \in \varphi(z_{i-1}) \\ y_i \in \varphi(z_i)}} \mathbf{g}_1(\mathbf{x}, y_{i-1:i}) \rangle \\ & + \langle \mathbf{v}_2, \mathbf{g}_2(\mathbf{x}, z_{i-1:i}) \rangle \Big),\end{aligned} \tag{7}$$

where $\mathbf{f}$ and $\mathbf{h}$ represent the state feature vectors on two label sets $\mathcal{Y}$ and $\mathcal{Z}$ respectively.

In Eq.(6) and (7), the score of the label of every character is decided by the weights of the corresponding mapping labels and itself.

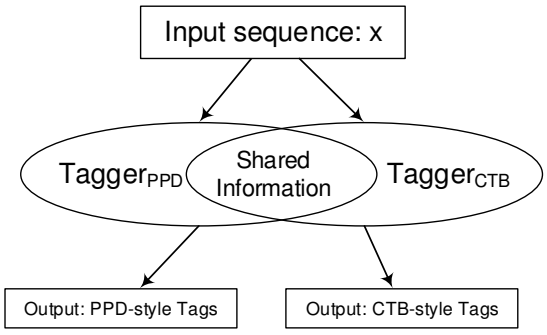


Figure 3: Our model for Heterogeneous POS Tagging

The main challenge of our model is the efficiency of decoding algorithm, which is similar to structured learning with latent variables(Liang et al., 2006) (Yu and Joachims, 2009). Most methods for structured learning with latent variables have not expand all possible mappings. In this paper, we also only expand the mapping that with highest according to the current model.

Our model is shown in Figure 3 and the flowchart is shown in Algorithm 1. If given the output type of label T, we only consider the labels in T to initialize the Viterbi matrix, and the score of each node is determined by all the involved heterogeneous labels according to the loose mapping function.

input : character sequence $x_{1:L}$
loose mapping function φ
output type: $T(T \in \{T_y, T_z\})$
output: label sequence **ls**
if $T == T_y$ **then**
  calculate **ls** using Eq. (6);
else if $T == T_z$ **then**
  calculate **ls** using Eq. (7) ;
else
  return null;
end
return ls

Algorithm 1: Flowchart of the Tagging process of the proposed model

6 Training

We use online Passive-Aggressive (PA) algorithm (Crammer and Singer, 2003; Crammer et al., 2006) to train the model parameters. Following (Collins, 2002), the average strategy is used to avoid the overfitting problem.

For the sake of simplicity, we merge the Eq.(6) and (7) into a unified formula.

Given a sequence $\mathbf{x}$ and the expect type of tags T, the merged model is

$$\hat{\mathbf{y}} = \arg\max_{\substack{\mathbf{y} \\ t(\mathbf{y})=T}} \langle \mathbf{w}, \sum_{\mathbf{z} \in \psi(\mathbf{y})} \Phi(\mathbf{x}, \mathbf{z}) \rangle, \quad (8)$$

where $t(\mathbf{y})$ is a function to judge the type of output tags; $\psi(\mathbf{y})$ represents the set $\{\varphi(y_1) \otimes \varphi(y_2) \otimes \cdots \otimes \varphi(y_L)\} \cup \{\mathbf{y}\}$, where $\otimes$ means Cartesian product; $\mathbf{w} = (\mathbf{u}^T, \mathbf{s}^T, \mathbf{v}_1^T, \mathbf{v}_2^T)^T$ and $\Phi = (\mathbf{f}^T, \mathbf{h}^T, \mathbf{g}_1^T, \mathbf{g}_2^T)^T$.

We redefine the score function as

$$S(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \sum_{\mathbf{z} \in \psi(\mathbf{y})} \Phi(\mathbf{x}, \mathbf{z}) \rangle. \quad (9)$$

Thus, we rewrite the model into a unified formula

$$\hat{\mathbf{y}} = \arg\max_{\substack{\mathbf{y} \\ t(\mathbf{y})=T}} S(\mathbf{w}, \mathbf{x}, \mathbf{y}). \quad (10)$$

Given an example $(\mathbf{x}, \mathbf{y})$, $\hat{\mathbf{y}}$ is denoted as the incorrect label sequence with the highest score

$$\hat{\mathbf{y}} = \arg\max_{\substack{\bar{\mathbf{y}} \neq \mathbf{y} \\ t(\bar{\mathbf{y}})=t(\mathbf{y})}} S(\mathbf{w}, \mathbf{x}, \bar{\mathbf{y}}). \quad (11)$$

The **margin** $\gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y}))$ is defined as

$$\gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y})) = S(\mathbf{w}, \mathbf{x}, \mathbf{y}) - S(\mathbf{w}, \mathbf{x}, \hat{\mathbf{y}}). \quad (12)$$

Thus, we calculate the **hinge loss** $\ell(\mathbf{w}; (\mathbf{x}, \mathbf{y}))$, (abbreviated as ℓ_w) by

$$\ell_w = \begin{cases} 0, & \gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y})) > 1 \\ 1 - \gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y})), & \text{otherwise} \end{cases} \quad (13)$$

In round k, the new weight vector $\mathbf{w}_{k+1}$ is calculated by

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w} - \mathbf{w}_k||^2 + \mathcal{C} \cdot \xi,$$
$$\textbf{s.t. } \ell(\mathbf{w}; (\mathbf{x}_k, \mathbf{y}_k)) <= \xi \textbf{ and } \xi >= 0 \quad (14)$$

where ξ is a non-negative slack variable, and $\mathcal{C}$ is a positive parameter which controls the influence of the slack term on the objective function.

Following the derivation in PA (Crammer et al., 2006), we can get the update rule,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \tau_k \mathbf{e}_k, \tag{15}$$

where

$$\begin{aligned} \mathbf{e}_k &= \sum_{\mathbf{z} \in \psi(\mathbf{y}_k)} \Phi(\mathbf{x}_k, \mathbf{z}) - \sum_{\mathbf{z} \in \psi(\hat{\mathbf{y}_k})} \Phi(\mathbf{x}_k, \mathbf{z}), \\ \tau_k &= \min(\mathcal{C}, \frac{\ell_{w_k}}{\|\mathbf{e}_k\|^2}). \end{aligned}$$

As we can see from the Eq. (15), when we update the weight vector, the update information includes not only the features extracted from current input, but also that extracted from the loose mapping sequence of input. For each feature, the weights of its corresponding related features derived from the loose mapping function will be updated with the same magnitude as well as itself.

Our method regards two annotations to be interdependence and peer relationship. Therefore, the two heterogeneous annotated corpora can be simultaneously used as the input of our training algorithm. Because of the tagging and training algorithm, the weights and tags of two corpora can be used separately with the only dependent part built by the loose mapping function.

Our training algorithm based on PA is shown in Algorithm 2.

input : mixed heterogeneous datasets: $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, N$;
parameters: $\mathcal{C}, K$;
loose mapping function: φ ;
output: $\mathbf{w}_K$

Initialize: $\mathbf{wTemp} \leftarrow 0, \mathbf{w} \leftarrow 0$;
for $k = 0 \cdots K-1$ **do**
  for $i = 1 \cdots N$ **do**
    receive an example $(\mathbf{x}_i, \mathbf{y}_i)$;
    predict: $\hat{\mathbf{y}}_i$ with Eq.(11);
    if ***hinge loss*** $\ell_w > 0$ **then**
      update $\mathbf{w}$ with Eq. (15);
    end
  end
  $\mathbf{wTemp} = \mathbf{wTemp} + \mathbf{w}$;
end
$\mathbf{w}_K = \mathbf{wTemp}/K$;

Algorithm 2: Training Algorithm

6.1 Analysis

Although our mapping function between two heterogeneous annotations is loose and uncertain, our online training method can automatically increase the relative weights of features from the beneficial mapping relations and decrease the relative weights of features from the unprofitable mapping relations.

Consider an illustrative loose mapping relation "NN/CTB↔n,nt,nz/PDD". For an input sequence $\mathbf{x}$ and PDD-style output is expected. If the algorithm tagging a character as "n/PDD"(with help of the weight of "NN/CTB") and the right tag isn't one of "n,nt,nz/PDD", the weight of "NN/CTB" will also be decreased, which is reasonable since it is beneficial to distinguish the right tag. And if the right tag is one of "n,nt,nz/PDD" but not "n/PDD" (for example, "nt/PDD"), which means it is a "NN/CTB", the weight of "NN/CTB" will remain unchanged according to the algorithm (updating "n/PDD" changes the "NN/CTB", but updating "nt/PDD" changes it back).

Therefore, after multiple iterations, useful features derived from the mapping function are typically receive more updates, which take relatively more responsibility for correct prediction. The final model has good parameter estimates for the shared information.

We implement our method based on FudanNLP(Qiu et al., 2013).

7 Experiments

7.1 Datasets

We use the two representative corpora mentioned above, Penn Chinese Treebank (CTB) and PKU's People's Daily (PPD) in our experiments.

Dataset	Partition	Sections	Words
CTB-5	Training	1−270 400−931 1001−1151	0.47M
	Develop	301−325	6.66K
	Test	271−300	7.82K
CTB-S	Training		0.64M
	Test	-	59.96K
PPD	Training	-	1.11M
	Test	-	0.16M

Table 3: Data partitioning for CTB and PD

7.1.1 CTB Dataset

To better comparison with the previous works, we use two commonly used criterions to partition CTB dataset into the train and test sets.

- One is the partition criterion used in (Jin and Chen, 2008; Jiang et al., 2009; Sun and Wan, 2012) for CTB 5.0.
- Another is the CTB dataset from the POS tagging task of the Fourth International Chinese Language Processing Bakeoff (SIGHAN Bakeoff 2008)(Jin and Chen, 2008).

7.1.2 PPD Dataset

For the PPD dataset, we use the PKU dataset from SIGHAN Bakeoff 2008.

The details of all datasets are shown in Table 3. Our experiment on these datasets may lead to a fair comparison of our system and the related works.

7.2 Setting

We conduct two experiments on **CTB-5 + PPD** and **CTB-S + PPD** respectively.

The form of feature templates we used is shown in Table 7.2, where C represents a Chinese character, and T represents the character-based tag. The subscript i indicates its position related to the current character.

Our method can be easily combined with some other complicated models, but we only use the simple one for the purpose of observing the sole influence of our unified model. The parameter C is tested on develop dataset, and we found that it just impact the speed of convergence and have no effect on the accuracy. Moreover, since we use the averaged strategy, we wish more iterations to avoid overfitting and set a small value 0.01 to it. The maximum number of iterations K is 50.

$C_i, T_0 (i = -2, -1, 0, 1, 2)$
$C_i, C_{i+1}, T_0 (i = -1, 0)$
T_{-1}, T_0

Table 4: Feature Templates

The $F1$ score is used for evaluation, which is the harmonic mean of precision P (percentage of predict phrases that exactly match the reference phrases) and recall R (percentage of reference phrases that returned by system).

7.3 Evaluation on CTB-5 + PPD

The experiment results on the heterogeneous corpora CTB-5 + PPD are shown in Table 5. Our method obtains an error reductions of 24.08% and 90.8% over the baseline on CTB-5 and PDD respectively.

Our method also gives better performance than the pipeline-based methods on heterogeneous corpora, such as (Jiang et al., 2009) and (Sun and Wan, 2012).

The reason is that our model can utilize the information of both corpora effectively, which can boost the performance of each other.

Although the loose mapping function are bidirectional between two annotation tagsets, we may also use unidirectional mapping. Therefore, we also evaluate the performance when we use unidirectional mapping. We just use the mapping function $\psi_{\mathbf{PDD}\rightarrow\mathbf{CTB}}$, which means we obtain the PDD-style output without the information from CTB in tagging stage. Thus, in training stage, there are no updates for the weights of CTB-features for the instances from PDD corpus, while instances from CTB corpus can result to updates for PDD-features.

Surprisedly, we find that the one-way mapping can also improve the performances of both corpora. The results are shown in Table 7. The

Method	Training Dataset	Test Dataset	P	R	F1
(Jiang et al., 2009)	CTB-5, PDD	CTB-5	-	-	94.02
(Sun and Wan, 2012)	CTB-5, PDD	CTB-5	94.42	94.93	94.68
Our Model	CTB-5	CTB-5	93.28	93.35	93.31
Our Model	PDD	PDD	89.41	88.58	88.99
Our Model	CTB-5, PDD	CTB-5	**94.74**	**95.11**	**94.92**
Our Model	CTB-5, PDD	PDD	**90.25**	**89.73**	**89.99**

Table 5: Performances of different systems on CTB-5 and PPD.

Method	Training Dataset	Test Dataset	P	R	F1
Our Model	CTB-S	CTB-S	89.11	89.16	89.13
Our Model	PDD	PDD	89.41	88.58	88.99
Our Model	CTB-S, PDD	CTB-S	**89.86**	**90.02**	**89.94**
Our Model	CTB-S, PDD	PDD	**90.5**	**89.82**	**90.16**

Table 6: Performances of different systems on CTB-S and PPD.

model$_{PPD \rightarrow CTB}$ obtains an error reductions of 14.63% and 6.12% over the baseline on CTB-5 and PDD respectively.

Method	P	R	F1
Model$_S$ on CTB-5	93.86	94.73	94.29
Model$_S$ on PDD	90.05	89.28	89.66

"Model$_S$" is the model which is trained on both CTB-5 and PDD training datasets with just just using the unidirectional mapping function $\psi_{\mathbf{PDD \rightarrow CTB}}$.

Table 7: Performances of unidirectional PPD→CTB mapping on CTB-5 and PPD.

7.4 Evaluation on CTB-S + PPD

Table 6 shows the experiment results on the heterogeneous corpora CTB-S + PPD. Our method obtains an error reductions of 7.41% and 10.59% over the baseline on CTB-S and PDD respectively.

7.5 Analysis

As we can see from the above experiments, our proposed unified model can improve the performances of the two heterogeneous corpora with unidirectional or bidirectional loose mapping functions. Different to the pipeline-based methods, our model can use the shared information between two heterogeneous POS taggers. Although the mapping function is loose and uncertain, it is still can boost the performances. The features derived from the wrong mapping function take relatively less responsibility for prediction after multiple updates of their weights in training stage. The final model has good parameter estimates for the shared information.

Another phenomenon is that the performance of one corpus can gains when the data size of another corpus increases. In our two experiments, the training set's size of CTB-S is larger than CTB-5, so the performance of PDD is higher in latter experiment.

8 Conclusion

We proposed a method for joint Chinese word segmentation and POS tagging with heterogeneous annotation data. Different to the previous pipeline-based works, our model is learned on heterogeneous annotation data simultaneously. Our method also does not require the exact corresponding relation between the standards of heterogeneous annotations. The experimental results show our method leads to a significant improvement with heterogeneous annotations over the best performance for this task. Although our work is for a specific task on joint Chinese word segmentation and POS, the key idea to leverage heterogeneous annotations is very general and applicable to other NLP tasks.

In the future, we will continue to refine the proposed model in two ways: (1) We wish to use the unsupervised method to extract the loose mapping relation between the different annotation standards, which is useful to the corpora without loose mapping guideline. (2) We will analyze the shared information (weights of the features derived from the tags which have the mapping relation) in detail and propose a more effective model. Besides, we would also like to investigate for other NLP tasks which have different annotation-style corpora.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was funded by NSFC (No.61003091), Key Projects in the National Science & Technology Pillar Program (2012BAH18B01), Shanghai Municipal Science and Technology Commission (12511504500), Shanghai Leading Academic Discipline Project (B114) and 973 Program (No.2010CB327900).

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, December.

S. Ben-David and R. Schuller. 2003. Exploiting task relatedness for multiple task learning. *Learning Theory and Kernel Machines*, pages 567–580.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

J. Gao, A. Wu, M. Li, C.N. Huang, H. Li, X. Xia, and H. Qin. 2004. Adaptive chinese word segmentation. In *Proceedings of ACL-2004*.

W. Jiang, L. Huang, Q. Liu, and Y. Lu. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Citeseer.

W. Jiang, L. Huang, and Q. Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging: a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*, pages 522–530.

Wenbin Jiang, Fandong Meng, Qun Liu, and Yajuan Lü. 2012. Iterative annotation transformation with predict-self reestimation for Chinese word segmentation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 412–420, Jeju Island, Korea, July. Association for Computational Linguistics.

C. Jin and X. Chen. 2008. The fourth international Chinese language processing bakeoff: Chinese word segmentation, named entity recognition and Chinese pos tagging. In *Sixth SIGHAN Workshop on Chinese Language Processing*, page 69.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768. Association for Computational Linguistics.

H.T. Ng and J.K. Low. 2004. Chinese part-of-speech tagging: one-at-a-time or all-at-once? word-based or character-based. In *Proceedings of EMNLP*, volume 4.

Xipeng Qiu, Feng Ji, Jiayi Zhao, and Xuanjing Huang. 2012. Joint segmentation and tagging with coupled sequences labeling. In *Proceedings of COLING 2012*, pages 951–964, Mumbai, India, December. The COLING 2012 Organizing Committee.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. FudanNLP: A toolkit for Chinese natural language processing. In *Proceedings of ACL*.

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for Chinese lexical processing with heterogeneous annotations. In *Proceedings of the 50th Annual Meeting of the*

Association for Computational Linguistics, pages 232–241.

W. Sun. 2011. A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1385–1394.

F. Xia, 2000. *The part-of-speech tagging guidelines for the penn Chinese treebank (3.0)*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM.

S. Yu, J. Lu, X. Zhu, H. Duan, S. Kang, H. Sun, H. Wang, Q. Zhao, and W. Zhan. 2001. Processing norms of modern Chinese corpus. Technical report, Technical report.

Jiayi Zhao, Xipeng Qiu, and Xuanjing Huang. 2013. A unified model for joint chinese word segmentation and pos tagging with heterogeneous annotation corpora. In *International Conference on Asian Language Processing, IALP*.

The Topology of Semantic Knowledge

Jimmy Dubuisson **Jean-Pierre Eckmann**
Département de Physique Théorique and Section de Mathématiques
Université de Genève
Jimmy.Dubuisson@unige.ch

Christian Scheible
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
scheibcn@ims.uni-stuttgart.de

Hinrich Schütze
Center for Information
and Language Processing
University of Munich

Abstract

Studies of the graph of dictionary definitions (DD) (Picard et al., 2009; Levary et al., 2012) have revealed strong semantic coherence of local topological structures. The techniques used in these papers are simple and the main results are found by understanding the structure of cycles in the directed graph (where words point to definitions). Based on our earlier work (Levary et al., 2012), we study a different class of word definitions, namely those of the Free Association (FA) dataset (Nelson et al., 2004). These are responses by subjects to a cue word, which are then summarized by a directed, free association graph.

We find that the structure of this network is quite different from both the Wordnet and the dictionary networks. This difference can be explained by the very nature of free association as compared to the more "logical" construction of dictionaries. It thus sheds some (quantitative) light on the psychology of free association.

In NLP, semantic groups or clusters are interesting for various applications such as word sense disambiguation. The FA graph is tighter than the DD graph, because of the large number of triangles. This also makes drift of meaning quite measurable so that FA graphs provide a quantitative measure of the semantic coherence of small groups of words.

1 Introduction

The computer study of semantic networks has been around since the advent of computers (Brunet, 1974) and has been used to study semantic relations between concepts and for analyzing semantic data. Traditionally, a popular lexical database of English is Wordnet (Miller, 1995; Miller and Fellbaum, 1998), which organizes the semantic network in terms of graph theory. In contrast to manual approaches, the automatic analysis of semantically interesting graph structures of language has received increasing attention. For example, it has become clear more recently that cycles and triangles play an important role in semantic networks, see *e.g.*, (Dorow et al., 2005). These results suggest that the underlying semantic structure of language may be discovered through graph-theoretical methods. This is in line with similar findings in much wider realms than NLP (Eckmann and Moses, 2002).

In this paper, we compare two different types of association networks. The first network is constructed from an English dictionary (DD), the second from a free association (FA) database (Nelson et al., 2004). We represent both datasets through directed graphs. For DD, the nodes are words and the directed edges point from a word to its definition(s). For FA, the nodes are again words, and each cue word has a directed edge to each association it elicits.

Although the links in these graphs were not constructed by following a rational centralized process, their graph exhibits very specific features and we concentrate on the study of its topological properties. We will show that these graphs are quite different in global and local structure, and we interpret this as a reflection of the different nature of DD vs. FA. The first is an *objective* set of rela-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 669–680,
Seattle, Washington, USA, 18-21 October 2013.

tions between words and their meaning, as explained by other words, while the second reveals the nature of *subjective* reactions to cue words by individuals. This matter of fact is reflected by several quantitative differences in the structure of the corresponding graphs.

The main contribution of this paper is an empirical analysis of the way semantic knowledge is structured, comparing two different types of association networks (DD and FA). We conduct a mathematical analysis of the structure of the graphs to show that the way humans express their thoughts exhibits structural properties in which one can find semantic patterns. We show that a simple graph-based approach can leverage the information encoded in free association to narrow down the ambiguity of meaning, resulting in precise semantic groups. In particular, we find that the main strongly connected component of the FA graph (the so-called **core**) is very cyclic in nature and contains a large predominance of short cycles (*i.e.*, co-links and triangles). In contrast to the DD graph, bunches of triangles form well-delimited lexical fields of collective semantic knowledge. This property may be promising for downstream tasks. Further, the methods developed in this paper may be applicable to graph representations that occur in other problems such as word sense disambiguation (*e.g.*, (Heylighen, 2001; Agirre and Soroa, 2009)) or sentiment polarity induction (Hassan and Radev, 2010; Scheible, 2010).

To show the semantic coherence of these lexical fields of the FA graph, we perform an experiment with human raters and find that cycles are strongly semantically connected even when compared to close neighbors in the graph.

The reader might wonder why sets of pairwise associations can lead to any interesting structure. One of the deep results in graph theory, (Bollobás, 2001), is that in sparse graphs, *i.e.*, in graphs with few links per node, the number of triangles is extremely rare. Therefore, if one does find many triangles in a graph, they must be not only a signal of non-randomness, but carry relevant information about the domain of research as shown earlier (Eckmann and Moses, 2002).

2 The USF FA dataset

This dataset is one of the largest existing databases of free associations (FA) and has been collected at the University of South Florida since 1973 by researchers in psychology (Nelson et al., 2004). Over the years, more than 6'000 participants produced about 750'000 responses to 5'019 stimulus words.

The procedure for collecting the data is called discrete association task and consists in asking participants to give the first word that comes to mind (**target**) when presented a stimulus word (**cue**).

For creating the initial set of stimulus words, the Jenkins and Palermo word association norms (Palermo and Jenkins, 1964) proved useful but too limited as they consist of only 200 words. For this reason, additional words have been regularly added to the pool of normed words, unfortunately without well established rules being followed. For instance, some were selected as potentially interesting cues, some were added as responses to the first sets of cues and, some others were collected for supporting new studies on verbs. We still work with this database, because of its breadth.

The final pool of stimuli comprises 5'019 words of which 76% are nouns, 13% adjectives, and 7% verbs. A word association is said to be **normed** when the target is also part of the set of norms, *i.e.*, a cue. The USF dataset of free associations contains 72'176 cue-target pairs, 63'619 of which are normed. As an example, the association *puberty-sex* is normed whereas the association *puberty-thirteen* is not, because *thirteen* is not a cue.

3 Mathematical definitions

We collect here those notions we need for the analysis of the data.

A **directed graph** is a pair $G = (V, E)$ of a set V of **vertices** and, a set E of ordered pairs of vertices also called **directed edges**. For a directed edge $(u, v) \in E$, u is called the **tail** and v the **head** of the edge. The number of edges incident to a vertex $v \in V$ is called the **degree** of v. The **in-degree** (resp. **out-degree**) of a vertex v is the number of edge heads (resp. edge tails) adjacent to it. A vertex with null in-degree is called a **source** and a vertex with null out-degree is called a **sink**.

A **directed path** is a sequence of vertices such

that a directed edge exists between each consecutive pair of vertices of the graph. A directed graph is said to be **strongly connected**, (resp. **weakly connected**) if for every pair of vertices in the graph, there exists a directed path (resp. undirected path) between them. A **strongly connected component**, SCC, (resp. **weakly connected component**, WCC) of a directed graph G is a maximal strongly connected (resp. weakly connected) subgraph of G.

A **directed cycle** is a directed path such that its **start vertex** is the same as its **end vertex**. A **colink** is a directed cycle of length 2 and a **triangle** a directed cycle of length 3.

The **distance** between two vertices in a graph is the number of edges in the shortest path connecting them. The **diameter** of a graph G is the greatest distance between any pair of vertices. The **characteristic path length** is the average distance between any two vertices of G.

The **density** of a directed graph $G(V, E)$ is the proportion of existing edges over the total number of possible edges and is defined as:

$$d = |E|/(|V|(|V| - 1))$$

The **neighborhood** N_i of a vertex v_i is $N_i = \{v_j : e_{ij} \in E \text{ or } e_{ji} \in E\}$.

The **local clustering coefficient** C_i for a vertex v_i corresponds to the density of its neighborhood subgraph. For a directed graph, it is thus given by:

$$C_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{|N_i|(|N_i| - 1)}$$

The **clustering coefficient** of a graph G is the average of the local clustering coefficients of all its vertices.

The **efficiency** Eff of a directed graph G is an indicator of the traffic capacity of a network. It is the harmonic mean of the distance between any two vertices of G. It is defined as:

$$\text{Eff} = \frac{1}{|V|(|V| - 1)} \sum_{i \neq j \in V} \frac{1}{d_{ij}}$$

The linear correlation coefficient between two random variables X and Y is defined as:

$$\rho(X, Y) = (E[XY] - \mu_X \mu_Y)/(\sigma_X \sigma_Y)$$

where μ_X and σ_X are respectively the mean and standard deviation of the random variable X.

The linear degree correlation coefficient of a graph is called **assortativity** and is expressed as:

$$\rho_D = \sum_{xy} xy(e_{xy} - a_x b_y)/(\sigma_a \sigma_b)$$

where e_{xy} is the fraction of all links that connect nodes of degree x and y and where a_x and b_y are respectively the fraction of links whose tail is adjacent to nodes with degree x and whose head is adjacent to nodes with degree y, satisfying the following three conditions:

$$\sum_{xy} e_{xy} = 1, a_x = \sum_{y} e_{xy}, b_y = \sum_{x} e_{xy}$$

When ρ_D is positive, the graph possesses **assortative mixing** and high-degree nodes tend to connect to other high-degree nodes. On the other hand, when ρ_D is negative, the graph features **disassortative mixing** and high-degree nodes tend to connect to low degree nodes.

The **intersection graph** of sets A_i, $i = 1, \ldots, m$, is constructed by representing each set A_i as a vertex $v_i \in V$ and adding an edge for each pair of sets with a non-empty intersection:

$$E = \{(v_i, v_j) : A_i \cap A_j \neq \varnothing\}$$

4 Graph topology analysis

4.1 Graph generation

Our goal being to study the FA network topology, we first concentrate on the generation of an unweighted directed graph. We generate the corresponding graph by adding a directed edge for each cue-target pair of the dataset. We only consider pairs whose target was normed in order to avoid overloading the graph with noisy data (*e.g.*, a response meaningful only to a specific participant). The graph has 5'019 vertices and 63'619 edges. It is composed of a single WCC and 166 SCCs.

For comparison with dictionary definitions (DD), we construct a graph from the Wordnet2 dictionary (nouns only), following (Levary et al., 2012). This graph contains 54'453 vertices and 179'848 edges.

4.2 Core extraction

The so-called **core** was defined previously in (Picard et al., 2009; Levary et al., 2012) as that subset of nodes in which a random walker gets trapped after only a few steps.

The shave algorithm was used in (Levary et al., 2012) to isolate this subset. It consists in recursively removing the source and sink nodes from a weakly connected directed graph and permits to get the subgraph induced by the union of its strongly connected components. Note that the dictionary graph (DD) has no sinks (*i.e.*, words that never get defined) and that it contains a giant SCC whose size is comparable to the one of the initial graph.

It turns out that the FA graph also contains a giant SCC, therefore getting the core consists more simply in extracting the main SCC of the initial graph. We use Tarjan's algorithm (Tarjan, 1972) for isolating the FA core.

4.3 Vertex degree analysis

The FA core has a maximum in-degree of 313, a maximum out-degree of 33 and an average degree of 25.42. The in-degree distribution follows a power law ($\gamma = 1.93$) and the out-degrees are Poisson-like distributed with a peak at 14 (Steyvers and Tenenbaum, 2005; Gravino et al., 2012).

Words having a high in-degree are **targets** that tend to be cited more frequently. On the other hand, words having a high out-degree are **cues** that evoke many different targets.

The most evocative cues are, in decreasing order of out-degree: *field* (33), *body* (31), *condemn* (29), *farmer* (29), *crisis* (28), *plan* (28), *attention* (27), *animal* (27), and *hang* (27). Interestingly, the most cited targets (*i.e.*, targets with highest in-degree) are in decreasing order: *food* (313), *money* (295), *water* (271), *car* (251), *good* (246), *bad* (221), *work* (187), *house* (183), *school* (182), *love* (179).

4.4 Cycle decomposition of the core

We define the **vertex k-cycle multiplicity** (resp. **edge k-cycle multiplicity**) as the number of k-cycles a given vertex (resp. edge) belongs to. We call **core-ER** the set of Erdös-Rényi (ER) random graphs $G(n, M)$ having the same number of nodes and the same number of edges as the FA core. We start by extracting the 2- and 3-cycles by using a customized version of Johnson's algorithm (Johnson, 1975). The first thing we observe is that the core has a very high density of short cycles: the subset of nodes belonging to 2-cycles (*i.e.*, nodes with 2-cycle multiplicities > 0) cover 95% of the core vertices and the 3-cycles cover 88% of the core vertices. The corresponding core-ER graphs have on average about 100 times fewer 2-cycles and almost 20 times fewer 3-cycles.

This shows that the core is very cyclic in nature and that it remains very well connected for short-length cycles: most vertices of the core indeed belong to at least one co-link or triangle.

In order to limit computation times, we only considered shortest cycles for lengths ≥ 3 and analyzed the distribution of the number of shortest cycles in the core compared to equivalent random graphs. Whereas there are many more short cycles in the core, we observe a predominance of 4, 5 and 6-cycles in core-ER graphs. However, we find again a slight predominance of long cycles (length between 7 and 15) in the core (see Fig. 1). See (Levary et al., 2012), Fig. 3, where the cycle distribution is very different, with a minimum at length 5.

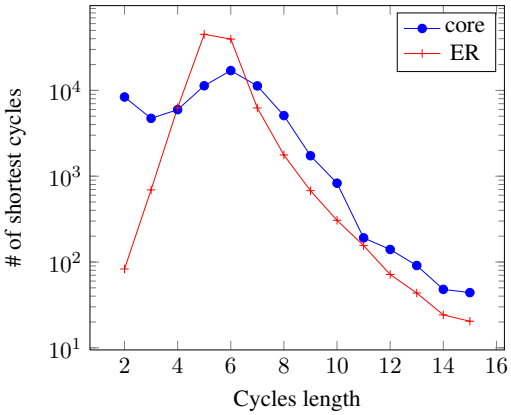


Figure 1: Distribution of shortest cycles lengths in the core compared to equivalent ER models

One should bear in mind that we only consider the set of *shortest* cycles. Thus, a k-cycle is not counted if each of its nodes belongs to a cycle whose length is $< k$. Although the number of 4-shortest cycles is comparable in the core and core-ER graphs for example, there are in reality far more 4-cycles in the core (*i.e.*, 42'738 versus 6'517). We see that when considering shortest cycles, short cycles tend to hide long ones, and, as a large proportion of nodes in the core belong to 2- and 3-cycles, many longer cycles do not get counted at all.

4.5 Interpretation of cycles

2-cycles are composed of concretely related words (*e.g.*, *drug-coke*, *destiny-fate*, *einstein-genius*, ...). The vertex with highest 2-cycle multiplicity is *music* (22).

Words in 3- and 4-cycles often belong to the same lexical field. Examples of 3-cycles: *protect-guard-defend* or *space-universe-star*. The vertex (resp. edge) with highest 3-cycle multiplicity is *car* (86) (resp. *bad-crime* (11)). Examples of 4-cycles: *monster-dracula-vampire-ghost* or *flu-virus-infection-sick*.

Longer cycles are more difficult to describe: Relations linking words of a given cycle exhibit semantic drift with increasing length (cf. (Levary et al., 2012)). Examples of 5-cycles: *yellow-coward-chicken-soup-noodles* and *sleep-relax-music-art-beauty*.

The cumulated set of free associations reflects the way in which a group of people retrieved its semantic knowledge. As the associated graph is highly circular, this suggests that this knowledge is not stored in a hierarchical way (Steyvers and Tenenbaum, 2005). The large predominance of short cycles in the core may indeed be a natural consequence of the semantic information being acquired by means of associative learning (Ashcraft and Radvansky, 2009; Shanks, 1995).

4.6 FA core clustering

4.6.1 The walktrap community algorithm

Complex networks are globally sparse but contain locally dense subgraphs. These groups of highly interconnected vertices are called **communities** and convey important properties of the network.

Although the notion of community is difficult to define formally, the current consensus establishes that a partition $P = \{C_1, C_2, \ldots, C_k\}$ of the vertex set of a graph G represents a good community structure if the proportion of edges inside the C_i is higher than the proportion of edges between them (Fortunato, 2010).

Computing such communities in a large graph is generally computationally expensive (Lancichinetti and Fortunato, 2009). We use the so-called 'Walktrap' community detection algorithm (Pons and Latapy, 2006) for extracting communities from the FA networks. The idea lying behind this algorithm is that random walks on a graph will tend to get trapped in the densely connected subgraphs.

Let P_{ij}^t be the probability of going from vertex i to vertex j through a random walk of length t. The distance between two vertices i and j of the graph is defined as:

$$r_{ij}(t) = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}}$$

where $d(k)$ is the degree of vertex k.

One defines the probability $P_{C,j}^t$ to go from community C to vertex j in t steps: $P_{C,j}^t = \sum_{i \in C} P_{ij}^t / |C|$, and then the distance is easily generalized for two communities C_1, C_2.

The algorithm starts with a partition $P_1 = \{\{v\} \in V\}$ of the initial graph into n communities each of which is a single vertex. At each step, two communities are chosen and merged according to the criterion described below and the distances between communities are updated. The process goes on until we obtain the partition $P_n = \{V\}$.

In order to reduce complexity, only adjacent communities are considered for merging. The decision is then made according to Ward's method (Everitt et al., 2001): at each step k, the two communities that minimize the mean σ_k of the squared distances between each vertex and its community are merged:

$$\sigma_k = \frac{1}{n} \sum_{C \in P_k} \sum_{i \in C} r_{iC}^2$$

4.6.2 Clustering of the core

We first identify the communities of the FA core using the Walktrap algorithm. We immediately observe that when the path length parameter increases, the number of identified communities decreases (*i.e.*, for a length of 2, we find 35 communities whereas for a length of 9, we only find 8 communities).

For a path length of 2, the algorithm extracts 35 communities, 7 of which contain more than 100 vertices, 3 of which contain between 100 and 50 vertices and 25 of which contain less than 50 vertices.

We observe that for most small communities (*i.e.*, the ones containing less than 50 vertices), there exists a clear relation between the labels of their ver-

tices. Typically, the labels are part of the same lexical field (*e.g.*, all the planets (except *earth*) or related by a common grammatical function (such as *why*, *where*, *what*, ...).

4.6.3 Clustering of the core co-links

We define the **k-cycle induced subgraph** of a graph G as the subgraph of G induced by the set of its vertices with k-cycle multiplicity > 0.

The **co-link graph** of a graph $G(V, E)$ is the undirected graph obtained by replacing each co-link (*i.e.*, 2-cycle) of the 2-cycle induced subgraph of G by a single undirected edge and removing all other edges.

The co-link graph of the FA core has 4'508 vertices and 8'309 edges for a density of 8×10^{-4}. It is composed of a single weakly connected component that can be seen as a projection of the strongest semantic links from the original graph. Extracting the co-link graph is thus an efficient way of selecting the set of most important semantic links (*i.e.*, the set of 2-cycles that appear in large predominance in the core compared to what is found in an equivalent random graph) while filtering out the noisy or negligible ones.

The sets of communities extracted by the Walktrap algorithm exhibit different degrees of granularity depending on the length parameter. For short paths, a large number of very small communities are returned (*e.g.*, 923 communities when length equals 2) whereas for longer paths the average size of the communities increases more and more.

The community detection exhibits thus a far finer degree of granularity for the core co-links graph than for the core itself. The size of the communities being much smaller in average, it is striking to notice to which extent the words of a given community are semantically related.

Examples of communities found in the core co-links graph include (*standards*, *values*, *morals*, *ethics*), (*hopeless*, *romantic*, *worthless*, *useless*), (*thesaurus*, *dictionary*, *vocabulary*, *encyclopedia*) or (*molecule*, *atom*, *electron*, *nucleus*, *proton*, *neutron*).

4.6.4 DD core clustering vs FA core clustering

The clustering of both cores has very different characteristics: We illustrate the neighborhoods of *conflict* for both cases in Fig. 2 and 3.

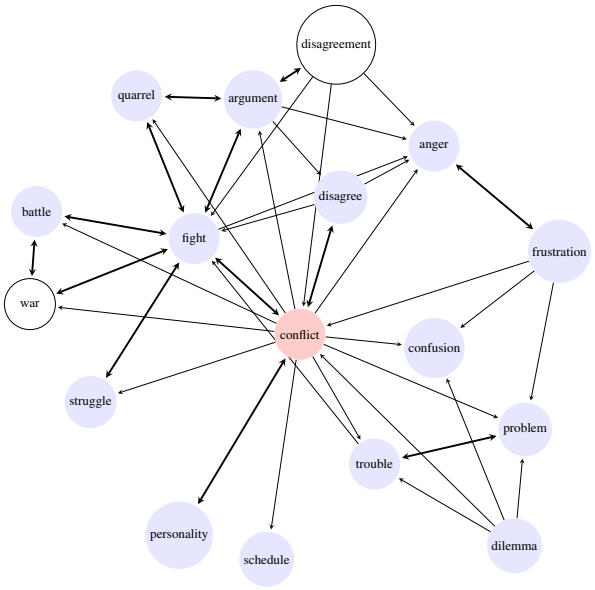


Figure 2: Neighborhood of *conflict* in the FA core
The set of words belonging to the neighborhood of *conflict* are clearly part of the same lexical field. The high density of co-links leads to cyclicity and we see that many directed triangles are present in the local subgraph (*e.g.*, *conflict-trouble-fight, conflict-argument-disagree*). We can even find triangles of co-links that link together words semantically strongly related (*e.g.*, *fight-war-battle, fight-quarrel-argument*). Nodes that are part of the neighborhood of *conflict* in both FA and DD are in empty circles.

On one hand, the words in communities of the DD core are in most cases either synonyms, *e.g.*, *(declaration, assertion, claim)* or an instance-of kind of relation, *e.g.*, *(signal, gesture, motion)* or *(zero, integer)*.

On the other hand, communities of the FA core are generally composed of words belonging to the same lexical field and sharing the same level of abstraction.

Moreover, we notice that it is often difficult to establish the semantic relation existing between words of many small communities (*i.e.*, containing less than 10 words) of the DD core. Two such examples are: (*choice*, *probate*, *executor*, *chosen*, *certificate*, *testator*, *will*) and (*numeral*, *monarchy*, *monarch*, *crown*, *significance*, *autocracy*, *symbol*, *interpretation*).

The comparison of DD and FA reveals, in a quantitative way, fundamental differences between the two realms. The interesting data are shown in table 1.

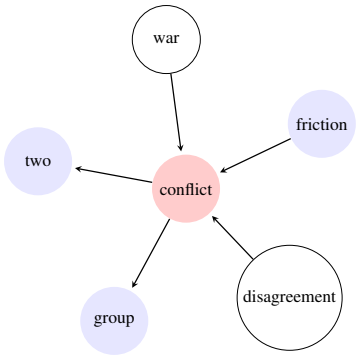


Figure 3: Neighborhood *conflict* in the DD core
First, we note that the neighborhood has a lower density than in the FA core. We also see that there is no cycle and there seems to be a flow going from source nodes to sink nodes. As it generally happens in the neighborhood subgraphs of the DD core, source nodes are rather specific words whereas sink nodes are generic words.

	FA core	DD core
# vertices	4'843	1'496
# edges	61'544	4'766
density	2.5×10^{-3}	2.1×10^{-3}
avg degree	25.4	6.37
max in-degree	313	59
directed diameter	10	29
characteristic path length	4.26	10.42
efficiency	2.5×10^{-1}	1.2×10^{-1}
clustering coefficient	8.5×10^{-2}	5.1×10^{-2}
assortativity	5.5×10^{-2}	6.1×10^{-2}

Table 1: Comparison FA vs DD

Note that while the FA core is in fact larger than the DD core, its diameter is smaller. This illustrates in a beautiful way the nature of free association as compared to the more neutral dictionary. In particular, the characteristic path length is smaller in the FA graph, because humans use generalized event knowledge (McRae and Matsuki, 2009) in free association, producing semantic shortcuts. For example, FA contains a direct link *mirage→water*, whereas in DD, the shortest path between the two words is *mirage→refraction→wave→water*.

5 The Bricks of Meaning

5.1 Extraction of the seed

We already saw that most vertices of the core belong to directed 2- and 3-cycles. Whereas 2-cycles establish strong semantic links (*i.e.*, synonymy or antonymy relations) and provide cyclicity to the underlying directed graph, we claim that 3-cycles (*i.e.*, triangles) form the set of elementary concepts of the core.

These structues are common to DD and to FA, but we will see that the links in FA are somehow more direct than in DD.

We call **seed** the subgraph of the core induced by the set V_3 of vertices belonging to directed triangles and **shell** the subgraph of the core induced by the set $V \backslash V_3$ (*i.e.*, the set of vertices with a null 3-cycle multiplicity), see Fig. 4.

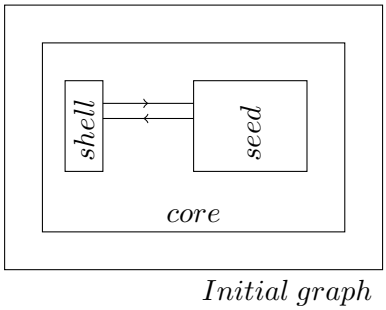


Figure 4: Composition of the FA graph
The graph of FA contains a giant SCC (the core). The subgraph of the core induced by the set of nodes belonging to at least one triangle also forms a giant component we call the 'seed'. The subgraph of the core induced by the set of nodes not belonging to any triangle is called the 'shell' and is composed of many small SCCs, including single vertices. Although the shell has a low density, its nodes are very well connected to the seed.

The shell contains 530 nodes and 309 edges. There are 7'035 edges connecting the shell to the seed. The shell consists of of many small SCCs and although its average degree is low (1.17), its vertices have on average many (13.27) connections to the seed.

The seed contains 4'313 vertices (89% of the core) and 54'197 edges. The first thing to notice is that it has 100 times more co-links (7'895) and 20 times more triangles (13'119) than an equivalent random graph. We call **shortcuts** the 32'773 edges of the seed that do not belong to 3-cycles, see Fig. 5.

The seed obviously also contains cycles whose length is greater than 3. One can check that there exist only 5 basic motifs involving 2 attached triangles and 1 shortcut for creating 4- and 5-cycles, and that linking 2 isolated triangles with 2 shortcuts also per-

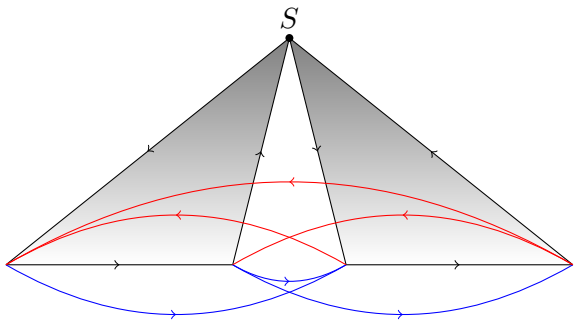


Figure 5: Shortcut edges between two triangles sharing a single vertex S
Two triangles can share 0, 1 or 2 vertices. For each of these three basic motifs, we count the maximum number of shortcut edges (*i.e.*, edges not belonging to 3-cycles) that can be added. By linking two triangles, these shortcuts permit to move two basic semantic units closer together and create longer cycles (*i.e.*, 4, 5, and 6-cycles). Long cycles can be thus considered as groupings of basic semantic units. In the case of two triangles sharing one vertex for example, it is possible to add at most 6 shortcuts, whereas, for two triangles sharing two vertices, at most 2 shortcuts can be added.

mit to form 4-, 5- and 6-cycles. All longer cycles are simply made of a juxtaposition of these basic motifs.

Furthermore, there is a limit on the number of shortcuts that can possibly be added in the seed before it gets saturated, as all its vertices belong to at least one triangle. We show that at most 16 shortcuts can be added between two isolated triangles, at most 6 between 2 triangles sharing 2 vertex and at most 2 between 2 triangles sharing 2 vertices (see Fig. 5).

5.2 The elementary lexical fields

Once the seed is isolated, we go on digging into its structure. We focus on the arrangements of triangles as they constitute the set of elementary concepts.

We start by removing all shortcuts from the seed and convert it then to an undirected graph, in order to get a homogeneous simplicial 2-complex.

Let **t** be the graph operator which transforms a graph G into the intersection graph $\mathbf{t}G$ of its 2-simplices (*i.e.*, triangles sharing an edge). We apply **t** to the homogeneous simplicial 2-complex found previously. The result represents the links between the basic semantic units of the seed. We call **seed-crux** the giant WCC in the intersection graph.

We enumerate the 8'380 maximal cliques of FA seed-crux and get the list of words composing each

Distance	Acc	κ	KS
original	–	0.404	30
1	74	0.522	42
2	97	0.899	89
∞	99	0.899	89

Table 2: Accuracy, κ, and *count*$(p < 0.05)$ for KS

clique. By removing the ones that are subsets of bigger lists, we finally obtain 3'577 lists of words .

These lists of words have a rather small and homogeneous size (between 4 and 17) and 95% have a size comprised between 4 and 10. More interestingly, they clearly define well-delimited lexical fields. We will show this through two experiments in the following sections. A few examples include *(honest, trustworthy, reliable, responsible)*, *(stress, problem, worry, frustration)* and *(data, process, computer, information)*.

From a topological perspective, we deduce that bunches of triangles (*i.e.*, cliques of elementary concepts) span the seed in a homogeneous way. These bunches form a set of cohesive lexical fields and constitute essential bricks of semantic knowledge.

5.3 Semantic similarity of the lexical fields

In order to quantify the relative meaning of words in the lexical fields of the seed-crux, we define the following semantic similarity metric based on the Wordnet *WUP* metric (Wu and Palmer, 1994) for a given set of words L:

$$S_\ell(L) = 2 \sum_{w_i,w_j \in L, w_i \neq w_j} S_w(w_i, w_j)/(|L|(|L|-1))$$

where $S_w(w_i, w_j) = \max_{S_k \ni w_i \text{and} S_\ell \ni w_j} \{wup(S_k, S_\ell)\}$ and wup is the WUP semantic metric and S_k and S_ℓ are Wordnet synsets.

The average value of S_ℓ for the set of cliques of seed-crux is 0.6 whereas it is only 0.43 for randomly sampled set of words. This suggests the corresponding lists of words are indeed semantically related. We will show the strength of this relation in the following experiment with human raters.

5.4 Human evaluation of the lexical fields

To validate our findings, we conducted an empirical evaluation through human annotators. Starting from

the 1'204 4-groups, we designed the following experiment: We corrupt the groups by exchanging one of the 4 elements with a randomly chosen word at a distance from the group of 1, 2, and "infinity" (*i.e.*, any word of the whole core). We presented 100 random samples for each of the 3 distances as well as 100 unperturbed groups (*original*) to annotators at Amazon Mechanical Turk[1], asking which word fits the group the least. Intuitively, the closer the randomly chosen words get to the group, the closer the distribution of the votes for each sample should be to the uniform distribution. We collected 10 votes for each of the 4 problems of 100 random samples. We calculated accuracy (*i.e.*, the relative frequency of correctly identified random words) for the 3 random confounder experiments and Fleiss' κ. Further, we used the Kolmogorov-Smirnov (KS) test for how uniform the label distribution is, reporting the relative frequency of samples that are significantly ($p < 0.05$) different from the uniform distribution. The results of this experiment are summarized in Table 2 and show clearly that the certainty about the "odd man out" increases together with the distance.

5.5 Error analysis

If we view our results as a resource for a downstream task, it is important to know about possible downsides. First, we note that there are words which are not in a triangle and will thus be missing in the intersection graph. This is an indication that the corresponding word is less well embedded contextually, so conversely, any prediction made about it from the data may be less reliable. Additionally, semantic leaps caused by generalized event knowledge may lead to lesser-connected groups such as (*steel*, *pipe*, *lead*, *copper*). Jumps like these may or may not be desired in a subsequent application.

6 The Case of the EAT FA dataset

The Edinburgh Associative Thesaurus (EAT) (Kiss et al., 1973) is a large dataset of free associations. We extract the EAT FA seed-crux with the previously described methods.

We start by generating the initial graph (23'219 vertices and 325'589 edges), then extract its core (7'754 vertices and 247'172 edges) and its seed (7'500 vertices and 238'677 edges). It is interesting to notice at this stage that the EAT seed contains 74% of the words belonging to the USF seed. After generating the seed-crux which contains 63'363 vertices, 6'825'731 edges, and 342'490 maximal cliques, we finally obtain 40'998 lists of words.

These lists comprise between 4 and 233 words but 80% of them have a relatively small size between 4 and 20. Although we find exceptions for this graph, most of the extracted lists again form well-delimited lexical fields (*e.g.*, *(health, resort, spa, bath, salts)* or *(god, devil, angel, satan)*.

Comparing the two association experiments, we see that the local topologies are quite similar. Both FA cores have a high density of connected triangles, whereas cycles in the DD graph tend to be longer and most triangles are isolated. This can be attributed to the different ways in which DD and FA are obtained, the former being built rationally by following a humanly-driven process and the latter reflecting an implicit collective semantic knowledge.

7 Related Work

A number of metrics like *Latent Semantic Analysis* (Deerwester et al., 1990) and *Word Association Spaces* (Steyvers et al., 2004) have been recently developed for quantifying the relative meaning of words. As the topological properties of free association graphs reflect key aspects of semantic knowledge, we believe some graph theory metrics could be used efficiently to derive new ways of measuring semantic similarity between words.

Topological analysis of the Florida Word Associations (FA) was started by (Steyvers and Tenenbaum, 2005; Gravino et al., 2012), who extracted global statistics. We follow the basic methodology of these studies, but extend their approach. First, we conduct deeper analyses by examining the neighborhood of nodes and extracting the statistics of cycles. Second, we compare the properties of FA and DD graphs.

Word clustering based on graphs has been the subject of various earlier studies. Close to our work is (Widdows and Dorow, 2002). These authors recognize that nearest-neighbor-based clustering of co-occurrence give rise to semantic groups. This type of approach has since been applied in various modified forms, *e.g.*, by (Biemann, 2006) who performs label-

[1] `http://www.mturk.com`

propagation based on randomized nearest neighbors, or Matsuo et al. (2006) who perform greedy clustering. Hierarchical clustering algorithms (*e.g.*, (Jonyer et al., 2002; Manning et al., 2008)) are related as well, however, the key difference is that in hierarchical clustering, the granularity of a cluster is difficult to determine.

Dorow et al. (2005) recognize that triangles form semantically strongly cohesive groups and apply clustering coefficients for word sense disambiguation. Their work focuses on undirected graphs of corpus co-occurrences whereas our work builds on directed associations. Building on this work, we take finer topological graph structures into account, which is one of the main contributions in this paper.

8 Conclusion

The cognitive process of discrete free association being an epiphenomenon of our semantic memory at work, the cumulative set of free associations of the USF dataset can be viewed as the projection of a collective semantic memory.

To analyze the semantic memory, we use the tools of graph theory, and compare it also to dictionary graphs. In both cases, triangles play a crucial role in the local topology and they form the set of elementary concepts of the underlying graph. We also show that cohesive lexical fields (taking the form of cliques of concepts) constitute essential bricks of meaning, and span the core homogeneously at the global level; 89% of all words in the core belong to at least one triangle, and 77% belong to cliques of triangles containing 4 words (i.e., pairs of triangles sharing an edge or forming tetrahedras). As the words of a graph of free associations acquire their meaning from the set of associations they are involved in (Deese, 1962), we go a step further by examining the neighborhood of nodes and extracting the statistics of cycles. We further check through human evaluation that the clustering is strongly related to meaning, and furthermore, the meaning becomes measurably more confused as one walks away from a cluster.

-¿ -¿I call the pairs of triangles sharing an edge the 2-clovers ;-)

Comparing dictionaries to free association, we find the free association graph being more concept driven, with words in small clusters being on the same level of abstraction. Moreover, we think that graphs of free associations could find interesting applications for *Word Sense Disambiguation* (*e.g.*, (Heylighen, 2001; Agirre and Soroa, 2009)), and could be used for detecting psychological disorders (*e.g.*, depression, psychopathy) or whether someone is lying (Hancock et al., 2013; Kent and Rosanoff, 1910).

Finally, we believe that studying the dynamics of graphs of free associations may be of particular interest for observing the change in meaning of certain words (Deese, 1967), or more generally to follow the cultural evolution arising among a social group.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 33–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark H Ashcraft and Gabriel A Radvansky. 2009. *Cognition*. Pearson Prentice Hall.

Chris Biemann. 2006. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 73–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Béla Bollobás. 2001. *Random graphs*, volume 73 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, second edition.

Etienne Brunet. 1974. Le traitement des faits linguistiques et stylistiques sur ordinateur. *Texte d'application: Giraudoux, Statistique et Linguistique. David, J. y Martin, R.(eds.). Paris: Klincksieck*, pages 105–137.

Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

James Deese. 1962. On the structure of associative meaning. *Psychological review*, 69:161.

James Deese. 1967. Meaning and change of meaning. *The American psychologist*, 22(8):641.

Beate Dorow, Dominic Widdows, Katerina Ling, Jean-Pierre Eckmann, Danilo Sergi, and Elisha Moses.

2005. Using curvature and Markov clustering in graphs for lexical acquisition and word sense discrimination. In *MEANING-2005, 2nd Workshop organized by the MEANING Project, February 3rd-4th 2005, Trento, Italy.*

Jean-Pierre Eckmann and Elisha Moses. 2002. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proc. Natl. Acad. Sci. USA*, 99(9):5825–5829 (electronic).

Brian Everitt, Sabine Landau, and Morven Leese. 2001. Cluster analysis. 4th Edition. *Arnold, London*.

Santo Fortunato. 2010. Community Detection in Graphs. *Physics Reports*, 486(3):75–174.

Pietro Gravino, Vito DP Servedio, Alain Barrat, and Vittorio Loreto. 2012. Complex structures and semantics in free word association. *Advances in Complex Systems*, 15(03n04).

Jeffrey T Hancock, Michael T Woodworth, and Stephen Porter. 2013. Hungry like the wolf: A word-pattern analysis of the language of psychopaths. *Legal and Criminological Psychology*, 18(1):102–114.

Ahmed Hassan and Dragomir Radev. 2010. Identifying text polarity using random walks. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 395–403. Association for Computational Linguistics.

Francis Heylighen. 2001. Mining associative meanings from the web: from word disambiguation to the global brain. In *Proceedings of Trends in Special Language & Language Technology*, pages 15–44.

Donald B Johnson. 1975. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84.

Istvan Jonyer, Diane J Cook, and Lawrence B Holder. 2002. Graph-based hierarchical conceptual clustering. *The Journal of Machine Learning Research*, 2:19–43.

Grace H Kent and Aaron J Rosanoff. 1910. *A study of association in insanity*. American Journal of Insanity.

George R Kiss, Christine Armstrong, Robert Milroy, and James Piper. 1973. An associative thesaurus of english and its computer analysis. *The computer and literary studies*, pages 153–165.

Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: A comparative analysis. *Physical review E*, 80(5):056117.

David Levary, Jean-Pierre Eckmann, Elisha Moses, and Tsvi Tlusty. 2012. Loops and self-reference in the construction of dictionaries. *Phys. Rev. X*, 2:031018.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge.

Yutaka Matsuo, Takeshi Sakaki, Kôki Uchiyama, and Mitsuru Ishizuka. 2006. Graph-based word clustering using a web search engine. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 542–550, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ken McRae and Kazunaga Matsuki. 2009. People use their knowledge of common events to understand language, and do so as quickly as possible. *Language and linguistics compass*, 3(6):1417–1429.

George Miller and Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical database*. MIT Press, Cambridge, MA.

George A Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

David S Palermo and James J Jenkins. 1964. Word association norms: Grade school through college. *University of Minnesota Press*.

Olivier Picard, Alexandre Blondin-Massé, Stevan Harnad, Odile Marcotte, Guillaume Chicoisne, and Yassine Gargouri. 2009. Hierarchies in dictionary definition space. In *Annual Conference on Neural Information Processing Systems*.

Pascal Pons and Matthieu Latapy. 2006. Computing communities in large networks using random walks. In *Journal of Graph Algorithms and Applications*, pages 284–293. Springer.

Christian Scheible. 2010. Sentiment translation through lexicon induction. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 25–30, Uppsala, Sweden, July. Association for Computational Linguistics.

David R Shanks. 1995. *The psychology of associative learning*, volume 13. Cambridge University Press.

Mark Steyvers and Joshua B Tenenbaum. 2005. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1):41–78.

Mark Steyvers, Richard M Shiffrin, and Douglas L Nelson. 2004. Word association spaces for predicting semantic similarity effects in episodic memory. *Experimental cognitive psychology and its applications: Festschrift in honor of Lyle Bourne, Walter Kintsch, and Thomas Landauer*, pages 237–249.

Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.

Unsupervised Induction of Cross-lingual Semantic Relations

Mike Lewis
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
mike.lewis@ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

Creating a language-independent meaning representation would benefit many cross-lingual NLP tasks. We introduce the first unsupervised approach to this problem, learning clusters of semantically equivalent English and French relations between referring expressions, based on their named-entity arguments in large monolingual corpora. The clusters can be used as language-independent semantic relations, by mapping clustered expressions in different languages onto the same relation. Our approach needs no parallel text for training, but outperforms a baseline that uses machine translation on a cross-lingual question answering task. We also show how to use the semantics to improve the accuracy of machine translation, by using it in a simple reranker.

1 Introduction

Identifying a language-independent semantics is a major long term goal of computational linguistics, and is interesting both theoretically and for practical applications. It assumes that semantically equivalent sentences in any language can be mapped onto a common meaning representation. Such a representation would be of great utility for tasks such as translation, relation extraction, summarization, question answering, and information retrieval. Regardless of whether it is even possible to create such a semantics, we show that an incomplete version can be useful for downstream tasks.

Semantic machine translation aims to map a source language to a language-independent meaning representation, and then generate the target language translation from this. It is hoped this would alleviate the difficulties of simpler models when translating between languages with very different word ordering and syntax (Vauquois, 1968). Despite many attempts to define interlingual representations (Mitamura et al., 1991; Beale et al., 1995; Banarescu et al., 2013), state-of-the-art machine translation still uses phrase-based models (Koehn et al., 2007). The major obstacle to defining interlinguas has been devising a meaning representation that is language-independent, but capable of expressing the limitless number of meanings that natural languages can express (Dorr et al., 2004).

Our approach avoids this problem by utilizing the methods of distributional semantics. Recent work has shown that paraphrases of expressions can be learned by clustering those with similar arguments (Poon and Domingos, 2009; Yao et al., 2011; Lewis and Steedman, 2013)—for example learning that *X wrote Y* and *X is the author of Y* are equivalent if they appear in a corpus with similar *(X, Y)* argument-pairs such as {*(Shakespeare, Macbeth), (Dickens, Oliver Twist)*}. We extend this to the multilingual case, aiming to also map the French equivalents *X a écrit Y* and *Y est un roman de X* on to the same cluster as the English paraphrases. Conceptually, we treat a foreign expression as a paraphrase of an English expression. The cluster identifier can be used as a predicate in a logical form, suggesting that the fundamental predicates of an interlingua can be learnt in an unsupervised manner via clustering.

In this paper we focus on learning binary relations between named entities. This problem is much simpler than attempting complete interlingual semantic

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 681–692,
Seattle, Washington, USA, 18-21 October 2013.

interpretation, but the approach could be generalized. This class of expressions has proved extremely useful in the monolingual case, with direct applications for question answering and relation extraction (Poon and Domingos, 2009; Mintz et al., 2009), and we demonstrate how to use them to improve machine translation. It is important to be able to extract knowledge across languages, as many facts will not be expressed in all languages—either due to less-complete encyclopedias being available in some languages, or facts being most relevant to a single country.

In contrast to most previous work on machine translation and cross-lingual clustering, our method requires no parallel text (see Section 8 for discussion of some exceptions). It instead exploits an alignment between named-entities in different languages. The limited size of parallel corpora is a significant bottleneck for machine translation (Resnik and Smith, 2003), whereas our approach can be used on much larger monolingual corpora. This means it is potentially useful for language-pairs where little parallel text is available, for domain adaptation, or for semi-supervised approaches.

2 Basic Approach

Our work builds on clustering-based approaches to monolingual distributional semantics, aiming to create clusters of semantically equivalent predicates, based on their arguments in a corpus. In each language, we first map each sentence in a large monolingual corpus onto a simple logical form, by extracting binary predicates between named entities. Then, we cluster predicates both within and between languages into those with similar arguments.

When parsing a new sentence, instead of using the monolingual predicate, we use the cluster identifier as a language-independent semantic relation, as shown in Figure 1. The resulting logical form can be used for inference in question answering.

Unlike traditional approaches to translation, this does not require parallel text—but it does impose some additional constraints on language resources. Our approach requires:

- A large amount of factual text, as we rely on the same facts being expressed in different languages. We use Wikipedia, which contains articles in 250 languages, including 121 with at least 10,000 articles.[1] Other domains, such as Newswire, may also be effective.
- A method for extracting binary relations from sentences. This is straightforward from dependency parses, which are available for many languages. It is also possible without a parser, with some language-specific work (Fader et al., 2011). We describe our approach in Section 3.
- A method for linking entities in the training data to some canonical representation. McNamee et al. (2011) report good results on this task in 21 languages. We describe our method for this in Section 4.1.

3 Predicate Extraction

Our method relies on extracting binary predicates between entities from sentences. Various representations have been suggested for binary predicates, such as Reverb patterns (Fader et al., 2011), dependency paths (Lin and Pantel, 2001; Yao et al., 2011), and binarized predicate-argument relations derived from a CCG-parse (Lewis and Steedman, 2013). Our approach is formalism-independent, and is compatible with any method of expressing binary predicates.

We choose the CCG-based parser of Lewis and Steedman (2013) for several reasons. It outputs a logical form derived automatically from the CCG-parse, containing predicates such as: $write_{arg0,arg1}(shakespeare,macbeth)$. By using the close relationship between the CCG syntax and semantics, it is able to generalize over many semantically equivalent syntactic constructions (such as passives, conjunctions and relative clauses), meaning we can map both *Shakespeare wrote Macbeth* and *Macbeth was written by Shakespeare* to the same logical form. Using a dependency-based representation, these would have different predicates, which would need to be clustered later. CCG also has a well developed theory of operator semantics (Steedman, 2012), so is able to represent semantic operators such as quantifiers, negation and tense—understanding these is crucial to high performance on question answering or translation tasks. As in

[1] As of June 2013.

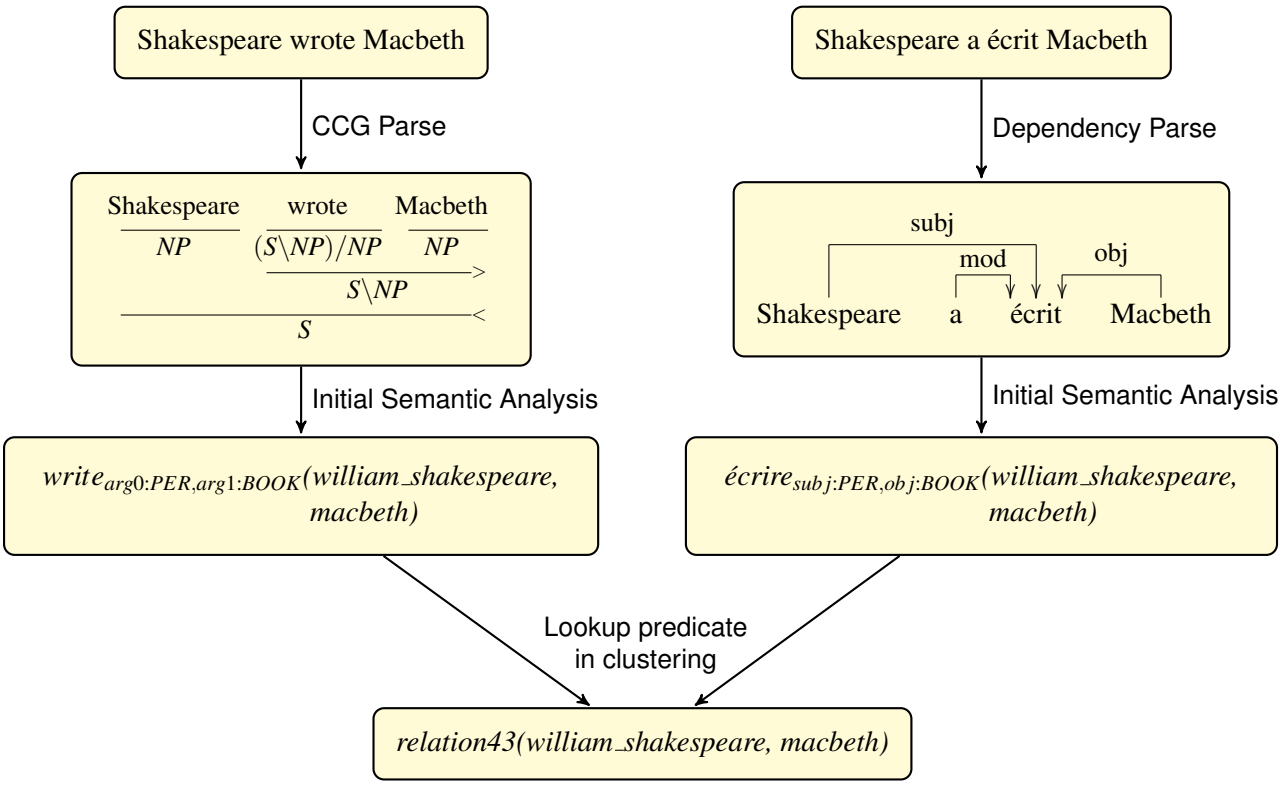


Figure 1: Example showing how our system can map sentences in different languages to the same meaning representation, assuming we have clustered the equivalent predicates $write_{arg0:PER,arg1:BOOK}$ and $écrire_{subj:PER,obj:BOOK}$

Lewis and Steedman (2013), clusters derived from the output from the parser can be integrated into the lexicon, allowing us to build logical forms which capture both operator and lexical semantics.

Accurate CCG syntactic parsers are currently only available for English, whereas dependency treebanks and parsers exist for many languages (Buchholz and Marsi, 2006). Consequently, for French we use the dependency path representation, which captures the nodes and edges connecting two named entities in a dependency parse. The extraction of these paths is language-independent, and does not depend on the dependency grammar used, which means our approach could be adapted to new languages with minimal work.

4 Entity Semantics

4.1 Entity Linking

As discussed, our approach assumes that semantically similar predicates will have similar argument entities. This requires us to be able to identify coreferring entities across languages during training. In the monolingual case, it suffices to represent entities by the string used in the sentence. This is inadequate in the multilingual case, as many entities may be referred to by different names in different languages—for example the *United States* translates as *les États-Unis* in French and *die Vereinigte Staaten* in German. This problem is worsened by the ambiguity of named-entity strings—for example, in the context of a sports article, *United States* may refer specifically to a team, rather than a country.

Recent work on multilingual named-entity linking (McNamee et al., 2011) shows how to link named entities in multiple languages onto English Wikipedia articles, which can be used as unique identifiers for entities. This means that we could gain the information we need from unrestricted text. However, as we use Wikipedia itself for our training corpora, we can bootstrap entity information directly from its markup. Wikipedia contains cross-language links, e.g. between the United States articles in different languages, allowing us to determine the equivalence of entities in differ-

ent languages. Wikipedia links also help us automatically disambiguate entities to a given article. For unlinked named-entity mentions, we perform some simple heuristic co-reference—based on word-overlap with previously mentioned entities in the document, whether the mention name is the title of a Wikipedia article, or whether the mention name is a Freebase (Bollacker et al., 2008) alias of an entity. We emphasise that this does not mean our approach is only applicable to the Wikipedia corpus.

4.2 Entity Typing

It has become standard in clustering approaches to distributional semantics to assign types to predicates before clustering, and only cluster predicates with the same type (Schoenmackers et al., 2010; Berant et al., 2011; Yao et al., 2012). This is useful for resolving ambiguity—for example the phrase *born in* may express a place-of-birth or date-of-birth relation depending on whether its second argument has a *LOC* or *DAT* type. Ambiguous expressions may translate differently in other languages—for example, the two interpretations of *was born in* translate in French as *est né à* and *est né en* respectively. The type of a predicate is determined by the type of its arguments, and predicates with different types are treated as distinct.

Lewis and Steedman (2013) induce an unsupervised model of entity types using Latent Dirichlet Allocation (Blei et al., 2003), based on selectional preferences of verbs and argument-taking nouns. When applied cross-linguistically, we found this technique tended to create language-specific topics. Instead, we exploit the fact that many Wikipedia entities are linked to the Freebase database, which has a detailed manually-built type-schema. This means for a Wikipedia entity, we can look up its set of types in Freebase.[2] We use the simplified type-set of 112 types created by Ling and Weld (2012). Where entities have multiple types (for example, Shakespeare is both an *author* and a *person*), we create a separate relation for each type.

[2]Named entities not present in Freebase are ignored during training.

5 Relation Clustering

Predicates are clustered into those which are semantically equivalent, based on their argument-pairs in a corpus. The initial semantic analysis is run over the corpora, and for each predicate we build a vector containing counts for each of its argument-pairs (we divide these counts by the overall frequency of an argument-pair in the corpus, so that rarer argument-pairs are more significant). These vectors are used to compute similarity between predicates.

First, we run the clustering algorithm on each language independently, and then we attempt to find an alignment between the clusters. Duc et al. (2011) and Täckström et al. (2012) use similar two-step approaches. Running the clustering on both languages simultaneously was found to produce many clusters only containing predicates from a single language. This appears to be because even if predicates in two different languages are truth-conditionally equivalent, the language biases the sample of entity-pairs found in a corpus. For example, the French verb *écrire* may contain more French author/book pairs than the English equivalent *write*. This difference can make the verbs appear to represent different predicates to the clustering algorithm. Our two-step approach also means that advances in monolingual clustering should directly lead to improved cross-lingual clusters.

5.1 Monolingual Clustering

Following Lewis and Steedman (2013), we use the Chinese Whispers algorithm (Biemann, 2006) for monolingual clustering—summarized in Algorithm 1. The algorithm is non-parametric, meaning that the number of relation clusters is induced from the data, and highly scalable. We create a separate graph for each type of predicate in each language—for example, predicates between types *AUTHOR* and *BOOK* in French (so only predicates with the same type will be clustered). We create one node per predicate in the graph, and edges represent the distributional similarity between the predicates.

The distributional similarity between a pair of predicates is calculated as the cosine-similarity of their argument pair vectors in the corpus. Many more sophisticated approaches to determining similarity have been proposed (Kotlerman et al., 2010;

Weisman et al., 2012), and future work should explore these. We prune nodes with less than 25 occurrences, edges of weight less than 0.05, and a short list of stop predicates. We find many of our French dependency paths do not have a clear semantic interpretation, so add the requirement that dependency paths contain at least one content word, contain at most 5 edges, and that one of the dependencies connected to the root is subject, object or the French preposition *de*.

Data: Set of predicates P
Result: A cluster assignment r_p for all $p \in P$
$\forall p \in P : r_p \longleftarrow$ unique cluster identifier;
while *not converged* **do**
  randomize order of P
  for $p \in P$ **do**
    $r_p \longleftarrow \arg\max_r \sum_{p'} \mathbb{1}_{r=r_{p'}} sim(p, p')$
  end
end

Algorithm 1: Chinese Whispers algorithm, used for monolingual predicate clustering. $sim(p, p')$ is the distributional similarity between p and p', and $\mathbb{1}_{r=r'}$ is 1 iff r=r' and 0 otherwise

5.2 Cross-lingual Cluster Alignment

We use a simple greedy procedure to find an alignment between the monolingual clusters in different languages. First, the entity-pair vectors for each predicate in a relation cluster are merged. Then, the cosine similarity between entity-pair vectors for clusters in different languages is calculated—we base this only on argument-pairs that occur in both languages, to reduce the potential bias of some entities being more relevant to one language. Clusters are then greedily aligned, in order of their similarity, as in Algorithm 2 (pruning similarities less than 0.01). This means that clusters are aligned with their most similar foreign cluster. We only attempt to align clusters with the same argument types.

6 Cross Lingual Question Answering Experiments

We evaluate our system on English and French, using Wikipedia for corpora. The English corpus is POS-tagged and CCG-parsed with the C&C tools

Data: Sets of monolingual relation clusters R_{L1} and R_{L2}
Result: An alignment between the monolingual clusters A
$A \longleftarrow \{\}$;
while $R_{L1} \neq \{\} \wedge R_{L2} \neq \{\}$ **do**
  $(r1, r2) \longleftarrow \arg\max_{(r_1, r_2) \in R_{L1} \times R_{L2}} sim(r1, r2)$;
  $A \longleftarrow A \cup \{(r1, r2)\}$;
  $R_{L1} \longleftarrow R_{L1} / \{r1\}$;
  $R_{L2} \longleftarrow R_{L2} / \{r2\}$;
end

Algorithm 2: Cluster alignment algorithm

English	French
X invades Y	X envahit Y invasion de Y par X
X orbits Y	X est un satellite de Y X est une lune de Y
X is a skyscraper in Y	X est un gratte-ciel de Y
X is a novel by Y	X est un roman de Y
X joins Y X is a member of Y	X adhère à Y X entre dans Y X rejoint Y

Table 1: Some example cross-lingual clusters. Predicates are given in a human-readable form, and predicate types are suppressed.

(Clark and Curran, 2004). The French corpus is tagged with MElt (Denis et al., 2009) and parsed with MaltParser (Nivre et al., 2007), trained on the French Treebank (Candito et al., 2010). Wikipedia markup is filtered using Wikiprep (Gabrilovich and Markovitch, 2007)—replacing internal links with the name of their target article, to help entity linking. Some example clusters learnt by our model are shown in Table 1. We find that the cross-lingual clusters typically contain more French expressions than English, possibly due to the differing sizes of the corpora—adjusting the parameters in Section 5 results in larger clusters, but introduces noise.

6.1 Experimental Setup

We evaluate our system on a cross-lingual question answering task, similar to monolingual QA evaluations by Poon and Domingos (2009) and Lewis and

Steedman (2013). A question is asked in language *L*, and is answered by the system from a corpus of language *L'*. Human annotators are shown the question, answer entity, and the sentence that provided the answer, and are then asked whether the answer is a reasonable conclusion based on the sentence. Whilst this task is much easier than full translation, it is both a practical application for our approach, and a reasonably direct extrinsic evaluation for our cross-lingual clusters.

Following Poon and Domingos (2009) and Lewis and Steedman (2013), the question dataset is automatically generated from the corpus. This approach has the advantage of evaluating on expressions in proportion to their corpus frequency, so understanding frequent expressions is more important than rare ones. We then sample 1000 questions for each language, by extracting binary relations matching certain patterns ($X \overset{nsubj}{\leftarrow} verb \overset{dobj}{\rightarrow} Y$, $X \overset{nsubj}{\leftarrow} verb \overset{pobj}{\rightarrow} Y$ or $X \overset{nsubj}{\leftarrow} be \overset{dobj}{\rightarrow} noun \overset{pobj}{\rightarrow} Y$), and removing one of the arguments. For example, from the sentence *Obama lives in Washington* we create the questions *X lives in Washington?*, and *Obama lives in X?*.[3] Answers are judged by fluent bilingual humans, and do not have to match the entity that originally instantiated *X*. Multiple answers can be returned for the same question.

Our system attempts this task by mapping both the question and candidate answer sentences (which will be in a different language to the question) on to a logical form using the clusters, and determining whether they express the same relation. This tests the ability of our approach to cluster expressions into those which are semantically equivalent between languages. It is possible for entities to have multiple types (see Section 4.2), and answers are ranked by the number of types in which the entailment relation is predicted to hold.

[3]Questions are given in a declarative form, to make the tasks simpler for the machine translation baseline. We found the machine translation performed poorly on questions such as *What is Obama the president of?*, as inverted word-orders and long-range dependencies are difficult to handle with re-ordering models and language models (though are straightforward to handle for a CCG system (Clark et al., 2004)). We find that machine translation performs much better on declarative equivalents, such as: *Obama is the president of X*.

6.2 Baseline

Our baseline makes use of the Moses machine translation system (Koehn et al., 2007), and is similar to previous approaches to cross-lingual question answering such as Ahn et al. (2004). We train a Moses model on the Europarl corpus (Koehn, 2005). First, the question is translated from language *L* to *L'*, taking the 50-best translations. As the questions are typically shorter than corpus sentences, this is substantially easier for the machine-translation than translating the corpus. These are then parsed, and patterns are extracted (as in Section 3). We also manually supply a translation of the named-entity in the question (based on the Freebase entity name translation), to avoid penalizing the translation system for failing to translate named-entities that have not been seen in its training data. These patterns are then used to find answers to the questions. Answers are ranked by the score of the best translation that produced the pattern. Figure 2 illustrates this pipeline.

The choice of languages is very favourable to the machine-translation system, English and French have similar word-order, and there is a large amount of parallel text available (Koehn and Monz, 2006). Our system works with any word-order, and does not require parallel text for training, so we would expect better performance relative to machine-translation on other language pairs. Future work will experiment with more diverse languages. The sentences to be translated are also very short, reducing the potential for error.

6.3 Results

Results are shown in Table 3, based on a sample of 100 answers from the output of each of the systems. Unsurprisingly, the machine-translation has high accuracy on this task, given the choice of languages and the short queries. Pleasingly, our clusters achieve similar accuracy, with much greater recall, with no usage of parallel text.

Examining the results, we see that the distribution of answers is highly skewed for all systems, with many answers to a smaller number of questions (multiple answers can be returned to the same question). This is due to the Zipfian nature of language, the difficulty of the task (which is far from

Question	Answer
X dies in Moscow	**Sergueï Guerassimov** meurt d'une crise cardiaque le mardi 26 novembre 1985 à Moscou
Germany invades X	... depuis l'invasion de la **Pologne** par l'Allemagne et l'URSS
X wins the FA Cup	**Portsmouth FC** remporte la FA Challenge Cup en s'imposant en finale face à Wolverhampton Wanderers FC
X is a band from Finland	**Yearning** est un groupe Finlande de doom metal atmosphérique
X vit en France	**Dewi Sukarno** ... has lived in different countries including Switzerland, France and the United States
X bat Kurt Angle	**Anderson** defeated Kurt Angle and Abyss to advance to the finals
X est une ville de Kirghizistan	**Il'chibay** is a village in the Issyk Kul Province of Kyrgyzstan

Table 2: Example questions correctly answered using our clusters, with the answer entity highlighted in bold.

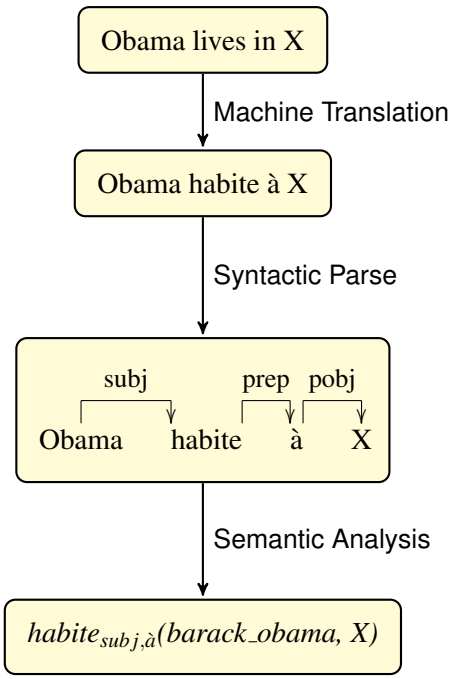


Figure 2: Pipeline used by baseline system for answering French questions. The pattern extracted from the translated sentence is used to search for answers in an English corpus.

English→ French	Answers	Correct
Baseline	269	86%
Clusters (best 270)	270	100%
Clusters (all)	1032	72%
French→ English	**Answers**	**Correct**
Baseline	274	85%
Clusters (all)	401	93%

Table 3: Results on wide-coverage Question Answering task. Best-N results are shown to illustrate the accuracy of our cluster-based system at the same rank as the baseline. It is not possible to give a recall figure, as the total number of correct answers in the corpus is unknown. English→ French results are from the full French Wikipedia corpus, whereas French→ English results are from a 10% sample.

solved in the monolingual case), and the possibility that questions may have no answers in the foreign corpus. This is particuarly true for the clustering approach—although the clustering system finds more answers with the English corpus, the baseline system answers slightly more unique questions (57 vs 66). The 1032 answers found by the clusters in the French corpus came from just 56 questions (compared to 29 unique questions answered by the baseline). This suggests that the translations found by the clustering can be more useful than those of Moses on this task—for example, it may find an equivalence between a rare French term and a common related English term, where machine translation may only find a more literal translation.

Despite this, we see the clusters have learnt to

paraphrase a variety of relations between languages with high accuracy, suggesting that there is much potential for the use of unsupervised clusters in cross-lingual semantic applications. Some examples answers are given in Table 2. Most of the errors are caused by a small number of questions.

7 Translation Reranking Experiments

Ultimately, we would like to be able to translate using semantic parsing with cross-lingual clusters. As a step towards this, we investigated whether we could rerank the output of a machine translation system, on the basis of whether the semantic parse of the source sentence is consistent with that of candidate translations.

We sample French sentences where we can produce a semantic parse (i.e. we can extract a predicate between named entities that maps to a cross-lingual cluster). These sentences are translated to English using Moses, taking the 50-best list, and semantic parses are produced for each of these. If the semantic parse for the 1-best translation does not match the source semantic parse, we take the parse from the 50-best list that most closely matches it—otherwise we discard the sentence from our evaluation, as our semantics agrees with the machine-translation.

To ensure that the evaluation focuses on the clusters, we try to exclude several other factors that might affect the results. The coverage of our CCG parsing and semantic analysis drops significantly on noisy translated sentences, and potentially acts as a language model by failing to produce any semantic parse on ungrammatical output sentences. We therefore only consider sentences where we can produce a semantic parse for the 1-best machine translation output. We also try to avoid penalizing the machine-translation system for failing to translate named entities correctly, so we do not attempt to rerank sentences where the entities from the source sentence are not present in the 1-best translation.

Human annotators were shown the source sentence, the 1-best translation, and the translation chosen by the reranker (the translations were shown in a random order). To focus the evaluation on the semantic relations we are modelling, we ask the annotators which sentence best preserves the meaning between the named entities that have different relations in the semantic parse. This avoids our system being penalized for choosing a translation that is worse in aspects other than the relations it is modelling. An example is shown in Table 4. The data was annotated jointly by two fluent bilingual speakers, who reported high agreement on this task.

	Percentage of translations preferred
1-best Moses translation	5%
Cluster-based Reranker	39%
No preference	56%

Table 5: Human preference judgements for the translation reranking experiment, based on a sample of 87 sentences. Results show the percentage of sentences for which the annotators preferred the original translation, the reranked translation, or neither. As discussed in the text, results where annotators had no preference were typically due to syntactic parse errors.

Results are shown in Table 5, and are highly encouraging, with the original Moses output being preferred to the reranked translation in only 5% of cases where our model makes a positive prediction.

Inspecting the results, we see that many of the cases where the annotators had no preference were caused by syntactic parse errors. For example, if the 1-best translation is correct, but a prepositional phrase is incorrectly attached, it will appear to have an incorrect semantics. A similar translation in the 50-best list may be correctly parsed, and consequently selected by our reranker. However, a human will have no preference between these translations. Incorporating K-Best parsing into our pipeline may help mitigate against such cases.

This preliminary experiment suggests that there is potential for future improvements in machine translation using cross-lingual distributional semantics. The system only attempts to rerank a very small proportion of sentences, but we believe the coverage could be greatly improved by including relations between common nouns (rather than just named-entities)—future work should explore this.

8 Related Work

Our work builds on recent progress in monolingual distributional semantics (Poon and Domingos, 2009; Yao et al., 2011; Lewis and Steedman, 2013) by

Source	Le Princess Elizabeth arrive à Dunkerque le 3 août 1999
Machine translation 1-best	Le Princess Elizabeth is to manage to Dunkirk on 3 August 1999
Reranked translation	The Princess Elizabeth arrives at Dunkirk on 3 August 1999

Table 4: Example sentence that is reranked by our clusters. Human evaluators were asked which translation best preserved the meaning between *Princess Elizabeth* and *Dunkirk*.

clustering typed predicates into those which are semantically equivalent. We also show how to bootstrap semantic information about entities from the Wikipedia markup, and believe this makes it an interesting corpus for future work on monolingual distributional semantics.

Cross-language Latent Relational Analysis (Duc et al., 2011) is perhaps the most similar previous work to ours, which moves the work of Turney (2005) into a multilingual setting. Duc et al. (2011) aim to compute, for example, that the 'latent relation' between *(Obama, US)* in an English corpus is similar to that between *(Cameron, UK)* in a foreign corpus. This is solved by finding all textual patterns between the two entity-pairs, and computing their overall similarity. Like us, they compute similarity between expressions in different languages based on named-entity arguments and clustering (unlike us, they also rely on machine translation for computing similarity). A key difference is that their system aims to understand the overall relation between an entity-pair based on many observations, whereas our approach attempts to understand each sentence individually (as is required for tasks such as translation).

Various recent papers have explored the relationship between translation and monolingual paraphrases —for example Bannard and Callison-Burch (2005) create paraphrases by pivoting through a foreign translation, and Callison-Burch et al. (2006) show that including monolingual paraphrases improves the quality of translation by reducing sparsity. The success of these approaches depends on the many-to-many relationship between equivalent expressions in different languages. Our approach aims to model this relationship explicitly by clustering all equivalent paraphrases in different languages.

Current state-of-the-art machine translation systems circumvent the problem of full semantic interpretation, by using phrase-based models learnt from large parallel corpora (Brown et al., 1993). Although this approach has been very successful, it has significant limitations—for example, when translating between languages with very different word-orders (Birch et al., 2009), or with little parallel text.

Semantic machine translation aims to map the source language to an interlingual semantic representation, and then generate the target language sentence from this. Jones et al. (2012) show how this can be done on a small dataset using hyperedge replacement grammars. A major obstacle to this is designing a suitable meaning representation, which involves choosing a set of primitive concepts which are abstract enough to be capable of expressing meaning in any language (Dorr et al., 2004). A recent proposal for this is the Abstract Meaning Representation (Banarescu et al., 2013), which uses English verbs as a set of predicates. This is a less abstract form of semantic interpretation than our proposal, as semantically equivalent paraphrases may be given a different representation. Such an approach also relies on annotating large amounts of text with the semantic representation—whereas our unsupervised approach offers a way to build such an interlingua using only a method for extracting predicates from sentences.

Whilst almost all recent work on machine-translation has relied on parallel text, there have been several interesting approaches that do not. Rapp (1999) learns to translate words based on small seed bilingual dictionary. Klementiev et al. (2012a) exploit a variety of interesting indirect sources of information to learn a lexicon—for example assuming that equivalent Wikipedia articles in different languages will use semantically similar words. The Polylingual Topic Model (Mimno et al., 2009) makes use of similar intuitions. Whilst we exploit equivalent Wikipedia articles for entity linking, we do not require aligned articles. Incorporating such techniques into our model would be a natural next

step, allowing us to learn a more complete lexicon. To our knowledge, ours is the first approach to learn to translate semantic relations, rather than words and phrases.

Several other recent papers have learnt cross-lingual word clusters, and used these to improve cross-lingual tasks such as document-classification (Klementiev et al., 2012b), parsing (Täckström et al., 2012) and semantic role labelling (Kozhevnikov and Titov, 2013) in resource-poor languages. Cross-lingual word clusters are learnt by aligning monolingual clusters on the basis of parallel text—in language-pairs where parallel text is available, this offers an interesting complement to our method of clustering based on named entities.

9 Conclusions and Future Work

We have demonstated that our previous work on monolingual distributional semantics can simply be extended to learn a language-independent semantics of relations from unlabelled text, and that this semantics is powerful enough to aid applications such as question answering and translation reranking.

There is much potential for future extensions to address the limitations of the process described here. As we use a flat clustering of relations, we are only able to model synonyms and not hypernyms. More sophisticated clustering techniques, such as those used by Berant et al. (2011), seem to offer a way to address this. Our system clusters relations with similar named-entity arguments, but this means it does not cluster relations whose arguments are rarely named entities. However, using cross-lingual clusters of common nouns, such as those from Täckström et al. (2012), it should be possible to cluster relations that take semantically similar common noun arguments. Embedding cluster-identifiers in a logical form allows us to also model logical operators, such as negation and quantifiers, which may help to improve the translation of these. It would also be interesting to experiment with more diverse languages types.

Acknowledgements

We thank the anonymous reviewers for their helpful comments, and Eva Hasler for help training Moses. This work was funded by ERC Advanced Fellowship 249520 GRAMPLUS and IP EC-FP7-270273 Xperience.

References

Kisuh Ahn, Beatrix Alex, Johan Bos, Tiphaine Dalmas, Jochen L Leidner, and Matthew B Smillie. 2004. Cross-lingual question answering with QED. In *Working Notes, CLEF Cross-Language Evaluation Forum*, pages 335–342.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Stephen Beale, Sergei Nirenburg, and Kavi Mahesh. 1995. Semantic analysis in the Mikrokosmos machine translation project. In *Proceedings of the 2nd Symposium on Natural Language Processing*, pages 297–307.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 610–619. Association for Computational Linguistics.

C. Biemann. 2006. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80. Association for Computational Linguistics.

Alexandra Birch, Phil Blunsom, and Miles Osborne. 2009. A Quantitative Analysis of Reordering Phenomena. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 197–205, Athens, Greece, March. Association for Computational Linguistics.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring hu-

man knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 17–24, New York City, USA, June. Association for Computational Linguistics.

Marie Candito, Benoît Crabbé, Pascal Denis, et al. 2010. Statistical french dependency parsing: treebank conversion and first results. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, pages 1840–1847.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04. Association for Computational Linguistics.

S. Clark, M. Steedman, and J.R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the EMNLP Conference*, pages 111–118.

Pascal Denis, Benoît Sagot, et al. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art pos tagging with less human effort. In *PACLIC*, pages 110–119.

Bonnie J Dorr, Eduard H Hovy, and Lori S Levin. 2004. Machine translation: Interlingual methods.

Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka. 2011. Cross-language latent relational search: Mapping knowledge across languages. *Association for the Advancement of Artificial Intelligence*, pages 1237–1242.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545. Association for Computational Linguistics.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. *Proc. COLING, 2012*.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012a. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 130–140. Association for Computational Linguistics.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012b. Inducing crosslingual distributed representations of words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, Bombay, India, December.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 102–121, New York City, June. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Nat. Lang. Eng.*, 16(4):359–389, October.

Mikhail Kozhevnikov and Ivan Titov. 2013. Crosslingual transfer of semantic role models. In *To Appear in Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2013. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text. In *In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas W Oard, and David Doermann. 2011. Cross-language entity linking. *Proc. IJCNLP2011*.

David Mimno, Hanna M Wallach, Jason Naradowsky, David A Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 880–889. Association for Computational Linguistics.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Teruko Mitamura, Eric H Nyberg, and Jaime G Carbonell. 1991. An efficient interlingua translation system for multi-lingual document production.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 1–10. Association for Computational Linguistics.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 519–526. Association for Computational Linguistics.

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

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1088–1098. Association for Computational Linguistics.

Mark Steedman. 2012. *Taking Scope: The Natural Semantics of Quantifiers*. MIT Press.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 477–487. Association for Computational Linguistics.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 1136–1141, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bernard Vauquois. 1968. A survey of formal grammars and algorithms for recognition and transformation in machine translation. In *IFIP Congress*, volume 68, pages 254–260.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. 2012. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 194–204. Association for Computational Linguistics.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1456–1466. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *ACL (1)*, pages 712–720.

Two-stage Method for Large-scale Acquisition of Contradiction Pattern Pairs using Entailment

Julien Kloetzer[*] **Stijn De Saeger**[†] **Kentaro Torisawa**[‡] **Chikara Hashimoto**[§]
Jong-Hoon Oh[¶] **Motoki Sano**[‖] **Kiyonori Ohtake**[**]
Information Analysis Laboratory,
National Institute of Information and Communications Technology (NICT), Kyoto, Japan
{[*]julien, [†]stijn, [‡]torisawa, [§]ch, [¶]rovellia, [‖]msano, [**]kiyonori.ohtake}@nict.go.jp

Abstract

In this paper we propose a two-stage method to acquire contradiction relations between typed lexico-syntactic patterns such as X_{drug} *prevents* $Y_{disease}$ and $Y_{disease}$ *caused by* X_{drug}. In the first stage, we train an SVM classifier to detect contradiction pattern pairs in a large web archive by exploiting the *excitation* polarity (Hashimoto et al., 2012) of the patterns. In the second stage, we enlarge the first stage classifier's training data with new contradiction pairs obtained by combining the output of the first stage's classifier and that of an *entailment* classifier. We acquired this way 750,000 typed Japanese contradiction pattern pairs with an estimated precision of 80%. We plan to release this resource to the NLP community.

1 Introduction

The ability to detect contradictory information in text has many practical applications. Among those, Murakami et al. (2009) pointed out that a contradiction recognition system can detect conflicts and anomalies in large bodies of texts and flag them to help users identify unreliable information. For example, many Japanese web pages claim that *agaricus prevents cancer*, where agaricus is a species of mushroom found in a variety of commercial products. Although this has been accepted by many Japanese people, by Googling keywords "agaricus", "promotes" and "cancer", we can find pages claiming that "agaricus promotes cancer", some of which point to a study authorized by the Japanese Ministry of Health, Labour and Welfare[1] reporting that a commercial product containing agaricus promoted cancer. Obviously, the existence of these pages casts serious doubt on the ability of agaricus to prevent cancer and encourages readers to dig more about this subject.

The above example suggests that recognizing contradictory information can guide users to a *true* fact. Likewise, we believe that contradiction recognition is also useful when dealing with non-factual information that occupy most of our daily lives. For instance, there is a big controversy recently whether Japan should join an economic partnership agreement called the *Trans Pacific Partnership (TPP)*, and quite serious but contradictory claims are plentiful in the mass media and on the web, e.g., *TPP will wipe out Japan's agricultural businesses* and *TPP will strengthen Japan's agricultural businesses*. Neither of these are facts; they are *predictions* that can only be realized or disputed after the underlying decision-making is done: joining or refusing the TPP.

Furthermore, after reading documents including contradictory predictions, one should notice that each of them is supported by a *convincing* theory that has no obvious defect, e.g., "Exports of Japan's agricultural products will increase thanks to the TPP" or "A large amount of low-price agricultural products will be imported to Japan due to the TPP". Even if one of these predictions may just happen to be true because of unexpected reasons such as minor fluctuations in the Japanese yen, we must survey such theories that support contradictory predictions, conduct balanced decision-making, and prepare counter measures for the expected problems after examining multiple viewpoints. Contradiction recognition should be useful to select documents to be surveyed.

[1] http://www.mhlw.go.jp/topics/bukyoku/iyaku/syoku-anzen/qa/060213-1.html

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 693–703,
Seattle, Washington, USA, 18-21 October 2013.

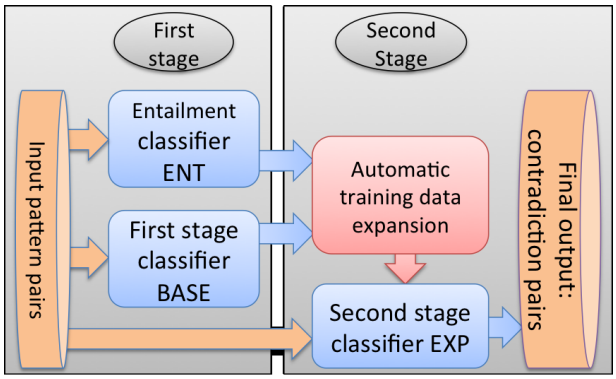


Figure 1: Method workflow

We have developed a method for recognizing pairs of contradictory binary patterns such as ⟨"*X promotes Y*", "*X prevents Y*"⟩ and ⟨"*X will wipe out Y*", "*X will strengthen Y*"⟩. To solve the problem described above, we can easily develop a system that can find contradictory text fragments from the web like "*agaricus promotes cancer*" and "*agaricus prevents cancer*" from the discovered contradictory pattern pairs.

Our method is a two-stage procedure with three supervised classifiers (Fig. 1). In the first stage, we build a classifier **BASE** to recognize contradictions between binary patterns, and a classifier **ENT** to recognize entailment. In the second stage, we combine the contradiction pairs recognized by **BASE** and the entailment pairs recognized by **ENT** to expand **BASE**'s training data and train a new contradiction classifier, **EXP**. This expansion using entailment is one key idea of this work: we acquired 750,000 contradiction pairs with 80% precision using the expanded training data, more than doubling the 285,000 pairs acquired at the same precision level without expansion. We also demonstrate that this result is not trivial by showing that our method outperforms an alternative one based on *Integer Linear Programming* inspired by the successful entailment recognition method of Berant et al. (2011).

As another technical contribution of this work, we exploit the recently proposed semantic polarity of *excitation* (Hashimoto et al., 2012) to recognize contradictions between binary patterns. Hashimoto et al. (2012) previously showed that excitation polarities are useful to recognize contradictions between phrases that consist of a noun and a predicate, such as "promote cancer" and "prevent cancer". While it is trivial to extend this framework to contradictions between *unary* patterns such as "promote X" and "prevent X" by replacing the common nouns in each pair with a variable, the information represented in unary patterns is often vague, and it is unlikely that a contradiction between unary patterns directly leads to the discovery of unreliable information to be flagged or to a meaningful survey of complex problems. As exemplified by the *agaricus* and *TPP* examples, contradictions between *binary* patterns that include two variables such as "X promotes Y" or "X will wipe out Y" are more useful than those between unary patterns. We also show that it is not trivial to recognize contradictions between binary patterns using contradictions between unary patterns.

Most works dealing with contradiction recognition up till now (Harabagiu et al., 2006; Bobrow et al., 2007; Kawahara et al., 2008; Kawahara et al., 2010; Ohki et al., 2011) focus on recognizing contradictions between full sentences or documents, not text fragments that match our relatively short patterns (survey in Section 5). We expect that the contradictory pattern pairs we acquired can be used as building blocks in such full-fledged contradiction recognition for full sentences or documents, similarly to antonym pairs in Harabagiu et al. (2006).

Also, we should emphasize that our method focuses on the most challenging part of contradiction recognition according to the classification of De Marneffe et al. (2008). Since we discard patterns with negations, an evident source of contradictions like ⟨"*X causes Y*", "*X does not cause Y*"⟩, most of our output are non-trivial contradictions related to high-level semantic phenomena, e.g., contradiction pairs related to antonyms like ⟨"X が Y を上げる", "X が Y を下げる"⟩ (⟨"*X increases Y*", "*X decreases Y*"⟩), lexical contradictions like ⟨"X が Y に勝つ", "Y が X に勝つ"⟩ (⟨"*X wins against Y*", "*Y wins against X*"⟩), or contradictions due to common-sense knowledge like ⟨"X が Y を安心させる", "X が Y を裏切る"⟩ (⟨"*X reassures Y*", "*X betrays Y*"⟩). We believe acquiring such contradictions in a large scale is a valuable contribution.

The following is the outline of this paper. Section 2 details our target and our proposed method. Evaluation results are discussed in Section 3. Sec-

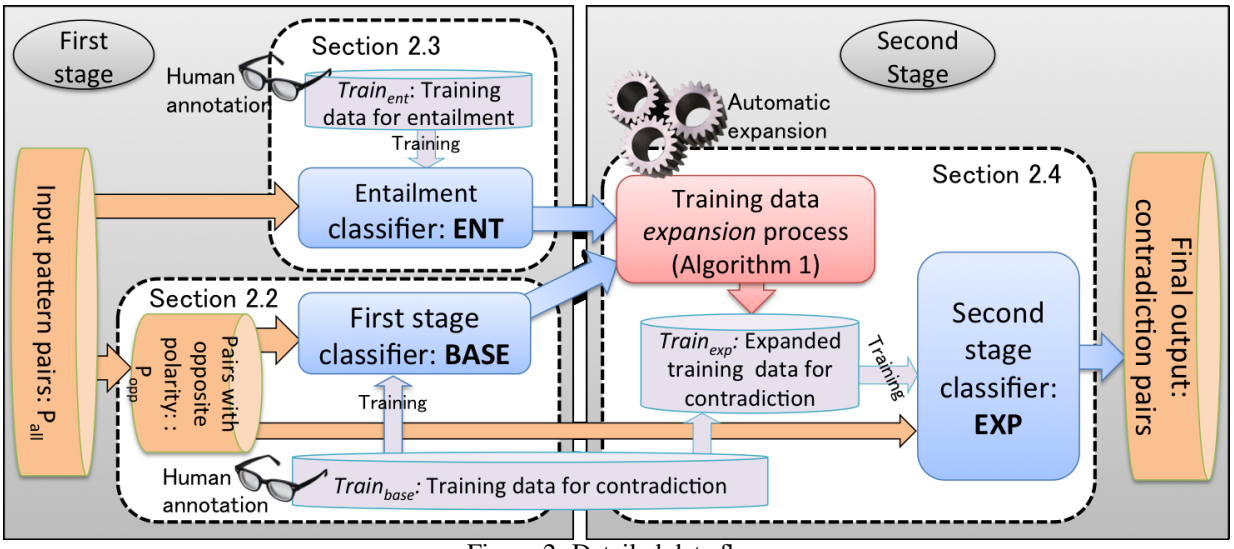


Figure 2: Detailed data flow

tion 4 details our features set, and Section 5 related work. Section 6 provides a conclusion.

2 Proposed method

As showed in Figure 1, our method consists of three supervised classifiers. Classifiers **BASE** and **EXP** recognize contradiction relations between binary patterns, and **ENT** recognizes entailment relations between binary patterns. The contradiction pairs recognized by **BASE** and the entailment pairs recognized by **ENT** are combined to generate new contradiction pairs, part of which are then added to **BASE** training data to train the **EXP** classifier. Our final output is the set of all binary pattern pairs regarded as contradictions by **EXP**. Since the dependencies between these three classifiers, their distinct sets of training data, and the two data sets to be classified (we describe those in the two sections below) is a bit complex, we show a complete description of the whole process in Figure 2.

The key idea is in the scheme that expands the training data. Logically speaking, patterns p and r are contradictory if there exists a pattern q such that p entails q and q contradicts r. For example, since "*X causes Y*" entails "*X promotes Y*" and "*X promotes Y*" contradicts "*X prevents Y*", then "*X causes Y*" contradicts "*X prevents Y*". Hence, by combining entailment and contradiction pairs, we can obtain more contradiction pairs.

Following this property of contradiction relations, we collect a set of pattern pairs $\{\langle p, r\rangle\}$ for which there exists a pattern q such that **ENT** recognizes that p entails q and **BASE** recognizes that q contradicts r. Then we rank these pairs based on a novel scoring function called *Contradiction Derivation Precision (CDP)* and expand **BASE** training data by adding to it the top-ranked pairs according to CDP in order to train **EXP**. This ranking scheme selects highly accurate contradiction pairs and prevents errors caused by **BASE** and **ENT** from being propagated to **EXP**.

In the following, after defining the patterns for which we acquire contradiction relations, we describe **BASE**, **EXP**, **ENT**, and our expansion scheme.

2.1 Patterns

In this work, a binary pattern is a word sequence on the path of dependency relations connecting two nouns in a syntactic dependency tree, like "*X causes Y*", and we say a noun pair *co-occurs with* a pattern if the two nouns are connected by this pattern in the dependency tree of a sentence in the corpus.

We focus on *typed* binary patterns, which place semantic class restrictions on the noun pairs they co-occur with, e.g., "$Y_{organization}$ *is in* $X_{location}$". Subscripts *organization* and *location* indicate the semantic classes of the X and Y slots. Since typed patterns can distinguish between multiple senses of ambiguous patterns, they greatly reduce errors due to pattern ambiguity (De Saeger et al., 2009; Schoenmackers et al., 2010; Berant et al., 2011). We automatically induced semantic classes from our corpus using the EM-based noun clustering algo-

rithm presented in Kazama and Torisawa (2008), and clustered one million nouns into 500 relatively clean semantic classes, including for example classes of *diseases* and of *chemical substances*.

The binary patterns and their co-occurring noun pairs were extracted from our corpus of 600 million Japanese web pages dependency parsed with KNP (Kurohashi and Nagao, 1994). We restricted our patterns to the most frequent 3.9 million patterns of the form "*X-[case particle] Y-[case particle] predicate*" such as "*X-ga Y-ni aru*" ("*X is in Y*") which do not contain any negation, number, symbol or punctuation character. Based on our observation that patterns in meaningful contradiction and entailment pairs tend to share many co-occurring noun pairs, we used as input to our classifiers the set P_{all} of 792 million pattern pairs for which both patterns share three co-occurring noun pairs.

2.2 BASE: First stage Classifier for Contradiction

Below, we detail **BASE**: its training data and input data to be classified, and some experimental results.

Our first stage classifier for contradictions, **BASE**, is an SVM that uses commonsensical surface and lexical resources based features, such as n-grams extracted from patterns, which will be detailed in Section 4. An important point to be stressed here is that we restricted the pattern pairs to be classified by **BASE** by exploiting their *excitation* polarity, a semantic orientation proposed by Hashimoto et al. (2012). Excitation characterizes unary patterns as *excitatory*, *inhibitory*, or *neutral*. *Excitatory* unary patterns, such as "*cause X*" or "*increase X*", entail that the function, effect, purpose, or role of their argument's referent is activated or enhanced, and *inhibitory* unary patterns, such as "*prevent X*" or "*X disappears*", entail that the function, effect, purpose, or role of their argument's referent is deactivated or suppressed. Neutral unary patterns like "*close to X*" are neither excitatory nor inhibitory.

We exploited *excitation* to restrict the input of **BASE**. Based on the result of Hashimoto et al. (2012) showing that two unary patterns with opposite polarity have a higher chance to be a contradiction, we extracted from set P_{all} the set P_{opp} of binary pattern pairs that contain unary patterns with opposite excitation polarities as sub-patterns. $\langle$"*Y cause X*", "*Y prevent X*"$\rangle$ is an example of such a pair since the unary sub-patterns "cause X" and "prevent X" are respectively excitatory and inhibitory. We used here 6,470 excitation unary patterns hand-labeled as either excitatory (4,882 patterns) or inhibitory (1,588 patterns). Set P_{opp} contains 8 million pattern pairs with roughly 38% *true* contradiction pairs, and is the input to **BASE**. We will show in experiments at the end of this section that this restriction is necessary to obtain good performance for **BASE**. We also tried to add the excitation polarities in **BASE**'s feature set and classify P_{all}, but the performance was worse.

Training Data Another key feature of **BASE** is that it is distantly supervised. We did not use training samples that are directly manually annotated. Instead we automatically generated training data from a smaller set of (non-)contradiction unary pattern pairs. We first prepared a set of roughly 800 unary pattern pairs hand-labeled by three human annotators as contradictions (238 pairs) and non-contradictions (558 pairs) using majority vote. The inter-annotator agreement was 0.78 (Fleiss' kappa). Inspired by Hashimoto et al. (2012), we selected these unary pattern pairs among pairs with high distributional similarity, with and without restricting them to having opposite excitation polarity, such as to get a fair distribution of contradictions and non-contradictions.

We then extracted from set P_{all} all 256,000 pattern pairs containing a contradictory unary pattern pair, and all 5.2 million pattern pairs containing a non-contradictory unary pattern pair, which we respectively used as positive and negative training data (estimated 79% and 73% accuracy from 200 hand-labeled samples). Table 1 shows some examples.

The optimal composition of training data for **BASE** was determined according to preliminary experiments using our development set (1,000 manually labelled samples. See Section 3.1). We trained 20 different classifiers using from 6,250 to 50,000 positive samples (4 sets) and from 12,500 to 200,000 negative samples (5 sets), doubling the amounts in each step, for a total of 20 configurations. We could not try a larger training data due to long training time but we do not expect it to be a problem because the worst performance was observed with large train-

Table 1: Examples of training samples for **BASE** obtained from unary pattern pairs

Binary pattern pair (the unary pattern pair that extracted it is underlined)	Unary pattern pair label
Y も X が悪い (X is bad in Y too) - Y でも X が良い (X is good even in Y)	contradiction
Y も X に向かう (Y too heads toward X) - Y も X を出る (Y too comes out of X)	contradiction
X にY を 加える (add Y to X) - X をY に 入れる (insert X into Y)	non-contradiction
Y も X に来る (Y too comes to X) - Y とは X に行く (go to X with Y)	non-contradiction

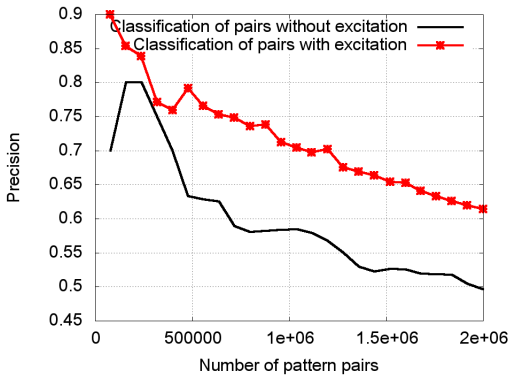


Figure 3: Effect of the restriction using excitation

ing data (25,000 positives and 200,000 negatives; the difference from the optimal setting was 2.3% in average precision). The optimal training data set, $Train_{base}$, consists of 12,500 positives and 100,000 negatives samples as described above and is the one we use in our experiments below and in Section 3.

Since **BASE** input for classification data is P_{opp} we also tried sampling $Train_{base}$ from P_{opp}. We obtained 56.27% average precision for our classifier **BASE**, and 52.99% when restricting the source of training data to pairs in P_{opp}. We believe that the difference lies in the size of the sets from which we sampled our training data: while there are 5.46 million binary pattern pairs in P_{all} with a hand-labeled unary pattern pair in P_{all}, there are only 237,000 pairs in P_{opp}. We believe this much smaller sample source lead to a lower performance because it included much less variations of the patterns.

To train **BASE** and other classifiers mentioned in this paper, we used the SVM tool *TinySVM*[2] with a polynomial kernel of degree 2, the setting which showed the best performance during our preliminary experiments.

Effect of Excitation Polarities We also empirically examined the effect of the restriction on the patterns using excitation polarities. We used our test set (2,000 manually annotated samples described in Section 3.1) and 250 manually annotated samples (majority vote from 3 annotators) from top ranked pairs of P_{all} to draw precision curves for **BASE** over the top 2 million binary pairs from both P_{opp} and P_{all}. In each case we assumed that pairs were distributed uniformly (i.e., with a constant interval) in the ranked list of pairs of P_{opp} and P_{all}, and computed precision accordingly. Since the pairs sets are reasonably large and were sampled randomly we thought this was a reasonable hypothesis. The precision over P_{opp} is higher than that over P_{all} with a large margin, suggesting that the restriction using excitation polarities is beneficial.

[2] http://chasen.org/~taku/software/TinySVM/

2.3 ENT: First stage Classifier for Entailment

ENT is an SVM classifier for entailment trained using 27,500 hand-annotated binary pattern pairs (set $Train_{ent}$, 45% of positive entailment pairs) created for some previous work (Kloetzer et al., 2013). It essentially uses the same feature set as that for **BASE** with the addition of several *distributional* similarity measures (see Section 4 below for more details). This classifier is given all pairs of P_{all} as input and scores each of them. For this study, we considered the 44.5 million pattern pairs with a positive SVM score as entailment pairs. Manual annotation of 200 random samples revealed that the precision of these pairs was 63% and that the top 7.1 million pairs had 80% precision (result interpolated from the top 16% of the annotated samples).

2.4 Second stage: Training Data Expansion and Classifier EXP

Below, we show how we combine **BASE**'s top output (hereafter C) and **ENT**'s top output (hereafter E) in the second stage of our method to expand $Train_{base}$ and train a new classifier, **EXP**.

The training data expansion process is based on the following logical constraint: if a pattern p entails a pattern q and pattern q contradicts a third pattern r, then p must contradict r. For example, because "X

Table 2: Examples of triplets $\langle p, q, r \rangle$ where p entails q, q contradicts r, and hence p contradicts r

Pattern p	Pattern q	Pattern r	X/Y examples	$SVM\ Score(p,r)$	$CDP(p,r)$
Y から X が消える X disappears from Y	Y から X が無くなる X vanishes from Y	Y が X に満ちる Y is full of X	怒り/眼 anger/eye	0.3	0.98
Y に X を停止する stop X in Y	Y に X を終える finish X in Y	Y から X を始める start X in Y	4月/活動 April/activity	-0.3	0.61
X は Y を示す X shows Y	X が Y を持つ X have Y	X は Y を失う X loses Y	チーム/自信 team/confidence	0.07	0.45

Algorithm 1 Training data expansion: C is the top 5% output of **BASE**, E is the top output of **ENT** (score > 0)

1: **procedure** EXPAND(C, E)
2: Compute the set of expanded pairs $C' = \{\langle p, r\rangle \mid \exists q : \langle p, q\rangle \in E, \langle q, r\rangle \in C\}$.
3: Rank the pairs in C' using *CDP*.
4: Add the N top-ranked pairs in $C' \setminus C$ as new positive samples to $Train_{base}$.
5: Remove incoherent negative training samples using *negative cleaning*.
6: **end procedure**

causes Y" (pattern p) entails "*X promotes Y*" (pattern q) and the latter contradicts "*X prevents Y*" (pattern r), we conclude that "*X causes Y*" (p) contradicts "*X prevents Y*" (r). We call the former contradiction $\langle q, r\rangle$ a *source* contradiction pair, and the later pair $\langle p, r\rangle$ an *expanded* contradiction pair. Based on this idea, we combine C and E to aggressively expand $Train_{base}$. This process is described in Algorithm 1, and Table 2 shows examples of triples $\langle p, q, r\rangle$ obtained in our experiments.

Expanding pairs from C and E compounds the errors made by **BASE** and **ENT**, hence it is crucial to select a highly precise subset of the expanded pairs. Taking the top pairs according to their SVM score would achieve this, but since **BASE** already handles correctly such pairs, they should not help much as new training data. We therefore propose a new scoring function for selecting highly precise expanded pairs: *Contradiction Derivation Precision* (CDP).

CDP was designed according to the following assumption: a source contradiction pair that derives *correct* expanded pairs with a high *precision* should be *reliable*. Probably, *all* the expanded pairs derived from such a reliable source pair will be *correct* and should be included in the new training data .

In our formulation of CDP, correctness of an expanded pair is judged according to the pair's SVM score using **BASE**. In other words, we regard an expanded pair that has an SVM score above some threshold α as a *true* contradiction. A source contradiction pair that derives *true* contradiction pairs with a high *precision* is regarded as a *reliable* source contradiction pair. CDP, which is defined over a expanded pairs, is the maximum precision among that of the source contradiction pairs that derive a given expanded pair.

We first define $CDPsub(q, r)$ over a source contradiction pair $\langle q, r\rangle$ as the ratio of expanded pairs obtained from $\langle q, r\rangle$ whose SVM score is above threshold α. This ratio corresponds to the *precision* of the expanded pairs derived from the source contradiction pair $\langle q, r\rangle$.

$$CDPsub(q,r) = \frac{|\{\langle p, r\rangle \in Ex(q,r) \mid Sc(p,r) > \alpha\}|}{|Ex(q,r)|}$$

Here $Ex(q, r)$ is the set of expanded pairs derived from a source pair $\langle q, r\rangle$, and Sc is the SVM score given by **BASE**. In our experiments, we set $\alpha = 0.46$ such that pattern pairs for which **BASE** gives a score over α corresponds to the top 5% of **BASE**'s output. $CDP(p, r)$ over an expanded pair is defined as follows, where $Source(p, r)$ is the set of source contradiction pairs that were derived into the expanded pair $\langle p, r\rangle$.

$$CDP(p,r) = max_{\langle q,r\rangle \in Source(p,r)} CDPsub(q,r)$$

We then expand the top 5% contradictions of **BASE**'s output (set C) and pattern pairs scored positively by **ENT** (set E), rank all expanded pairs not already in C according to CDP, and add the top N pairs with the highest CDP values as positives to $Train_{base}$ to train **EXP**. The value of N shall be determined empirically in later experiments using a development set. Note that, since $CDP(p, r)$ is independent of $\langle p, r\rangle$'s SVM score, even pairs that were assigned a negative score by **BASE** can become highly ranked by CDP (second triplet in Table 2)

and be added to train **EXP**, hence we expect **EXP** to learn something new from these pairs.

Finally, after the addition of expanded pairs, we remove incoherent training samples. We propose to remove from the *negative* training samples of **EXP** any pattern pair that may conflict with the newly added positives; we call this step *negative cleaning*. Intuitively, since the content word pairs in a pattern pair should present some of the strongest evidence for determining the patterns (non-)contradiction status, we remove any negative sample that shares a content word pair with one of the added expanded pairs. The final training data for **EXP**, set $Train_{exp}$, consists of the following: **(1)** positive samples from $Train_{base}$, **(2)** (positive) expanded pairs, and **(3)** negative training samples from $Train_{base}$, cleaned using *negative cleaning*. We confirmed in our experiments that *negative cleaning* was necessary to train a strong **EXP** classifier (details omitted for reason of space).

After training **EXP** with $Train_{exp}$, we classify P_{opp} with **EXP** to produce the final output of the whole method. Note that while this expansion process can be re-iterated with **EXP**'s output, our experiments failed to show any improvement with subsequent iterations.

3 Evaluation

This section presents our experimental results. We describe first how we constructed test and development data, and then report comparison results between our method and others including **BASE** and an Integer Linear Programming-based (ILP) method.

3.1 Development and Test Data

We asked three human annotators to label 3,000 binary pattern pairs randomly sampled from P_{opp} as contradiction or non-contradiction to be used as development (1,000 pairs) and test (2,000 pairs) sets. We considered a pattern pair as a true contradiction relation if at least two out of the three annotators marked it as positive. The inter-rater agreement score (Fleiss Kappa) was 0.523, indicating moderate agreement (Landis and Koch, 1977). As a definition of contradiction, we used the notion of *incompatibility* (i.e., two statements are extremely unlikely to be simultaneously true) proposed by De Marneffe et al. (2008). We then say binary patterns such as "*X causes Y*" and "*X prevents Y*" are contradictory if the above definition holds for any noun pair that can instantiate the patterns' variables in the provided semantic class pair.

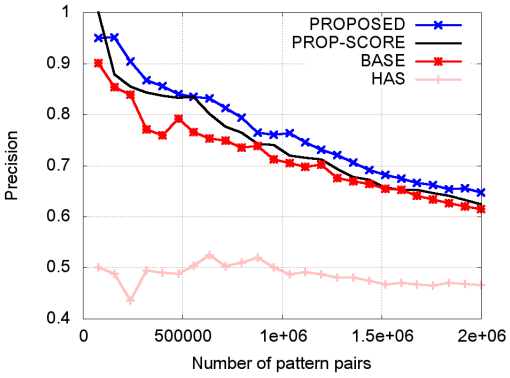


Figure 4: Precision of all the compared methods

Because our semantic classes are obtained by automatic clustering and have no meaningful labels, we followed Szpektor et al. (2007) and provided the annotators with three random noun pairs that co-occur with the patterns as a proxy for the class pair. The annotators marked a given pattern pair as positive if the contradiction relation between the patterns held for all three noun pairs presented.

3.2 Experimental Results

Here we show how our proposed method outperforms baseline methods. We compare the following four methods:

- **PROPOSED**: our proposed method. N, the number of newly added positive training samples during the training data expansion process, was set to 6,000 according to preliminary experiments using the development set. We tried 50 different values of N from 1,000 up to 50,000, adding 1,000 each time, and chose the N value giving the highest average precision against our development set (1,000 samples).
- **BASE**: our first stage classifier.
- **PROP-SCORE**: same as **PROPOSED** except for the use of **BASE**'s SVM score instead of CDP. N was set to 30,000 in the same way we set N for **PROPOSED**.
- **HAS**: an adaptation of the contradiction extraction method presented in Hashimoto et al.

(2012). For a binary pattern pair we first extracted its unary pattern pair with opposite polarity (or one at random in case there are two) and scored it based on our implementation of Hashimoto et al. (2012); the score is based on the distributional similarity between unary patterns and an excitation score obtained using a minimally supervised method based on the spin model. We then scored the binary pattern pair by the score of this unary pattern pair.

We ranked the pattern pairs of our test set (2,000 random pairs from set P_{opp}) based on the score produced by each method. For each tested method we assumed that pairs in the test set were distributed uniformly like explained in Section 2.2. The precision curves we obtained are shown in Figure 4.

PROPOSED clearly outperformed **BASE** and acquired around 750,000 contradiction pattern pairs with an estimated precision of 80%, out of which some examples are shown in Table 3. These pairs cover 26,941 content word pairs and reduce to 272,164 *untyped* pairs, showing that **PROPOSED** does not just acquire a handful of contradictions in many different class pairs. Also, when matching these pairs against an antonyms database (extracted from the dictionary of the morphical analyzer JUMAN) we found that only 100,886 of these pattern pairs contain an antonym pair, which means that most of the extracted pairs' contradictions are due to more complex phenomena than simple antonymy.

With the same precision, **BASE** and **PROP-SCORE** acquired only 285,000 pairs (covering 11,794 content word pairs) and 636,000 pairs respectively. This implies that our two-stage method can more than double the number of highly precise contradiction pairs we acquire as well as increasing their variety, and that ranking expanded pairs using our scoring function CDP is better than with SVM score, though even **PROP-SCORE** performs better than **BASE** in our setting. Finally, the poor performance of **HAS** suggests that extending the Hashimoto et al.'s framework to recognition of binary patterns is not a trivial task.

As to why adding only 6,000 top pairs ranked by CDP performs better than adding 30,000 pairs ranked by SVM score, the pattern pairs added in **PROP-SCORE** had high SVM scores given by **BASE** and as such are already handled nicely by **BASE**. Hence, we think the effect of adding a new sample from **PROP-SCORE** is smaller than that in **PROPOSED**, because in **PROPOSED** we add to the training data pattern pairs with both high and low (possibly negative) SVM scores.

Table 3: Examples of pairs acquired by **PROPOSED**: contradiction (label +) and non-contradiction (label -)

Lab.	Pattern pairs (with rank)	X/Y example
+	Y で X が終わる - Y より X を開始する X finished Y - X started from Y Rank 228,039	販売/昨日 sale/yesterday
+	X が Y に勝つ - Y が X に勝つ X wins against Y - Y wins against X Rank: 258,068	日本/ベトナム Japan/Vietnam
-	X は Y を失う - X には Y はある X lose Y - Have Y in X Rank 474,143	人/興味 people/interest
+	Y に X を無くす - Y にも X をもつ Lose X in Y - Have X in Y too Rank 522,534	自信/自分 confidence/ oneself
-	Y は X まで落ちる - X に Y を上げる Y falls down to X - raise Y to X Rank 538,901	9 位/順位 9th/ranking
+	X に Y が存在する - X から Y を防ぐ Y exists in X - Keep Y out X Rank 620,430	中/ウイルス inside/virus
-	X から Y を外す - X は Y で答える Remove Y off X - X answer with Y Rank 652,530	僕/目 I (or me)/eyes
+	Y を X から追い出す - X に Y が残る Kick out Y from X - Y remains in X Rank 697,177	体/疲労 body/fatigue
+	Y が X を安心させる - Y が X がを裏切る X reassures Y - X betrays Y Rank: 749,916	僕/彼女 I/her

Finally, while the quality of the entailment pairs plays a very important role in the assumption that was the base of CDP, these results show that even a simple rule such as "Use entailment pairs with SVM score over 0 to expand contradictions before ranking them with CDP" is sufficient to make the method work. Though it may be possible to design a more complex CDP formula which takes entailment score into account, we did not explore this direction in this work.

Comparison with an ILP-based method Finally, we would like to compare our method with an ILP-based method. The interaction between contradiction and entailment that forms the basis for our expansion method has a natural interpretation as an optimization problem. We thus compared our method to the following ILP formulation of this interaction inspired by Berant et al. (2011), using our test set:

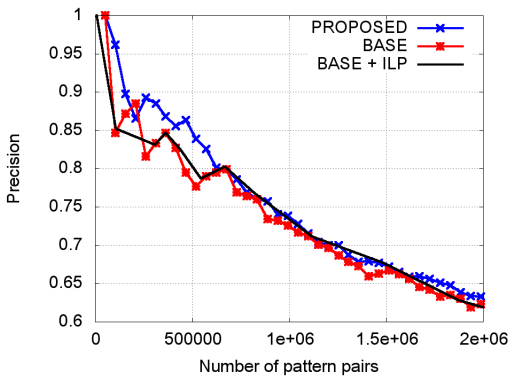


Figure 5: Comparison between **PROPOSED**, **BASE** and **BASE+ILP** on a restricted test set (1,306 samples)

(1) $G = argmax \sum_{p \neq q} (e(p,q) - \beta) * E_{pq} + (c(p,q) - \beta) * C_{pq}$

(2) s.t. $\forall_{p,q,r}\ E_{pq} + C_{qr} - C_{pr} \leq 1$

(3) $\forall_{p,q}\ E_{pq} + C_{pq} \leq 1$

(4) $\forall_{p,q}\ E_{pq} \in \{0,1\}$ **(5)** $\forall_{p,q}\ C_{pq} \in \{0,1\}$

The objective in Equation (1) is a sum over the weights of every pair of patterns $\langle p,q \rangle$, where E_{pq} indicates whether a pair $\langle p,q \rangle$ is an entailment pair (Equation (4)), and C_{pq} indicates whether it is a contradiction pair (Equation (5)). $e(p,q)$ and $c(p,q)$ are the score given respectively by **ENT** and **BASE**, and β is a prior defining the weight of a pair as neither entailment nor contradiction that shall be set before any experimentation. Equation (2) states the transitivity relation which is the basis of our expansion method. Finally, Equation (3) states that a given pattern pair cannot be a contradiction pair and an entailment pair at the same time. Since our patterns are class-dependent, we solved separate ILP instances for each semantic class pair.

We drew a precision curve for each of **BASE**, **PROPOSED** and **BASE+ILP**. To draw the curve for **BASE+ILP**, we incrementally raised the sample's non-contradiction non-entailment prior β (more details in Berant et al. (2011)). Because of the computational difficulty of ILP (NP-complete) and the size of our data, the computation for the ILP-based method ran out of memory on a 72GB machine for 116 class pairs out of the 1,031 that our test set covers. For this reason, we only used the 1,306 samples of the test set covered by the remaining 915 class pairs. We also measured the performance of **BASE** and **PROPOSED** on the same restricted test set.

Figure 5 shows that under these conditions the ILP-based method performance resembles **BASE** and is worse than **PROPOSED** on all data points. **PROPOSED** performs slightly worse in this setting compared to when classifying the whole of P_{opp}, but this only means that its performance is good for the 116 class pairs we ignored in this experiment. While this comparison is only made in a restricted setting, our expansion method still outperforms ILP and is clearly more scalable. The ILP results could be improved by adding more constraints (*contradiction is symmetric*, *entailment is transitive*), but this would also make the problem even more intractable in terms of computational costs.

4 Features

In this section we present the features used in our classifiers, which are mainly categorized into three: surface features (i.e., those reflecting the patterns' content itself), features based on external lexical resources, and distributional similarity based features; all features are listed in Table 4. **ENT** uses all the features while **BASE** and **EXP** use all except for the distributional similarity based ones. The optimality of the feature sets was confirmed through ablation tests using the development set (results omitted for the sake of space).

Since patterns with a contradiction or entailment relation are often superficially similar, for instance, in case structure or inflection, we use a number of *surface* features based on string similarity measures, extending the feature sets used by Malakasiotis and Androutsopoulos (2007) for entailment recognition. They include bag-of-words features such as n-grams and similarity scores concerning the bag-of-words such as their Euclidian distance.

To complement the surface features with knowledge about the content words, we used lexical databases including such as antonymy, synonymy, entailment, or allography. The presence of such word pairs is usually a good indicator of (non-)contradiction or (non-)entailment at the pattern level. More specifically, for any word pair $\langle wp, wq \rangle$ taken from a pattern pair $\langle p,q \rangle$ we mark the presence of $\langle wp, wq \rangle$ in each of the lexical resources as a binary feature. We used the Japanese lexical resources distributed by the *ALAGIN Forum*[3]: the verb entailment database (117,000 verb

[3] http://www.alagin.jp/

Table 4: Features summary, computed over a pair of patterns $\langle p, q \rangle$

surface	**Similarity measures:** common elements ratios, Dice coefficient, Jaccard and discounted Jaccard scores, Cosine, Euclidian, Manhattan, Levenshtein and Jaro distances; *computed over:* the patterns' 1-, 2- and 3-grams sets of: characters, morphemes, their stems & POS; content words and stems
	binary feature for each of the patterns' subtrees, 1- and 2-grams ; patterns' lengths and length ratios
lex.r.	entries in databases of verb entailments and non-entailments, synonyms, antonyms, allographs ; *checked over:* pairs of content words, pairs of content word stems, same for the reverse pattern pair $\langle q, p \rangle$
dis.s.	**Distributional similarity measures:** Common elements ratios, Jaccard and discounted Jaccard scores, sets and sets intersection cardinality, DIRT (Lin and Pantel, 2001), Weeds (Weeds and Weir, 2003) and Hashimoto (Hashimoto et al., 2009) scores; *computed over*: patterns' co-occurring noun pairs, POS tags of those, nouns co-occurring in each variable slot, nouns co-occurring with each unary sub-patterns
other	binary feature for each semantic class pair and individual semantic classes
	patterns frequency rank in the given semantic class pair

pairs; Alagin ID A-2), the databases of synonyms, antonyms and meronyms (respectively 111,000, 5000 and 2500 pairs; Alagin ID A-9), and the allographic word database (2.7 million pairs; Alagin ID A-7). We also used the information concerning allographic words in the dictionary of the morphological analyzer JUMAN[4].

Distributional similarity values between patterns are based on the idea that patterns that appear in similar contexts tend to have similar meanings and as such are useful to recognize entailment (Lin and Pantel, 2001). We computed as features several distributional similarity measures on the sets of each pattern's co-occurring noun pairs and their POS tags, of nouns co-occurring in each variable slot, and with each of the pattern's unary sub-patterns.

We also added a few more uncategorizable features. See Table 4 for more details.

5 Related Work

A number of previous work dealt with the recognition of contradictions between sentences. Harabagiu et al. (2006) proposed a contradiction detection method that focuses on negation, antonymy and some discourse information. Kawahara et al. (2010) also used negations and antonyms to extract contrastive/contradictory statements from the web to present users with a bird ' s-eye view of statements about a given topic. Bobrow et al. (2007) showed a method using logical forms with relatively precise results. Ohki et al. (2011) proposed a method to recognize *confinment*, a novel semantic relation related to both entailment and contradiction. While we do not deal ourselves directly with sentences, we expect that the binary pattern pairs we acquire can play a role similar to that of basic linguistic resources such as antonyms and negations in these works. Closer to our work, Ritter et al. (2008) presented a method for detecting contradictions between *functional* relations like "*X was born in Y*", but these constitute only a part of the semantic relations expressed by the binary patterns we deal with in this paper.

Other works analyzed contradictions from linguistic/semantic viewpoints. Voorhees (2008) analyzed the contradiction recognition-task of the RTE3 contest. Magnini and Cabrio (2010) examined relations between contradictions and textual entailment samples. De Marneffe et al. (2008) presented a typology of contradictions, and showed that contradictions can arise from a multitude of phenomena. They showed contradictions based on lexical or world knowledge are challenging and require a high-level understanding of language and/or the world. As stated in the introduction, these are the types of contradictions our method focuses on.

6 Conclusion

This paper showed how to acquire a large number of contradiction pairs between lexico-syntactic binary patterns by exploiting **(1)** the interaction between contradiction and entailment, and **(2)** *excitation* polarities. In the end, we could acquire 750,000 typed contradiction pattern pairs with an estimated 80% precision. The resulting contradiction pairs covered ones deeply related to world knowledge such as the pair $\langle$"*X reassures Y*", "*X betrays Y*"$\rangle$. We expect our work to lead to a high level analysis of textual information, such as flagging unreliable information or identifying important documents to be surveyed for understanding complex social problems. We plan to release the data we acquired to the NLP community through the *ALAGIN Forum*[5].

[4] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

[5] http://www.alagin.jp/

References

J. Berant, I. Dagan, and J. Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL 2011*, pages 610–619.

D. G. Bobrow, C. Condoravdi, R. Crouch, V. De Paiva, L. Karttunen, T. H. King, R. Nairn, L. Price, and A. Zaenen. 2007. Precision-focused textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, page 16—21.

M.-C. De Marneffe, A. N. Rafferty, and C. D. Manning. 2008. Finding contradictions in text. *Proceedings of ACL 2008*, page 1039—1047.

S. De Saeger, K. Torisawa, J. Kazama, K. Kuroda, and M. Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proceedings of ICDM 2009*, page 764—769.

S.M. Harabagiu, A. Hickl, and V.F. Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *Proceedings of AAAI 2006*, pages 755–762.

C. Hashimoto, K. Torisawa, K. Kuroda, S. De Saeger, M. Murata, and J. Kazama. 2009. Large-scale verb entailment acquisition from the web. In *Proceedings of EMNLP 2009*, volume 3, page 1172—1181.

C. Hashimoto, K. Torisawa, S. De Saeger, J.-H. Oh, and J. Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of EMNLP 2012*.

D. Kawahara, S. Kurohashi, and K. Inui. 2008. Grasping major statements and their contradictions toward information credibility analysis of web contents. In *Proceedings of WI-IAT 2008*, volume 1, page 393—397.

D. Kawahara, K. Inui, and S. Kurohashi. 2010. Identifying contradictory and contrastive relations between statements to outline web information on a given topic. In *Proceedings of COLING 2010*, page 534—542.

J. Kazama and K. Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. *Proceedings of ACL 2008*, page 407—415.

J. Kloetzer, S. De Saeger, K. Torisawa, M. Sano, C. Hashimoto, and J. Gotoh. 2013. Large-scale acquisition of entailment pattern pairs. In *Information Processing Society of Japan (IPSJ) Kansai-Branch Convention*.

S. Kurohashi and M. Nagao. 1994. KN parser: Japanese dependency/case structure analyzer. In *Proceedings of the Workshop on Sharable Natural Language Resources*, page 48—55.

J. R. Landis and G. G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, page 159—174.

D. Lin and P. Pantel. 2001. Dirt - discovery of inference rules from text. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

B. Magnini and E. Cabrio. 2010. Contradiction-focused qualitative evaluation of textual entailment. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, page 86—94.

P. Malakasiotis and I. Androutsopoulos. 2007. Learning textual entailment using SVMs and string similarity measures. In *Proceedings of the ACL- PASCAL Workshop on Textual Entailment and Paraphrasing*, page 42—47.

K. Murakami, E. Nichols, S. Matsuyoshi, A. Sumida, S. Masuda, K. Inui, and Y. Matumoto. 2009. Statement map: assisting information crediblity analysis by visualizing arguments. In *Proceedings of the 3rd workshop on Information credibility on the web*, page 43—50. ACM.

M. Ohki, S. Matsuyoshi, J. Mizuno, K. Inui, E. Nichols, K. Murakami, S. Masuda, and Y. Matsumoto. 2011. Recognizing confinement in web texts. In *the Proceedings of the Ninth International Conference on Computational Semantics*, page 215—224.

A. Ritter, D. Downey, S. Soderland, and O. Etzioni. 2008. It's a contradiction—no, it's not: a case study using functional relations. In *Proceedings of EMNLP 2008*, pages 11–20.

S. Schoenmackers, O. Etzioni, D. S Weld, and J. Davis. 2010. Learning first-order horn clauses from web text. In *Proceedings of EMNLP 2010*, page 1088—1098.

I. Szpektor, E. Shnarch, and I. Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of ACL 2007*, volume 45, page 456—463.

E. M. Voorhees. 2008. Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of ACL 2008*, page 63—71.

J. Weeds and D. Weir. 2003. A general framework for distributional similarity. In *Proceedings of EMNLP 2003*, page 81—88. Association for Computational Linguistics.

Sarcasm as Contrast between a Positive Sentiment and Negative Situation

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva,
Nathan Gilbert, Ruihong Huang
School Of Computing
University of Utah
Salt Lake City, UT 84112
{riloff,asheq,alnds,ngilbert,huangrh}@cs.utah.edu, prafulla.surve@gmail.com

Abstract

A common form of sarcasm on Twitter consists of a positive sentiment contrasted with a negative situation. For example, many sarcastic tweets include a positive sentiment, such as "love" or "enjoy", followed by an expression that describes an undesirable activity or state (e.g., "taking exams" or "being ignored"). We have developed a sarcasm recognizer to identify this type of sarcasm in tweets. We present a novel bootstrapping algorithm that automatically learns lists of positive sentiment phrases and negative situation phrases from sarcastic tweets. We show that identifying contrasting contexts using the phrases learned through bootstrapping yields improved recall for sarcasm recognition.

1 Introduction

Sarcasm is generally characterized as ironic or satirical wit that is intended to insult, mock, or amuse. Sarcasm can be manifested in many different ways, but recognizing sarcasm is important for natural language processing to avoid misinterpreting sarcastic statements as literal. For example, sentiment analysis can be easily misled by the presence of words that have a strong polarity but are used sarcastically, which means that the opposite polarity was intended. Consider the following tweet on Twitter, which includes the words "yay" and "thrilled" but actually expresses a negative sentiment: "*yay! it's a holiday weekend and i'm on call for work! couldn't be more thrilled! #sarcasm.*" In this case, the hashtag #sarcasm reveals the intended sarcasm, but we don't always have the benefit of an explicit sarcasm label.

In the realm of Twitter, we observed that many sarcastic tweets have a common structure that creates a positive/negative contrast between a sentiment and a situation. Specifically, sarcastic tweets often express a positive sentiment in reference to a negative activity or state. For example, consider the tweets below, where the positive sentiment terms are underlined and the negative activity/state terms are *italicized*.

(a) Oh how I love *being ignored*. #sarcasm

(b) Thoroughly enjoyed *shoveling the driveway* today! :) #sarcasm

(c) Absolutely adore it when *my bus is late* #sarcasm

(d) I'm so pleased mom *woke me up* with vacuuming my room this morning. :) #sarcasm

The sarcasm in these tweets arises from the juxtaposition of a positive sentiment word (e.g., love, enjoyed, adore, pleased) with a negative activity or state (e.g., being ignored, bus is late, shoveling, and being woken up).

The goal of our research is to identify sarcasm that arises from the contrast between a positive sentiment referring to a negative situation. A key challenge is to automatically recognize the stereotypically negative "situations", which are activities and states that most people consider to be unenjoyable or undesirable. For example, stereotypically unenjoyable activities include going to the dentist, taking an exam, and having to work on holidays. Stereotypically undesirable states include being ignored, having no friends, and feeling sick. People recognize

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 704–714,
Seattle, Washington, USA, 18-21 October 2013.

these situations as being negative through cultural norms and stereotypes, so they are rarely accompanied by an explicit negative sentiment. For example, *"I feel sick"* is universally understood to be a negative situation, even without an explicit expression of negative sentiment. Consequently, we must learn to recognize phrases that correspond to stereotypically negative situations.

We present a bootstrapping algorithm that automatically learns phrases corresponding to positive sentiments and phrases corresponding to negative situations. We use tweets that contain a sarcasm hashtag as positive instances for the learning process. The bootstrapping algorithm begins with a single seed word, "love", and a large set of sarcastic tweets. First, we learn *negative situation phrases* that follow a positive sentiment (initially, the seed word "love"). Second, we learn *positive sentiment phrases* that occur near a negative situation phrase. The bootstrapping process iterates, alternately learning new negative situations and new positive sentiment phrases. Finally, we use the learned lists of sentiment and situation phrases to recognize sarcasm in new tweets by identifying contexts that contain a positive sentiment in close proximity to a negative situation phrase.

2 Related Work

Researchers have investigated the use of lexical and syntactic features to recognize sarcasm in text. Kreuz and Caucci (2007) studied the role that different lexical factors play, such as interjections (e.g., "gee" or "gosh") and punctuation symbols (e.g., '?') in recognizing sarcasm in narratives. Lukin and Walker (2013) explored the potential of a bootstrapping method for sarcasm classification in social dialogue to learn lexical N-gram cues associated with sarcasm (e.g., "oh really", "I get it", "no way", etc.) as well as lexico-syntactic patterns.

In opinionated user posts, Carvalho et al. (2009) found oral or gestural expressions, represented using punctuation and other keyboard characters, to be more predictive of irony[1] in contrast to features representing structured linguistic knowledge in Portuguese. Filatova (2012) presented a detailed description of sarcasm corpus creation with sarcasm annotations of Amazon product reviews. Their annotations capture sarcasm both at the document level and the text utterance level. Tsur et al. (2010) presented a semi-supervised learning framework that exploits syntactic and pattern based features in sarcastic sentences of Amazon product reviews. They observed correlated sentiment words such as "yay!" or "great!" often occurring in their most useful patterns.

Davidov et al. (2010) used sarcastic tweets and sarcastic Amazon product reviews to train a sarcasm classifier with syntactic and pattern-based features. They examined whether tweets with a sarcasm hashtag are reliable enough indicators of sarcasm to be used as a gold standard for evaluation, but found that sarcasm hashtags are noisy and possibly biased towards the hardest form of sarcasm (where even humans have difficulty). González-Ibáñez et al. (2011) explored the usefulness of lexical and pragmatic features for sarcasm detection in tweets. They used sarcasm hashtags as gold labels. They found positive and negative emotions in tweets, determined through fixed word dictionaries, to have a strong correlation with sarcasm. Liebrecht et al. (2013) explored N-gram features from 1 to 3-grams to build a classifier to recognize sarcasm in Dutch tweets. They made an interesting observation from their most effective N-gram features that people tend to be more sarcastic towards specific topics such as school, homework, weather, returning from vacation, public transport, the church, the dentist, etc. This observation has some overlap with our observation that stereotypically negative situations often occur in sarcasm.

The cues for recognizing sarcasm may come from a variety of sources. There exists a line of work that tries to identify facial and vocal cues in speech (e.g., (Gina M. Caucci, 2012; Rankin et al., 2009)). Cheang and Pell (2009) and Cheang and Pell (2008) performed studies to identify acoustic cues in sarcastic utterances by analyzing speech features such as speech rate, mean amplitude, amplitude range, etc. Tepperman et al. (2006) worked on sarcasm recognition in spoken dialogue using prosodic and spectral cues (e.g., average pitch, pitch slope, etc.) as well as contextual cues (e.g., laughter or response to questions) as features.

[1] They adopted the term 'irony' instead of 'sarcasm' to refer to the case when a word or expression with prior positive polarity is figuratively used to express a negative opinion.

While some of the previous work has identified specific expressions that correlate with sarcasm, none has tried to identify contrast between positive sentiments and negative situations. The novel contributions of our work include explicitly recognizing contexts that contrast a positive sentiment with a negative activity or state, as well as a bootstrapped learning framework to automatically acquire positive sentiment and negative situation phrases.

3 Bootstrapped Learning of Positive Sentiments and Negative Situations

Sarcasm is often defined in terms of contrast or "saying the opposite of what you mean". Our work focuses on one specific type of contrast that is common on Twitter: the expression of a positive sentiment (e.g., "love" or "enjoy") in reference to a negative activity or state (e.g., "taking an exam" or "being ignored"). Our goal is to create a sarcasm classifier for tweets that explicitly recognizes contexts that contain a positive sentiment contrasted with a negative situation.

Our approach learns rich phrasal lexicons of positive sentiments and negative situations using only the seed word "love" and a collection of sarcastic tweets as input. A key factor that makes the algorithm work is the presumption that if you find a positive sentiment or a negative situation in a sarcastic tweet, then you have found the source of the sarcasm. We further assume that the sarcasm probably arises from positive/negative contrast and we exploit syntactic structure to extract phrases that are likely to have contrasting polarity. Another key factor is that we focus specifically on tweets. The short nature of tweets limits the search space for the source of the sarcasm. The brevity of tweets also probably contributes to the prevalence of this relatively compact form of sarcasm.

3.1 Overview of the Learning Process

Our bootstrapping algorithm operates on the assumption that many sarcastic tweets contain both a positive sentiment and a negative situation in close proximity, which is the source of the sarcasm.[2] Although sentiments and situations can be expressed in numerous ways, we focus on positive sentiments that are expressed as a verb phrase or as a predicative expression (predicate adjective or predicate nominal), and negative activities or states that can be a complement to a verb phrase. Ideally, we would like to parse the text and extract verb complement phrase structures, but tweets are often informally written and ungrammatical. Therefore we try to recognize these syntactic structures heuristically using only part-of-speech tags and proximity.

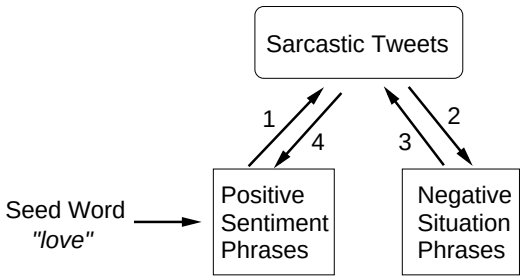


Figure 1: Bootstrapped Learning of Positive Sentiment and Negative Situation Phrases

The learning process relies on an assumption that a positive sentiment verb phrase usually appears to the left of a negative situation phrase and in close proximity (usually, but not always, adjacent). Pictorially, we assume that many sarcastic tweets contain this structure:

[+ VERB PHRASE] [– SITUATION PHRASE]

This structural assumption drives our bootstrapping algorithm, which is illustrated in Figure 1. The bootstrapping process begins with a single seed word, "love", which seems to be the most common positive sentiment term in sarcastic tweets. Given a sarcastic tweet containing the word "love", our structural assumption infers that "love" is probably followed by an expression that refers to a negative situation. So we harvest the n-grams that follow the word "love" as negative situation candidates. We select the best candidates using a scoring metric, and add them to a list of negative situation phrases.

Next, we exploit the structural assumption in the opposite direction. Given a sarcastic tweet that contains a negative situation phrase, we infer that the negative situation phrase is preceded by a positive sentiment. We harvest the n-grams that precede the negative situation phrases as positive sentiment candidates, score and select the best candidates, and

[2] Sarcasm can arise from a negative sentiment contrasted with a positive situation too, but our observation is that this is much less common, at least on Twitter.

add them to a list of positive sentiment phrases. The bootstrapping process then iterates, alternately learning more positive sentiment phrases and more negative situation phrases.

We also observed that positive sentiments are frequently expressed as predicative phrases (i.e., predicate adjectives and predicate nominals). For example: *"I'm taking calculus. It is awesome. #sarcasm"*. Wiegand et al. (2013) offered a related observation that adjectives occurring in predicate adjective constructions are more likely to convey subjectivity than adjectives occurring in non-predicative structures. Therefore we also include a step in the learning process to harvest predicative phrases that occur in close proximity to a negative situation phrase. In the following sections, we explain each step of the bootstrapping process in more detail.

3.2 Bootstrapping Data

For the learning process, we used Twitter's streaming API to obtain a large set of tweets. We collected 35,000 tweets that contain the hashtag #sarcasm or #sarcastic to use as positive instances of sarcasm. We also collected 140,000 additional tweets from Twitter's random daily stream. We removed the tweets that contain a sarcasm hashtag, and considered the rest to be negative instances of sarcasm. Of course, there will be some sarcastic tweets that do not have a sarcasm hashtag, so the negative instances will contain some noise. But we expect that a very small percentage of these tweets will be sarcastic, so the noise should not be a major issue. There will also be noise in the positive instances because a sarcasm hashtag does not guarantee that there is sarcasm in the body of the tweet (e.g., the sarcastic content may be in a linked url, or in a prior tweet). But again, we expect the amount of noise to be relatively small.

Our tweet collection therefore contains a total of 175,000 tweets: 20% are labeled as sarcastic and 80% are labeled as not sarcastic. We applied CMU's part-of-speech tagger designed for tweets (Owoputi et al., 2013) to this data set.

3.3 Seeding

The bootstrapping process begins by initializing the positive sentiment lexicon with one seed word: *love*. We chose this seed because it seems to be the most common positive sentiment word in sarcastic tweets.

3.4 Learning Negative Situation Phrases

The first stage of bootstrapping learns new phrases that correspond to negative situations. The learning process consists of two steps: (1) harvesting candidate phrases, and (2) scoring and selecting the best candidates.

To collect candidate phrases for negative situations, we extract n-grams that follow a positive sentiment phrase in a sarcastic tweet. We extract every 1-gram, 2-gram, and 3-gram that occurs immediately after (on the right-hand side) of a positive sentiment phrase. As an example, consider the tweet in Figure 2, where "love" is the positive sentiment:

I love waiting forever for the doctor #sarcasm

Figure 2: Example Sarcastic Tweet

We extract three n-grams as candidate negative situation phrases:

waiting, waiting forever, waiting forever for

Next, we apply the part-of-speech (POS) tagger and filter the candidate list based on POS patterns so we only keep n-grams that have a desired syntactic structure. For negative situation phrases, our goal is to learn possible verb phrase (VP) complements that are themselves verb phrases because they should represent activities and states. So we require a candidate phrase to be either a unigram tagged as a verb (V) or the phrase must match one of 7 POS-based bigram patterns or 20 POS-based trigram patterns that we created to try to approximate the recognition of verbal complement structures. The 7 POS bigram patterns are: V+V, V+ADV, ADV+V, "to"+V, V+NOUN, V+PRO, V+ADJ. Note that we used a POS tagger designed for Twitter, which has a smaller set of POS tags than more traditional POS taggers. For example there is just a single V tag that covers all types of verbs. The V+V pattern will therefore capture negative situation phrases that consist of a present participle verb followed by a past participle verb, such as "being ignored" or "getting hit".[3] We also allow verb particles to match a V tag in our patterns. The remaining bigram patterns capture verb phrases that include a verb and adverb, an

[3]In some cases it may be more appropriate to consider the second verb to be an adjective, but in practice they were usually tagged as verbs.

infinitive form (e.g., "to clean"), a verb and noun phrase (e.g., "shoveling snow"), or a verb and adjective (e.g., "being alone"). We use some simple heuristics to try to ensure that we are at the end of an adjective or noun phrase (e.g., if the following word is tagged as an adjective or noun, then we assume we are *not* at the end).

The 20 POS trigram patterns are similar in nature and are designed to capture seven general types of verb phrases: verb and adverb mixtures, an infinitive VP that includes an adverb, a verb phrase followed by a noun phrase, a verb phrase followed by a prepositional phrase, a verb followed by an adjective phrase, or an infinitive VP followed by an adjective, noun, or pronoun.

Returning to Figure 2, only two of the n-grams match our POS patterns, so we are left with two candidate phrases for negative situations:

waiting, waiting forever

Next, we score each negative situation candidate by estimating the probability that a tweet is sarcastic given that it contains the candidate phrase following a positive sentiment phrase:

$$\frac{|\ \text{follows(–candidate, +sentiment) \& sarcastic}\ |}{|\ \text{follows(–candidate, +sentiment)}\ |}$$

We compute the number of times that the negative situation candidate immediately follows a positive sentiment in sarcastic tweets divided by the number of times that the candidate immediately follows a positive sentiment in all tweets. We discard phrases that have a frequency < 3 in the tweet collection since they are too sparse.

Finally, we rank the candidate phrases based on this probability, using their frequency as a secondary key in case of ties. The top 20 phrases with a probability $\geq .80$ are added to the negative situation phrase list.[4] When we add a phrase to the negative situation list, we immediately remove all other candidates that are subsumed by the selected phrase. For example, if we add the phrase "waiting", then the phrase "waiting forever" would be removed from the candidate list because it is subsumed by "waiting". This process reduces redundancy in the set of phrases that we add during each bootstrapping iteration. The bootstrapping process stops when no more candidate phrases pass the probability threshold.

3.5 Learning Positive Verb Phrases

The procedure for learning positive sentiment phrases is analogous. First, we collect phrases that potentially convey a positive sentiment by extracting n-grams that precede a negative situation phrase in a sarcastic tweet. To learn positive sentiment verb phrases, we extract every 1-gram and 2-gram that occurs immediately before (on the left-hand side of) a negative situation phrase.

Next, we apply the POS tagger and filter the n-grams using POS tag patterns so that we only keep n-grams that have a desired syntactic structure. Here our goal is to learn simple verb phrases (VPs) so we only retain n-grams that contain at least one verb and consist only of verbs and (optionally) adverbs. Finally, we score each candidate sentiment verb phrase by estimating the probability that a tweet is sarcastic given that it contains the candidate phrase preceding a negative situation phrase:

$$\frac{|\ \text{precedes(+candidateVP,–situation) \& sarcastic}\ |}{|\ \text{precedes(+candidateVP,–situation)}\ |}$$

3.6 Learning Positive Predicative Phrases

We also use the negative situation phrases to harvest predicative expressions (predicate adjective or predicate nominal structures) that occur nearby. Based on the same assumption that sarcasm often arises from the contrast between a positive sentiment and a negative situation, we identify tweets that contain a negative situation and a predicative expression in close proximity. We then assume that the predicative expression is likely to convey a positive sentiment.

To learn predicative expressions, we use 24 copular verbs from Wikipedia[5] and their inflections. We extract positive sentiment candidates by extracting 1-grams, 2-grams, and 3-grams that appear immediately after a copular verb and occur within 5 words of the negative situation phrase, on either side. This constraint only enforces proximity because predicative expressions often appear in a separate clause or sentence (e.g., *"It is just great that my iphone was stolen"* or *"My iphone was stolen. This is great."*)

[4] Fewer than 20 phrases will be learned if < 20 phrases pass this threshold.

[5] http://en.wikipedia.org/wiki/List_of_English_copulae

We then apply POS patterns to identify n-grams that correspond to predicate adjective and predicate nominal phrases. For predicate adjectives, we retain ADJ and ADV+ADJ n-grams. We use a few heuristics to check that the adjective is not part of a noun phrase (e.g., we check that the following word is not a noun). For predicate nominals, we retain ADV+ADJ+N, DET+ADJ+N and ADJ+N n-grams. We excluded noun phrases consisting only of nouns because they rarely seemed to represent a sentiment. The sentiment in predicate nominals was usually conveyed by the adjective. We discard all candidates with frequency < 3 as being too sparse. Finally, we score each remaining candidate by estimating the probability that a tweet is sarcastic given that it contains the predicative expression near (within 5 words of) a negative situation phrase:

$$\frac{|\ \text{near(+candidatePRED,–situation) \& sarcastic}\ |}{|\ \text{near(+candidatePRED,–situation)}\ |}$$

We found that the diversity of positive sentiment verb phrases and predicative expressions is much lower than the diversity of negative situation phrases. As a result, we sort the candidates by their probability and conservatively add only the top 5 positive verb phrases and top 5 positive predicative expressions in each bootstrapping iteration. Both types of sentiment phrases must pass a probability threshold of $\geq .70$.

3.7 The Learned Phrase Lists

The bootstrapping process alternately learns positive sentiments and negative situations until no more phrases can be learned. In our experiments, we learned 26 positive sentiment verb phrases, 20 predicative expressions and 239 negative situation phrases.

Table 1 shows the first 15 positive verb phrases, the first 15 positive predicative expressions, and the first 40 negative situation phrases learned by the bootstrapping algorithm. Some of the negative situation phrases are not complete expressions, but it is clear that they will often match negative activities and states. For example, "getting yelled" was generated from sarcastic comments such as "I love getting yelled at", "being home" occurred in tweets about "being home alone", and "being told" is often being told what to do. Shorter phrases often outranked longer phrases because they are more general, and will therefore match more contexts. But an avenue for future work is to learn linguistic expressions that more precisely characterize specific negative situations.

Positive Verb Phrases (26): missed, loves, enjoy, cant wait, excited, wanted, can't wait, get, appreciate, decided, loving, really like, looooove, just keeps, loveee, ...
Positive Predicative Expressions (20): great, so much fun, good, so happy, better, my favorite thing, cool, funny, nice, always fun, fun, awesome, the best feeling, amazing, happy, ...
Negative Situations (239): being ignored, being sick, waiting, feeling, waking up early, being woken, fighting, staying, writing, being home, cleaning, not getting, crying, sitting at home, being stuck, starting, being told, being left, getting ignored, being treated, doing homework, learning, getting up early, going to bed, getting sick, riding, being ditched, getting ditched, missing, not sleeping, not talking, trying, falling, walking home, getting yelled, being awake, being talked, taking care, doing nothing, wasting, ...

Table 1: Examples of Learned Phrases

4 Evaluation

4.1 Data

For evaluation purposes, we created a gold standard data set of manually annotated tweets. Even for people, it is not always easy to identify sarcasm in tweets because sarcasm often depends on conversational context that spans more than a single tweet. Extracting conversational threads from Twitter, and analyzing conversational exchanges, has its own challenges and is beyond the scope of this research. We focus on identifying sarcasm that is self-contained in one tweet and does not depend on prior conversational context.

We defined annotation guidelines that instructed human annotators to read isolated tweets and label

a tweet as *sarcastic* if it contains comments judged to be sarcastic based solely on the content of that tweet. Tweets that do not contain sarcasm, or where potential sarcasm is unclear without seeing the prior conversational context, were labeled as *not sarcastic*. For example, a tweet such as *"Yes, I meant that sarcastically."* should be labeled as *not sarcastic* because the sarcastic content was (presumably) in a previous tweet. The guidelines did not contain any instructions that required positive/negative contrast to be present in the tweet, so all forms of sarcasm were considered to be positive examples.

To ensure that our evaluation data had a healthy mix of both sarcastic and non-sarcastic tweets, we collected 1,600 tweets with a sarcasm hashtag (#sarcasm or #sarcastic), and 1,600 tweets without these sarcasm hashtags from Twitter's random streaming API. When presenting the tweets to the annotators, the sarcasm hashtags were removed so the annotators had to judge whether a tweet was sarcastic or not without seeing those hashtags.

To ensure that we had high-quality annotations, three annotators were asked to annotate the same set of 200 tweets (100 sarcastic + 100 not sarcastic). We computed inter-annotator agreement (IAA) between each pair of annotators using Cohen's kappa (κ). The pairwise IAA scores were κ=0.80, κ=0.81, and κ=0.82. We then gave each annotator an additional 1,000 tweets to annotate, yielding a total of 3,200 annotated tweets. We used the first 200 tweets as our Tuning Set, and the remaining 3000 tweets as our Test Set.

Our annotators judged 742 of the 3,200 tweets (23%) to be sarcastic. Only 713 of the 1,600 tweets with sarcasm hashtags (45%) were judged to be sarcastic based on our annotation guidelines. There are several reasons why a tweet with a sarcasm hashtag might not have been judged to be sarcastic. Sarcasm may not be apparent without prior conversational context (i.e., multiple tweets), or the sarcastic content may be in a URL and not the tweet itself, or the tweet's content may not obviously be sarcastic without seeing the sarcasm hashtag (e.g., *"The most boring hockey game ever #sarcasm"*).

Of the 1,600 tweets in our data set that were obtained from the random stream and did not have a sarcasm hashtag, 29 (1.8%) were judged to be sarcastic based on our annotation guidelines.

4.2 Baselines

Overall, 693 of the 3,000 tweets in our Test Set were annotated as sarcastic, so a system that classifies every tweet as sarcastic will have 23% precision. To assess the difficulty of recognizing the sarcastic tweets in our data set, we evaluated a variety of baseline systems.

We created two baseline systems that use n-gram features with supervised machine learning to create a sarcasm classifier. We used the LIBSVM (Chang and Lin, 2011) library to train two support vector machine (SVM) classifiers: one with just unigram features and one with both unigrams and bigrams. The features had binary values indicating the presence or absence of each n-gram in a tweet. The classifiers were evaluated using 10-fold cross-validation. We used the RBF kernel, and the cost and gamma parameters were optimized for accuracy using unigram features and 10-fold cross-validation on our Tuning Set. The first two rows of Table 2 show the results for these SVM classifiers, which achieved F scores of 46-48%.

We also conducted experiments with existing sentiment and subjectivity lexicons to see whether they could be leveraged to recognize sarcasm. We experimented with three resources:

Liu05 : A positive and negative opinion lexicon from (Liu et al., 2005). This lexicon contains 2,007 positive sentiment words and 4,783 negative sentiment words.

MPQA05 : The MPQA Subjectivity Lexicon that is part of the OpinionFinder system (Wilson et al., 2005a; Wilson et al., 2005b). This lexicon contains 2,718 subjective words with positive polarity and 4,910 subjective words with negative polarity.

AFINN11 The AFINN sentiment lexicon designed for microblogs (Nielsen, 2011; Hansen et al., 2011) contains 2,477 manually labeled words and phrases with integer values ranging from -5 (negativity) to 5 (positivity). We considered all words with negative values to have negative polarity (1598 words), and all words with positive values to have positive polarity (879 words).

We performed four sets of experiments with each resource to see how beneficial existing sentiment

System	Recall	Precision	F score
Supervised SVM Classifiers			
1grams	.35	.64	.46
1+2grams	.39	.64	.48
Positive Sentiment Only			
Liu05	.77	.34	.47
MPQA05	**.78**	.30	.43
AFINN11	.75	.32	.44
Negative Sentiment Only			
Liu05	.26	.23	.24
MPQA05	.34	.24	.28
AFINN11	.24	.22	.23
Positive and Negative Sentiment, Unordered			
Liu05	.19	.37	.25
MPQA05	.27	.30	.29
AFINN11	.17	.30	.22
Positive and Negative Sentiment, Ordered			
Liu05	.09	.40	.14
MPQA05	.13	.30	.18
AFINN11	.09	.35	.14
Our Bootstrapped Lexicons			
Positive VPs	.28	.45	.35
Negative Situations	.29	.38	.33
Contrast(+VPs, –Situations), Unordered	.11	.56	.18
Contrast(+VPs, –Situations), Ordered	.09	**.70**	.15
& Contrast(+Preds, –Situations)	.13	.63	.22
Our Bootstrapped Lexicons ∪ SVM Classifier			
Contrast(+VPs, –Situations), Ordered	.42	.63	.50
& Contrast(+Preds, –Situations)	.44	.62	**.51**

Table 2: Experimental results on the test set

lexicons could be for sarcasm recognition in tweets. Since our hypothesis is that sarcasm often arises from the contrast between something positive and something negative, we systematically evaluated the positive and negative phrases individually, jointly, and jointly in a specific order (a positive phrase *followed by* a negative phrase).

First, we labeled a tweet as sarcastic if it contains any positive term in each resource. The *Positive Sentiment Only* section of Table 2 shows that all three sentiment lexicons achieved high recall (75-78%) but low precision (30-34%). Second, we labeled a tweet as sarcastic if it contains any negative term from each resource. The *Negative Sentiment Only* section of Table 2 shows that this approach yields much lower recall and also lower precision of 22-24%, which is what would be expected of a random classifier since 23% of the tweets are sarcastic. These results suggest that explicit negative sentiments are not generally indicative of sarcasm.

Third, we labeled a tweet as sarcastic if it contains both a positive sentiment term and a negative sentiment term, in any order. The *Positive and Negative Sentiment, Unordered* section of Table 2 shows that this approach yields low recall, indicating that relatively few sarcastic tweets contain both positive and negative sentiments, and low precision as well.

Fourth, we required the contrasting sentiments to occur in a specific order (the positive term must precede the negative term) and near each other (no more than 5 words apart). This criteria reflects our observation that positive sentiments often closely precede negative situations in sarcastic tweets, so we wanted to see if the same ordering tendency holds for negative sentiments. The *Positive and Negative Sentiment, Ordered* section of Table 2 shows that this ordering constraint further decreases recall and only slightly improves precision, if at all. Our hypothe-

sis is that when positive and negative sentiments are expressed in the same tweet, they are referring to different things (e.g., different aspects of a product). Expressing positive and negative sentiments about the same thing would usually sound contradictory rather than sarcastic.

4.3 Evaluation of Bootstrapped Phrase Lists

The next set of experiments evaluates the effectiveness of the positive sentiment and negative situation phrases learned by our bootstrapping algorithm. The results are shown in the *Our Bootstrapped Lexicons* section of Table 2. For the sake of comparison with other sentiment resources, we first evaluated our positive sentiment verb phrases and negative situation phrases independently. Our positive verb phrases achieved much lower recall than the positive sentiment phrases in the other resources, but they had higher precision (45%). The low recall is undoubtedly because our bootstrapped lexicon is small and contains only verb phrases, while the other resources are much larger and contain terms with additional parts-of-speech, such as adjectives and nouns.

Despite its relatively small size, our list of negative situation phrases achieved 29% recall, which is comparable to the negative sentiments, but higher precision (38%).

Next, we classified a tweet as sarcastic if it contains both a positive verb phrase and a negative situation phrase from our bootstrapped lists, in any order. This approach produced low recall (11%) but higher precision (56%) than the sentiment lexicons. Finally, we enforced an ordering constraint so a tweet is labeled as sarcastic only if it contains a positive verb phrase that precedes a negative situation in close proximity (no more than 5 words apart). This ordering constraint further increased precision from 56% to 70%, with a decrease of only 2 points in recall. This precision gain supports our claim that this particular structure (positive verb phrase followed by a negative situation) is strongly indicative of sarcasm. Note that the same ordering constraint applied to a positive verb phrase followed by a negative *sentiment* produced much lower precision (at best 40% precision using the Liu05 lexicon). Contrasting a positive sentiment with a negative *situation* seems to be a key element of sarcasm.

In the last experiment, we added the positive predicative expressions and also labeled a tweet as sarcastic if a positive predicative appeared in close proximity to (within 5 words of) a negative situation. The positive predicatives improved recall to 13%, but decreased precision to 63%, which is comparable to the SVM classifiers.

4.4 A Hybrid Approach

Thus far, we have used the bootstrapped lexicons to recognize sarcasm by looking for phrases in our lists. We will refer to our approach as the Contrast method, which labels a tweet as sarcastic if it contains a positive sentiment phrase in close proximity to a negative situation phrase.

The Contrast method achieved 63% precision but with low recall (13%). The SVM classifier with unigram and bigram features achieved 64% precision with 39% recall. Since neither approach has high recall, we decided to see whether they are complementary and the Contrast method is finding sarcastic tweets that the SVM classifier overlooks.

In this hybrid approach, a tweet is labeled as sarcastic if either the SVM classifier or the Contrast method identifies it as sarcastic. This approach improves recall from 39% to 42% using the Contrast method with only positive verb phrases. Recall improves to 44% using the Contrast method with both positive verb phrases and predicative phrases. This hybrid approach has only a slight drop in precision, yielding an F score of 51%. This result shows that our bootstrapped phrase lists are recognizing sarcastic tweets that the SVM classifier misses.

Finally, we ran tests to see if the performance of the hybrid approach (Contrast $\cup$ SVM) is statistically significantly better than the performance of the SVM classifier alone. We used paired bootstrap significance testing as described in Berg-Kirkpatrick et al. (2012) by drawing 10^6 samples with repetition from the test set. These results showed that the Contrast $\cup$ SVM system is statistically significantly better than the SVM classifier at the $p < .01$ level (i.e., the null hypothesis was rejected with 99% confidence).

4.5 Analysis

To get a better sense of the strength and limitations of our approach, we manually inspected some of the

tweets that were labeled as sarcastic using our bootstrapped phrase lists. Table 3 shows some of the sarcastic tweets found by the Contrast method but not by the SVM classifier.

i love *fighting* with the one i love
love *working* on my last day of summer
i enjoy tweeting [user] and *not getting* a reply
working during vacation is awesome .
can't wait *to wake* up early to babysit !

Table 3: Five sarcastic tweets found by the Contrast method but not the SVM

These tweets are good examples of a positive sentiment (love, enjoy, awesome, can't wait) contrasting with a negative situation. However, the negative situation phrases are not always as specific as they should be. For example, "working" was learned as a negative situation phrase because it is often negative when it follows a positive sentiment ("I love working..."). But the attached prepositional phrases ("on my last day of summer" and "during vacation") should ideally have been captured as well.

We also examined tweets that were incorrectly labeled as sarcastic by the Contrast method. Some false hits come from situations that are frequently negative but not always negative (e.g., some people genuinely like waking up early). However, most false hits were due to overly general negative situation phrases (e.g., "I love *working* there" was labeled as sarcastic). We believe that an important direction for future work will be to learn longer phrases that represent more specific situations.

5 Conclusions

Sarcasm is a complex and rich linguistic phenomenon. Our work identifies just one type of sarcasm that is common in tweets: contrast between a positive sentiment and negative situation. We presented a bootstrapped learning method to acquire lists of positive sentiment phrases and negative activities and states, and show that these lists can be used to recognize sarcastic tweets.

This work has only scratched the surface of possibilities for identifying sarcasm arising from positive/negative contrast. The phrases that we learned were limited to specific syntactic structures and we required the contrasting phrases to appear in a highly constrained context. We plan to explore methods for allowing more flexibility and for learning additional types of phrases and contrasting structures.

We also would like to explore new ways to identify stereotypically negative activities and states because we believe this type of world knowledge is essential to recognize many instances of sarcasm. For example, sarcasm often arises from a description of a negative event followed by a positive emotion but in a separate clause or sentence, such as: *"Going to the dentist for a root canal this afternoon. Yay, I can't wait."* Recognizing the intensity of the negativity may also be useful to distinguish strong contrast from weak contrast. Having knowledge about stereotypically undesirable activities and states could also be important for other natural language understanding tasks, such as text summarization and narrative plot analysis.

6 Acknowledgments

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI / NBC) contract number D12PC00285. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBE, or the U.S. Government.

References

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 995–1005.

Paula Carvalho, Luís Sarmento, Mário J. Silva, and Eugénio de Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's "so easy" ;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, TSA 2009.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transac-*

tions on Intelligent Systems and Technology, 2:27:1–27:27.

Henry S. Cheang and Marc D. Pell. 2008. The sound of sarcasm. *Speech Commun.*, 50(5):366–381, May.

Henry S. Cheang and Marc D. Pell. 2009. Acoustic markers of sarcasm in cantonese and english. *The Journal of the Acoustical Society of America*, 126(3):1394–1405.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL 2010.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*.

Roger J. Kreuz Gina M. Caucci. 2012. Social and paralinguistic cues to sarcasm. *online 08/02/2012*, 25:1–22, February.

Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: A closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Lars Kai Hansen, Adam Arvidsson, Finn Arup Nielsen, Elanor Colleoni, and Michael Etter. 2011. Good friends, bad news - affect and virality in twitter. In *The 2011 International Workshop on Social Computing, Network, and Services (SocialComNet 2011)*.

Roger Kreuz and Gina Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*.

Christine Liebrecht, Florian Kunneman, and Antal Van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, WASSA 2013.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International World Wide Web conference (WWW-2005)*.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*.

Finn Arup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages (http://arxiv.org/abs/1103.2903)*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *The 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2013)*.

Katherine P. Rankin, Andrea Salazar, Maria Luisa Gorno-Tempini, Marc Sollberger, Stephen M. Wilson, Danijela Pavlic, Christine M. Stanley, Shenly Glenn, Michael W. Weiner, and Bruce L. Miller. 2009. Detecting sarcasm from paralinguistic cues: Anatomic and cognitive correlates in neurodegenerative disease. *Neuroimage*, 47:2005–2015.

Joseph Tepperman, David Traum, and Shrikanth Narayanan. 2006. "Yeah right": Sarcasm recognition for spoken dialogue systems. In *Proceedings of the INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing*.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. ICWSM - A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews. In *Proceedings of the Fourth International Conference on Weblogs and Social Media (ICWSM-2010)*, ICWSM 2010.

Michael Wiegand, Josef Ruppenhofer, and Dietrich Klakow. 2013. Predicative adjectives: An unsupervised criterion to extract subjective adjectives. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, Atlanta, Georgia, June. Association for Computational Linguistics.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. 2005a. OpinionFinder: A System for Subjectivity Analysis. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 34–35, Vancouver, Canada, October.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005b. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the 2005 Human Language Technology Conference / Conference on Empirical Methods in Natural Language Processing*.

Collective Personal Profile Summarization with Social Networks

Zhongqing Wang, Shoushan Li*, Kong Fang, and Guodong Zhou
Natural Language Processing Lab, School of Computer Science and Technology
Soochow University, Suzhou, 215006, China
{wangzq.antony, shoushan.li}@gmail.com,
{kongfang, gdzhou}@suda.edu.cn

Abstract

Personal profile information on social media like LinkedIn.com and Facebook.com is at the core of many interesting applications, such as talent recommendation and contextual advertising. However, personal profiles usually lack organization confronted with the large amount of available information. Therefore, it is always a challenge for people to find desired information from them. In this paper, we address the task of personal profile summarization by leveraging both personal profile textual information and social networks. Here, using social networks is motivated by the intuition that, people with similar academic, business or social connections (e.g. *co-major*, *co-university*, and *co-corporation*) tend to have similar experience and summaries. To achieve the learning process, we propose a collective factor graph (CoFG) model to incorporate all these resources of knowledge to summarize personal profiles with local textual attribute functions and social connection factors. Extensive evaluation on a large-scale dataset from LinkedIn.com demonstrates the effectiveness of the proposed approach.

1 Introduction

Web 2.0 has empowered people to actively interact with each other, forming social networks around mutually interesting information and publishing a large amount of useful user-generated content (UGC) online (Lappas et al., 2011; Tan et al., 2011). One popular and important type of UGC is the personal profile, where people post detailed information on online portals about their education, experiences and other personal information. Social websites like Facebook.com and LinkedIn.com have created a viable business as profile portals, with the popularity and success partially attributed to their comprehensive personal profiles.

Generally, online personal profiles provide valuable resources for businesses, especially for human resource managers to find talents, and help people connect with others of similar backgrounds (Yang et al., 2011a; Guy et al., 2010). However, as there is always large-scale information of experience and education fields, it is hardly for us to find useful information from the profile. Therefore, it is always a challenge for people to find desired information from them. For this regard, it is highly desirable to develop reliable methods to generate a summary of a person through his profile automatically.

To the best of our knowledge, this is the first research that explores automatic summarization of personal profiles in social media. A straightforward approach is to consider personal profile summarization as a traditional document summarization problem, which treating each personal profile independently and generate a summary for each personal profile individually. For example, the well-known extraction and ranking approaches (e.g. PageRank, HITS) extract a certain amount of important sentences from a document according to some ranking measurements to form a summary (Wan and Yang, 2008; Wan, 2011).

However, such straightforward approaches are not sufficient to benefit from the carrier of personal profiles. As the centroid of social networking, people are usually connected to others with similar

* Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 715–725,
Seattle, Washington, USA, 18-21 October 2013.

background in social media (e.g. *co-major*, *co-corporation*). Therefore, it is reasonable to leverage social connection to improve the performance of profile summarizing. For example if there are *co-major*, *co-university*, *co-corporation* or other academic and business relationships between two persons, we consider them sharing similar experience and having similar summaries.

The remaining challenge is how to incorporate both the profile textual information and the connection knowledge in the social networks. In this study, we propose a collective factor graph model (CoFG) to summarize the text of personal profile in social networks with local textual information and social connection information. The CoFG framework utilizes both the local textual attribute functions of an individual person and the social connection factor between different persons to collectively summarize personal profile on one person.

In this study, we treat the profile summarization as a supervised learning task. Specifically, we model each sentence of the profile as a vector. In the training phase, we use the vectors with the social connection between each person to build the CoFG model; while in the testing phase, we perform collective inference for the importance of each sentence and select a subset of sentences as the summary according to the trained model. Evaluation on a large-scale data from LinkedIn.com indicates that our proposed joint model and social connection information improve the performance of profile summarization.

The remainder of our paper is structured as follows. We go over the related work in Section 2. In Section 3, we introduce the data we collected from LinkedIn.com and the annotated corpus we constructed. In Section 4, we present some motivational analysis. In Section 5, we explain our proposed model and describe algorithms for parameter estimation and prediction. In Section 6, we present our experimental results. We sum up our work and discuss future directions in Section 7.

2 Related Work

In this section, we will introduce the related work on the traditional topic-based summarization, social-based summarization and factor graph model respectively.

2.1 Topic-based Summarization

Generally, traditional topic-based summarization can be categorized into two categories: extractive (Radev et al., 2004) and abstractive (Radev and McKeown, 1998) summarization. The former selects a subset of sentences from original document(s) to form a summary; the latter reorganizes some sentences to form a summary where several complex technologies, such as information fusion, sentence compression and reformulation are necessarily employed (Wan and Yang, 2008; Celikyilmaz and Hakkani-Tur, 2011; Wang and Zhou, 2012). This study focuses on extractive summarization.

Radev et al. (2004) proposed a centroid-based method to rank the sentences in a document set, using various kinds of features, such as the cluster centroid, position and TF-IDF features. Ryang and Abekawa (2012) proposed a reinforcement learning approach on text summarization, which models the summarization within a reinforcement learning-based framework.

Compared to unsupervised approaches, supervised learning for summarization is relatively rare. A typical work is Shen et al., (2007) which present a Conditional Random Fields (CRF) based framework to treat the summarization task as a sequence labeling problem. However, different from all existing studies, our work is the first attempt to consider both textual information and social relationship information for supervised summarization.

2.2 Social-based Summarization

As web 2.0 has empowered people to actively interact with each other, studies focusing on social media have attracted much attention recently (Meeder et al., 2011; Rosenthal and McKeown, 2011; Yang et al., 2011a). Social-based summarization is exactly a special case of summarization where the social connection is employed to help obtaining the summarization. Although topic-based summarization has been extensively studied, studies on social-based summarization are relative new and rare.

Hu et al., (2011) proposed an unsupervised PageRank-based social summarization approach by incorporating both document context and user context in the sentence evaluation process. Meng et al., (2012) proposed a unified optimization framework to produce opinion summaries of tweets through

integrating information from dimensions of topic, opinion and insight, as well as other factors (e.g. topic relevancy, redundancy and language styles).

Unlike all the above studies, this paper focuses on a novel task, profile summarization. Furthermore, we employ many other kinds of social information in profiles, such as *co-major*, and *co-corporation* between two people. They are shown to be very effective for profile summarization.

2.3 Factor Graph Model

As social network has been investigated for several years (Leskovec et al., 2010; Tan et al., 2011; Lu et al., 2010; Guy et al., 2010) and Factor Graph Model (FGM) is a popular approach to describe the relationship of social network (Tang et al., 2011a; Zhuang et al., 2012). Factor Graph Model builds a graph to represent the relationship of nodes on the social networks, and the factor functions are always considered to represent the relationship of the nodes.

Tang et al. (2011a) and Zhuang et al. (2012) formalized the problem of social relationship learning into a semi-supervised framework, and proposed Partially-labeled Pairwise Factor Graph Model (PLP-FGM) for learning to infer the type of social ties. Dong et al. (2012) gave a formal definition of link recommendation across heterogeneous networks, and proposed a ranking factor graph model (RFG) for predicting links in social networks, which effectively improves the predictive performance. Yang et al., (2011b) generated summaries by modeling tweets and social contexts into a dual wing factor graph (DWFG), which utilized the mutual reinforcement between Web documents and their associated social contexts.

Different from all above researches, this paper proposes a pair-wise factor graph model to collectively utilize both textual information and social connection factor to generate summary of profile.

3 Data Collection and Statistics

The personal profile summarization is a novel task and there exists no related data for accessing this issue. Therefore, in this study, we collect a data set containing personal summaries with the corresponding knowledge, such as the self-introduction and personal profiles. In this section, we will introduce this data set in detail.

3.1 Data Collection

We collect our data set from LinkedIn.com[1]. It contains a large number of personal profiles generated by users, containing various kinds of information, such as personal overview, summary, education, experience, projects and skills.

John Smith[2]	
Overview	
Current	Applied Researcher at Apple Inc.
Previous	Senior Research Scientist at IBM …
Education	MIT, Georgia Institute of Technology, …
Summary	
Machine learning researcher and engineer on many fields: Query understanding. Automatic Information extraction…	
Experience	
Applied Researcher *Apple Inc.,* September 2012 ~ Query recognition and relevance …	
Education	
MIT *Ph.D., Electrical Engineering*, 2002 – 2008 …	

Figure 1: An example of a profile webpage from LinkedIn.com

In this study, the data set is crawled in the following ways. To begin with, 10 random people's public profiles are selected as seed profiles, and then the profiles from their "People Also Viewed" field were collected. The data is composed of 3,182 public profiles[3] in total. We do not collect personal names in public profiles to protect people's privacy. Figure 1 shows an example of a person's profile from LinkedIn.com. The profile includes following fields:

- *Overview*: It gives a structure description of a person's general information, such as current/previous position and workplace, brief

[1] http://www.linkedin.com

[2] The information of the example is a pseudo one.

[3] We collect all the data from LinkedIn.com at Dec 17, 2012.

education background and general technical background.

- *Summary*: It summarizes a person's work, experience and education.
- *Experience*: It details a person's work experience.
- *Education*: It details a person's education background.

Among these fields, the *Overview* is required and the others are optional, such as *Project*, *Course* and *Interest groups*. However, compared with *Overview*, *Summary*, *Experience*, *Education* fields, they seem to be less important for summarization of personal profiles. Thus, we ignore them in our study.

3.2 Data Statistics of Major Fields

We collected 3,182 personal profiles from LinkedIn.com. Table 1 shows the statistics of major fields in our data collection.

Field	#Non-empty fields	Average field length
Overview	3,182	45.1
Summary	921	25.8
Experience	3,148	192.1
Education	2,932	33.6

Table 1: Statistics of major fields in our data set, i.e. the number of non-empty fields and the average length for each field

From Table 1, we can see that,

- The information of each profile is incomplete and inconsistent, That is, not all kinds of fields are available in each personal's profile.
- Most people provide their experience and education information. However, the *Summary* fields are popularly missing (Only about 30% of people provide it). This is mainly because writing summary is normally more difficult than other fields. Therefore, it is highly desirable to develop reliable automatic methods to generate a summary of a person through his/her profile.
- The length of the *Experience* field is the longest one, and work experience always could represent general information of people.

3.3 Corpus Construction and Annotation

Among the 921 profiles that contain the summary, we manually select 497 profiles with high quality summary to construct the corpus for our research. These high-quality summaries are all written by the authors themselves. Here, the quality is measured by manually checking that whether they are well capable of summarizing their profiles. That is, they are written carefully, and could give an overview of a person and represent the education and experience information of a person.

After carefully seeing the profiles, we observe that the *Experience* field contains the most abundant information of a person. Thus, we treat the text of *Experience* field as the source of summary for each profile. Besides, we collect social context information from *Education* and *Experience* field, and these social contexts are including by LinkedIn explicitly. Table 2 shows the average length of summary and experience fields we used for evaluating our summarization approach.

Field	Average length
Summary (*the summary of the profile*)	37.2
Experience (*the source text for the summarizing*)	372.0

Table 2: Average length of the high-quality summary and corresponding experience fields

From Table 2, we can see that,

- Compared with the average length of 25.8 in Table 1, summaries of high quality have longer length because they contain more information of the profiles.
- The compression ratio of our proposed corpus is 0.1 (37.2/372.0).

4 Motivation and Analysis

In this section, we propose the motivation of social connection to address the task of personal profile summarization. To preliminarily support the motivation, some statistics of the social connection are provided.

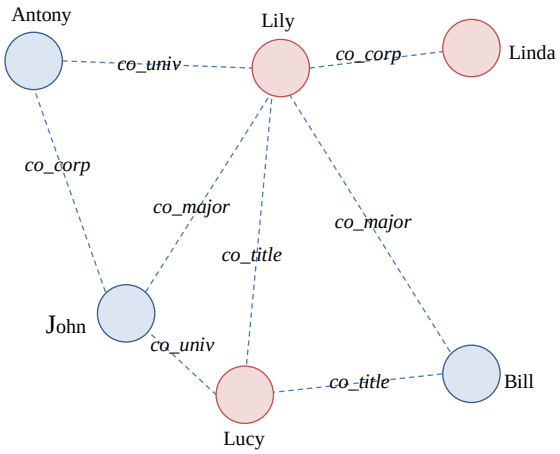


Figure 2: An example of personal profile network. *Red is for female, blue is for male, and the dotted line means the social connection between two persons.*

We first describe the social connections which we used. Figure 2 shows an example of social connection between people from the profiles of LinkedIn. We find that people are sometimes connected by several social connections. For example, John and Lucy are connected by *co_unvi* relationship, while Lily and Linda are connected by *co_corp* relationship. From LinkedIn, four kinds of social relationship between people are extracted from the *Education* field and *Experience* field. They are:

- *co_major* denotes that two persons have the same major at school
- *co_univ* denotes that two persons are graduated from the same university
- *co_title* denotes that two persons have the same title at corporation.
- *co_corp* denotes that two persons work at the same corporation.

Our basic motivation of using social connection lies in the fact that "connected" people will tend to hold related experience and similar summaries.

We then give the statistics of edges of social connection. Table 3 shows basic statistics across these edges. From Table 3, we can see that the number of users is 497 while the number of social connection edges is 14,307. The latter is much larger than the former. The number of the edges from *Education* field is similar with the number of the edges from *Experience* filed. Among all the relationships, *co_unvi* is the most common one.

	Numbers
# users	**497**
co_major	1,288
co_unvi	6,015
# education field	**7,303**
co_title	3,228
co_corp	3,776
# experience field	**7,004**
# total edges	**14,307**

Table 3: The statistic of edges for our main datasets

5 Collective Factor Graph Model

In this section, we propose a collective factor graph (CoFG) model for learning and summarizing the text of personal profile with local textual information and social connection.

5.1 Overview of Our Framework

To generate summaries for profiles, a straightforward approach is to treat each personal profile independently and generating a summary for each personal profile individually. As we mentioned on Section 3.3, we use the sentences of *Experience* field as a text document and consider it as the source of summary for each profile.

Instead, we formalize the problem of personal profile summarization in a pair-wise factor graph model and propose an approach referred to as Loopy Belief Propagation algorithm to learn the model for generating the summary of the profile. Our basic idea is to define the correlations using different types of factor functions. An objective function is defined based on the joint probability of the factor functions. Thus, the problem of collective personal profile summarization model learning is cast as learning model parameters that maximizes the joint probability of the input continuous dynamic network.

The overview of the proposed method is a supervised framework (as shown in Figure 3). First, we treat each sentence of the training data and testing data as vectors with textual information (local textual attribute functions); Second, all the vectors are connected by social connection relationships (social connection factors) and we model these vectors and their relationships into the collective factor graph; third, we propose Loopy Belief Prop-

agation algorithm to learn the model and predict the sentences of testing data; finally, we select a subset of sentences of each testing profile as the summary according to the models with top-*n* prediction score. Thus, the core issues of our framework are 1) how to define the collective factor graph model to connection profiles with social connection; 2) how to learn and predict the proposed CoFG model; 3) how to predict the sentences from the testing data with the proposed CoFG model, and generate the summary by the predict scores. We will discuss these issues on the following subsections.

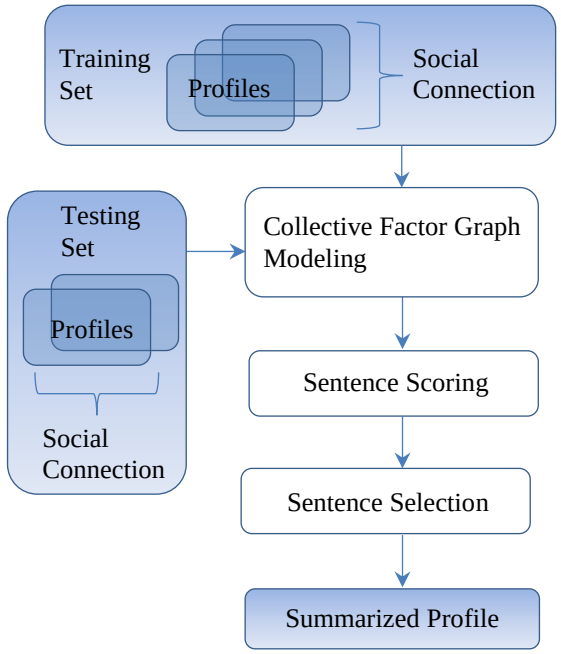


Figure 3: The overview of our proposed framework

5.2 Model Definition

Formally, given a network $G=(V,S^L,S^U,X)$, each sentence s_i is associated with an attribute vector x_i of the profile and a label y_i indicating whether the sentence is selected as a summary of the profile (The value of y_i is binary. 1 means that the sentence is selected as a summary sentence, whereas 0 stands for the opposite). V denotes the authors of the profiles, S^L denotes the labeled training data, and S^U denotes the unlabeled testing data. Let $X=\{x_i\}$ and $Y=\{y_i\}$. Then, we have the following formulation

$$P(Y \mid X,G)=\frac{P(X,G \mid Y)P(Y)}{P(X,G)} \quad (1)$$

Here, G denotes all forms of network information. This probabilistic formulation indicates that labels of skills depend on not only local attributes X, but also the structure of the network G. According to Bayes' rule, we have

$$P(Y \mid X,G)=\frac{P(X,G \mid Y)P(Y)}{P(X,G)} \propto P(X \mid Y)P(Y \mid G) \quad (2)$$

Where $P(Y \mid G)$ represents the probability of labels given the structure of the network and $P(X \mid Y)$ denotes the probability of generating attributes X associated to their labels Y. We assume that the generative probability of attributes given the label of each edge is conditionally independent, thus we have

$$P(Y \mid X,G) \propto P(Y \mid G)\prod_i P(x_i \mid y_i) \quad (3)$$

Where $P(x_i \mid y_i)$ is the probability of generating attributes x_i given the label y_i. Now, the problem becomes how to instantiate the probability $P(Y \mid G)$ and $P(x_i \mid y_i)$. We model them in a Markov random field, and thus according to the Hammersley-Clifford theorem (Hammersley and Clifford, 1971), the two probabilities can be instantiated as follows:

$$P(x_i \mid y_i)=\frac{1}{Z_1}\exp\left\{\sum_{j=1}^{d}\alpha_j f_j(x_{ij},y_i)\right\} \quad (4)$$

$$P(Y \mid G)=\frac{1}{Z_2}\exp\left\{\sum_i \sum_{j\in NB(i)} g(i,j)\right\} \quad (5)$$

Where Z_1 and Z_2 are normalization factors. Eq. 4 indicates that we define an attribute function $f(x_i,y_i)$ for each attribute x_{ij} associated with sentence s_i. α_j is the weight of the j^{th} attribute. Eq. 5 represents that we define a set of correlation factor functions $g(i,j)$ over each pair (i,j) in the network. $NB(i)$ denotes the set of social relationship neighbors nodes of i.

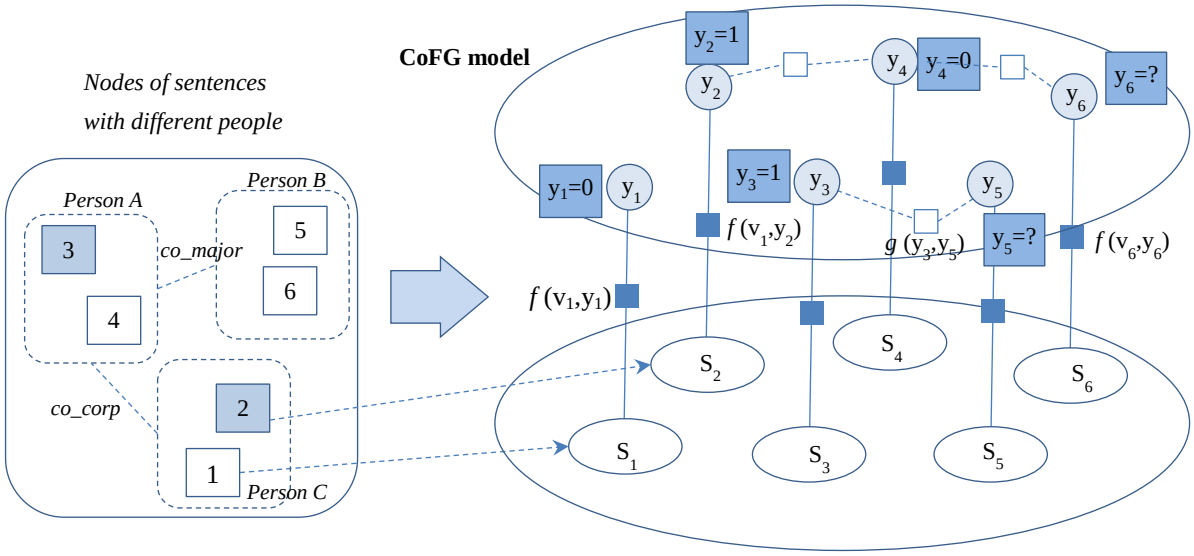


Figure 4: Graph representation of CoFG

The left figure shows the personal profile network. Each dotted line indicates a social connection. Each dotted square denotes a person, and the grey square denotes the sentence selected in the summary, and the white square denotes a sentence that is not selected as the summary..

The right figure shows the CoFG model derived from left figure. Each eclipse denotes a sentence vector of a person, and each circle indicates the hidden variable y_i. $f(v_i,y_i)$ indicates the attribute factor function. $g(y_i,y_j)$ indicates the social connection factor function.

We now briefly introduce possible ways to define the attribute functions $\{f(x_{ij}, y_i)\}_j$, and factor function $g(i, j)$.

Local textual attribute functions $\{f(x_{ij}, y_i)\}_j$: It denotes the attribute value associated with each sentence *i*. We define the local textual attribute as a feature (Lafferty et al., 2001). We can accumulate all the attribute functions and obtain local entropy for a person:

$$\frac{1}{Z_1}\exp\left(\sum_i\sum_k \alpha_k f_k(x_{ik}, y_i)\right) \qquad (6)$$

The textual attributes include following features (Shen et al., 2007; Yang et al., 2011b):

1) *BOW*: the bag-of-words of each sentence, we use unigram features as the basic textual features for each sentence.
2) *Length*: the number of terms of each sentence.
3) *Topic_words*: these are the most frequent words in the sentence after the stop words are removed.
4) *PageRank_scores*: as shown in the related work section, a document can be treated as a graph and applying a graph-based ranking algorithm (Wan and Yang., 2008). We thus use the PageRank score to reflect the importance of each sentence.

Social connection factor function $g(y_i, y_j)$: For the social correlation factor function, we define it through the pairwise network structure. That is, if the person of sentence *i* and the person of sentence *j* have a social relationship, a factor function for this social connection is defined (Tang et al., 2011a; Tang et al., 2011b), i.e.,

$$g(y_i, y_j) = \exp\left\{\beta_{ij}(y_i - y_j)^2\right\} \qquad (7)$$

The person-person social relationships are defined on Section 4, e.g. *co_major*, *co_univ*, *co_title*, and *co_corp*. We define that if two persons have at least one social connection edge, they have a social relationship. In addition, β_{ij} is the weight of the function, representing the influence degree of *i* on *j*.

To better understand our model, one example of factor decomposition is given in Figure 4. In this example, there are six sentences from three profiles. Among them, four sentences are labeled (two are labeled with the category of "1", i.e, $y=1$ and the other two are labeled with the category of "0', i.e., $y=0$) and two sentences are unlabeled (they are represented by y=?). We have six attribute functions. For example, $f(v_1, y_i)$ denotes the set

of local textual attribute functions of y_i. We also have five pairwise relationships (e.g., (y_2, y_4), (y_3, y_5)) based on the structure of the input personal profile social network. For example, $g(y_3, y_5)$ denotes social connection between y_3 and y_5, while they share the *co_major* relationship on the left figure.

5.3 Model Learning

We now address the problem of estimating the free parameters. The objective of learning the CoFG model is to estimate a parameter configuration $\theta = (\{\alpha\}, \{\beta\})$ to maximize the log-likelihood objective function $L(\theta) = \log P_\theta(Y \mid X, G)$, i.e.,

$$\theta^* = \arg\max L(\theta) \quad (9)$$

To solve the objective function, we adopt a gradient descent method. We use β (the weight of the social connection factor function $g(y_i, y_j)$) as the example to explain how we learn the parameters (the algorithm also applies to tune α by simply replacing β with α). Specifically, we first write the gradient of each β_k with regard to the objective function (Eq. 9) :

$$\frac{L(\theta)}{\beta_k} = E\left[g(i,j)\right] + E_{P_{\beta k}(Y|X,G)}\left[g(i,j)\right] \quad (10)$$

Where $E[g(i,j)]$ is the expectation of factor function $g(i,j)$ given the data distribution (essentially it can be considered as the average value of the factor function $g(i,j)$ over all pair in the training data); and $E_{P_{\beta k}(Y|X,G)}[g(i,j)]$ is the expectation of factor function $g(i,j)$ under the distribution $P_{\beta k}(Y \mid X, G)$ given by the estimated model. A similar gradient can be derived for parameter α_j.

We approximate the marginal distribution $E_{P_{\beta k}(Y|X,G)}[g(i,j)]$ using LBP (Tang et al., 2011; Zhuang et al., 2012). With the marginal probabilities, the gradient can be obtained by summing over all triads. It is worth noting that we need to perform the LBP process twice for each iteration: one is to estimate the marginal distribution of unknown variables $y_i = ?$ and the other is to estimate the marginal distribution over all pairs. In this way, the algorithm essentially performs a transfer learning over the complete network. Finally, with the obtained gradient, we update each parameter with a learning rate η. The learning algorithm is summarized in Figure 5.

Input: Network G, Learning rate η
Output: Estimated parameters θ
Initialize $\theta \leftarrow 0$
Repeat
1) Perform LBP to calculate the marginal distribution of unknown variables, i.e., $P(y_i \mid x_i, G)$
2) Perform LBP to calculate the marginal distribution of each variables, i.e., $P(y_i, y_j \mid X_{(i,j)}, G)$
3) Calculate the gradient of β_k according to Eq. 10 (for α with a similar formula)
4) Update parameter θ with the learning rate η

$$\theta_{\text{new}} = \theta_{\text{old}} + \eta \frac{L(\theta)}{\theta}$$

Until *Convergence*

Figure 5: The Learning Algorithm for CoFG model

5.4 Model Prediction and Summary Generated

We can see that in the learning process, the learning algorithm uses an additional loopy belief propagation to infer the label of unknown relationships. With the estimated parameter θ, the summarization process is to find the most likely configuration of Y for a given profile. This can be obtained by

$$Y^* = \arg\max L(Y \mid X, G, \theta) \quad (11)$$

Finally, we select a subset of sentences of each testing profile as the summary according to the trained models with top-*n* prediction scores by Y^* (Tang et al., 2011b; Dong et al, 2012).

6 Experimentation

In this section, we describe the settings of our experiment and present the experimental results of our proposed CoFG model.

	ROUGE-2	ROUGE-L	ROUGE-W	ROUGE-SU4
Random	0.0219	0.1363	0.0831	0.0288
HITS	0.0295	0.1499	0.0905	0.0355
PageRank	0.0307	0.1574	0.0944	0.0383
MaxEnt	0.0349	0.1659	0.0995	0.0377
CoFG	0.0383	0.1696	0.1015	0.0415

Table 4: Performances of different approaches to profile summarization in terms of different measurements

6.1 Experiment Settings

In the experiment, we use the corpus collected from LinkedIn.com that contains 497 profiles (see more details in Section 3). The existing summaries in these profiles are served as the reference summary (the standard answers). As discussed in subsection 3.3, the average length of summary is about 40 words. Thus, we extract 40 words to construct the summary for each profile. We use 200 personal profiles as the testing data, and the remaining ones as the training data.

We use the ROUGE-1.5.5 (Lin and Hovy, 2004) toolkit for evaluation, a popular tool that has been widely adopted by several evaluations such as DUC and TAC (Wan and Yang, 2008; Wan, 2011). We provide four of the ROUGE F-measure scores in the experimental results: ROUGE-2 (bigram-based), ROUGE-L (based on longest common subsequences), ROUGE-W (based on weighted longest common subsequence, weight=1.2), and ROUGE-SU4 (based on skip bigram with a maximum skip distance of 4).

6.2 Experimental Results

We compare the proposed CoFG approach with three baselines illustrated as follows:

- **Random**: we randomly select sentences of each profile to generate the summary for the profile.
- **HITS**: we employ the HITS algorithm to perform profile summarization (Wan and Yang, 2008). In detail, we first consider the words as hubs the sentences as authorities; Then, we rank the sentences with the authorities' scores for each profile individually; Finally, the highest ranked sentences are chosen to constitute the summary.
- **PageRank**: we employ the PageRank algorithm to perform profile summarization (Wan and Yang, 2008). In detail, we first connect the sentences of the profile with cosine text-based similar measure to construct a graph; Then, we apply PageRank algorithm to rank the sentence through the graph for each profile individually; Finally, the highest ranked sentences are chosen to constitute the summary.
- **MaxEnt**: as a supervised learning approach, maximum entropy uses textual attribute as features to train a classification model. Then, the classification model is employed to predict which sentences can be selected to generate the summary. For the implementation of MaxEnt, we employ the tool of *mallent toolkits*[4].

Table 4 shows the comparison results of our approach (CoFG) and the baseline approaches. From Table 4, we can see that 1) either HITS or PageRank outperforms the approach of random selection; 2) The supervised approach i.e. MaxEnt, outperforms both the HITS algorithm and the PageRank approach; 3) CoFG model performs best and it greatly outperforms both the unsupervised and supervised learning baseline approaches in terms of the ROUGE-2 F-measure score. This result verifies the effectiveness of considering the social connection between the sentences in different profiles,

Figure 6 shows the performance of our proposed CoFG model with different sizes of training data. From Figure 6, we can see that CoFG model with social connection always performs better than MaxEnt, and the performance of our approach descends slowly when the training dataset becomes small. Specifically, the performance of CoFG using only 10% training data achieves better performance than MaxEnt using 100% training data.

[4] http://mallet.cs.umass.edu/

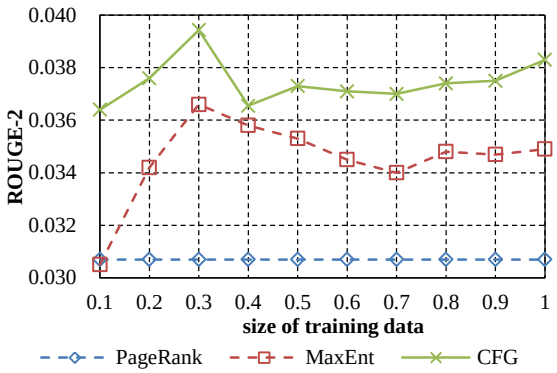


Figure 6: The performance of CoFG with different training data size

Table 5 shows the contribution of the social edges with CoFG. Specifically, CoFG is our proposed approach with both education and experience information, CoFG-*edu* means that the CoFG model considers the social edges of education field (*co_major*, *co_univ*) only, and CoFG-*exp* means that the CoFG model considers the social edges of work experience field (*co_title*, *co_corp*) only. MaxEnt can be considered as using textual information only.

	ROUGE-2
MaxEnt	0.0349
CoFG	0.0383
CoFG-*edu*	0.0382
CoFG-*exp*	0.0381

Table 5: ROUGE-2 F-Measure score of the contribution of social edges

From Table 5, we can see that all of our proposed approaches, i.e., CoFG-*edu*, CoFG-*exp*, and CoFG, outperform the baseline approach, i.e., MaxEnt. However, the performance of CoFG-*edu*, CoFG-*exp* and CoFG are similar. This result is mainly due to the fact that the information of social connection is redundant. For example, two persons who are connected by *co_major* (education field) might also be connected by *co_corp* (experience field).

7 Conclusion and Future Work

In this paper, we present a novel task named profile summarization and propose a novel approach called collective factor graph model to address this task. One distinguishing feature of the proposed approach lies in its incorporating the social connection. Empirical studies demonstrate that the social connection is effective for profile summarization, which enables our approach outperform some competitive supervised and unsupervised baselines.

The main contribution of this paper is to explore social context information to help generate the summary of the profiles, which represents an interesting research direction in social network mining. In the future work, we will explore more kinds of social context information and investigate better ways of incorporating them into profile summarization and a wider range of social network mining.

Acknowledgments

This research work is supported by the National Natural Science Foundation of China (No.61273320, No.61272257, No.61331011 and No.61375073), and National High-tech Research and Development Program of China (No.2012AA011102).

We thank Dr. Jie Tang and Honglei Zhuang for providing their software and useful suggestions about PGM. We acknowledge Dr. Xinfang Liu, Yunxia Xue and Yulai Shen for corpus construction and insightful comments. We also thank anonymous reviewers for their valuable suggestions and comments.

References

Baeza-Yates R. and B. Ribeiro-Neto. 1999. Modern Information Retrieval. *ACM Press and Addison Wesley*, 1999

Celikyilmaz A. and D. Hakkani-Tur. 2011. Discovery of Topically Coherent Sentences for Extractive Summarization. In *Proceeding of ACL-11*.

Dong Y., J. Tang, S. Wu, J. Tian, N. Chawla, J. Rao, and H. Cao. 2012. Link Prediction and Recommendation across Heterogeneous Social Networks. In *Proceedings of ICDM-12*.

Elson D., N. Dames and K. McKeown. 2010. Extracting Social Networks from Literary Fiction. In *Proceeding of ACL-10*.

Erkan G. and D. Radev. 2004. LexPageRank: Prestige in Multi-document Text Summarization. In *Proceedings of EMNLP-04*.

Guy I., N. Zwerdling, I. Ronen, D. Carmel, E. Uziel. 2010. Social Media Recommendation based on People and Tags. In *Proceeding of SIGIR-10*.

Hammersley J. and P. Clifford. 1971. Markov Field on Finite Graphs and Lattices, *Unpublished manuscript*. 1971.

Hu P., C. Sun, L. Wu, D. Ji and C. Teng. 1011. Social Summarization via Automatically Discovered Social Context. In *Proceeding of IJCNLP-11*.

Lafferty J, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML-01*.

Lappas T., K. Punera and T. Sarlos. 2011. Mining Tags Using Social Endorsement Networks. In *Proceeding of SIGIR-11*.

Leskovec J., D. Huttenlocher and J. Kleinberg. 2010. Predicting Positive and Negative Links in Online Social Networks. In *Proceedings of WWW-10*.

Lin, C. 2004. ROUGE: a Package for Automatic Evaluation of Summaries. In *Proceedings of ACL-04 Workshop on Text Summarization Branches Out.*

Lu Y., P. Tsaparas, A. Ntoulas and L. Polanyi. 2010. Exploiting Social Context for Review Quality Prediction. In *Proceeding of WWW-10*.

Meng X，F. Wei， X. Liu， M. Zhou， S. Li and H. Wang. 2012. Entity-Centric Topic-Oriented Opinion Summarization in Twitter. In *Proceeding of KDD-12*.

Murphy K., Y. Weiss, and M. Jordan. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proceedings of UAI-99*.

Radev D. and K. McKeown. 1998. Generating Natural Language Summaries from Multiple On-line Sources. *Computational Linguistics*, 24(3):469–500.

Radev D., H. Jing, M. Stys, and D. Tam. 2004. Centroid-based Summarization of Multiple Documents. *Information Processing and Management*. 40 (2004), 919-938.

Rosenthal S. and K. McKeown. 2011. Age Prediction in Blogs: A Study of Style, Content, and OnlineBehavior in Pre- and Post-Social Media Generations. In *Proceeding of ACL-11*.

Ryang S. and T. Abekawa. 2012. Framework of Automatic Text Summarization Using Reinforcement Learning. In *Proceeding of EMNLP-2012*.

Shen D., J. Sun, H. Li, Q. Yang and Zheng Chen. 2007. Document Summarization using Conditional Random Fields. In *Proceeding of IJCAI-07.*

Tan C., L. Lee, J. Tang, L. Jiang, M. Zhou and P. Li. 2011. User-Level Sentiment Analysis Incorporating Social Networks. In *Proceedings of KDD-11*.

Tang W., H. Zhuang, and J. Tang. 2011a. Learning to Infer Social Ties in Large Networks. In *Proceedings of ECML/PKDD-11*.

Tang J., Y. Zhang, J. Sun, J. Rao, W. Yu, Y. Chen, and A. Fong. 2011b. Quantitative Study of Individual Emotional States in Social Networks. I*EEE Transactions on Affective Computing*. vol.3(2), Pages 132-144.

Wan X. and J. Yang. 2008. Multi-document Summarization using Cluster-based Link Analysis. In *Proceedings of SIGIR-08.*

Wan X. 2011. Using Bilingual Information for Cross-Language Document Summarization. In P*roceedings of ACL-11*.

Wang H. and G. Zhou. 2012. Toward a Unified Framework for Standard and Update Multi-Document Summarization. *ACM Transactions on Asian Language Information Processing*. vol.11(2).

Xing E, M. Jordan, and S. Russell. 2003. A Generalized Mean Field Algorithm for Variational Inference in Exponential Families. In *Proceedings of UAI-03*.

Yang S., B. Long, A. Smola, N. Sadagopan, Z. Zheng and H. Zha. 2011a. Like like alike — Joint Friendship and Interest Propagation in Social Networks. In *Proceeding of WWW-11*.

Yang Z., K. Cai, J. Tang, L. Zhang, Z. Su and J. Li. 2011b. Social Context Summarization. In *Proceeding of SIGIR-11*.

Zhuang H, J. Tang, W. Tang, T. Lou, A. Chin, and X. Wang. 2012. Actively Learning to Infer Social Ties. In *Proceedings of Data Mining and Knowledge Discovery (DMKD-12)*, vol.25 (2), pages 270-297.

Optimized Event Storyline Generation based on Mixture-Event-Aspect Model

Lifu Huang
Shenzhen Key Lab for Cloud Computing Technology and Applications,
Peking University Shenzhen Graduate School,
Shenzhen, Guangdong 518055, P.R.China
warrior.fu@gmail.com

Lian'en Huang*
Shenzhen Key Lab for Cloud Computing Technology and Applications,
Peking University Shenzhen Graduate School,
Shenzhen, Guangdong 518055, P.R.China
hle@net.pku.edu.cn

Abstract

Recently, much research focuses on event storyline generation, which aims to produce a concise, global and temporal event summary from a collection of articles. Generally, each event contains multiple sub-events and the storyline should be composed by the component summaries of all the sub-events. However, different sub-events have different part-whole relationship with the major event, which is important to correspond to users' interests but seldom considered in previous work. To distinguish different types of sub-events, we propose a mixture-event-aspect model which models different sub-events into local and global aspects. Combining these local/global aspects with summarization requirements together, we utilize an optimization method to generate the component summaries along the timeline. We develop experimental systems on 6 distinctively different datasets. Evaluation and comparison results indicate the effectiveness of our proposed method.

1 Introduction

With the rapid growth of the World Wide Web, information explosion has become an important issue to modern people. Those who search for information from the Internet often get lost and confused by the overwhelmingly large collection of web documents. So how to get a concise and global picture for a given event subject is an urgent problem to be solved. Although many document understanding systems have been proposed, such as multi-document summarization systems, to generate a compressed summary by extracting the major information from the collection of documents, they ignored the dynamic development information of an event. Intuitively, each event is long-running and contains multiple sub-events, including related events. Users are likely to prefer a summary of all occurrences of all the sub-events along the timeline of the event. This motivates us to study the task of generating event storyline from a collection of web documents related to an event subject.

The research of event storyline summarization is popular in recent years. Its task is to summarize a collection of web documents by extracting representative information based on all the sub-events and generate a global summary. Generally, generating such a global storyline is quite interesting for the following main reasons: (1) It can help people catch the whole incident based on an overall temporal structured summary for a given subject, and understand the cause, climax, development process and result of an event. (2) It can also make people know what other events are related, or the effect of this incident to subsequent events, which can present the evolution of an event along a timeline.

Though several methods of generating event storyline have been proposed recently, there are still some problems unresolved. As event storyline summarization is a process to generate component summaries based on the multiple sub-events, which is different from traditional summarization focusing on only one subject, so how to exactly extract all the sub-events is the first challenge. Moreover, users tend to bias to the sub-events which have global

*Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 726–735,
Seattle, Washington, USA, 18-21 October 2013.

consistency with the given event subject, so the sub-events should not be considered equally when generating the component summaries. It is also a great challenge to generate a qualified summary based on the different types of sub-events. The component summaries should be correlative across different dates based on the global collection (Yan et al., 2011a).

Mei and Zhai (Mei and Zhai, 2005) proposed to use theme or topic to model different sub-events, which is to some extent similar to our method. To be different, in this paper we introduce "local/global" property to distinguish different part-whole relationship between the sub-events and the major event, which have not been considered before in storyline generation or summarization, to improve the quality of the storyline. The local/global property corresponds to the elements of an event, such as the place, characters and other body information. These information reflects the relationship between the sub-events and the major event. Some sub-events have distinctive body information and little relevance with each other. They generally occur for a local period, which we name as "local-sub-events". While other sub-events often share common properties with each other and have close relationship with the major event and we call them as "global-sub-events". Here we give some examples to illustrate the difference. For the event "Connecticut school shooting" which occurred on Dec.14 2012, its sub-events such as "Obama's speech for this massacre" or "Gun control Act" have little word co-occurrences and distinctive event body information to each other, while the process and result of this tragedy can be regarded as global-sub-events which have a lot of word co-occurrences and share common properties with the major event.

Inspired by these, to detect different types of sub-events based on word co-occurrences between sub-events and the major event, we propose a mixture-event-aspect (MEA) model to formalize different types of sub-events into local/global aspects, which are implicated with clusters of sentences. Then combining the local/global aspects with summarization requirements together, we utilized an optimization approach to get the optimal component summaries along the timeline. We evaluate our method on 6 distinctively different datasets. Performance comparisons among different system-generated storylines demonstrate the necessity to distinguish different types of sub-events and also indicates the effectiveness of the proposed mixture-event-aspect model.

The rest of the paper is organized as follows. We briefly review the related work in section 2. In section 3 we present the details of optimized event storyline generation based on mixture-event-aspect model. Experiments and results are discussed in Section 4. Finally we draw a conclusion of this study in Section 5.

2 Related Work

Our work is related to several lines of research in the literature including multi-document summarization (MDS), topic detection and tracking (TDT), temporal text mining (TXM) and temporal news summarization (TNS).

Multi-document summarization is a process to generate a summary by reducing documents in size while retaining the main information. To date, different features and ranking strategies have been studied. Radev et al. (Radev et al., 2004) proposed to implement MEAD as a centroid-based summarizer by combining several predefined features to score the sentence. LexPageRank (Erkan and Radev, 2004) is the representative work which is based on PageRank (Page et al., 1999) algorithm. Some methods have been proposed to extend the conventional graph-based models recently including multi-layer graph incorporated with different relationship (Wan, 2008), ToPageRank based on the topic information (Pei et al., 2012) and DivRank (Mei et al., 2010) balancing the prestige and diversity.

Topic detection and tracking (TDT) aims to group news articles based on the topics discussed in them, detect some novel and previously unreported events and track future events related to the topics (Wang et al., 2012). Kumaran and Allan (Kumaran and Allan, 2004) showed how performance on new event detection could be improved by the use of text classification techniques as well as by using named entities in a new way. Makkonen et al. (Makkonen et al., 2004) proposed a method that incorporated simple semantics into TDT by splitting the term space into groups of terms. Krause et al. Wang et al. (Wang et al., 2007) and Wang et al. (Wang et al.,

2009) worked on topic tracking from multiple news streams. Their methods extracted meaningful topics from multi-source news collections and tracked different topics as they evolved from one to another along the timeline.

Our work is also related to temporal text mining and temporal news summarization. The task of temporal news summarization is to generate news summaries along the timeline from massive data. Chieu et al. (Chieu and Lee, 2004) built a system that extracted events relevant to a query from a collection of related documents and placed such events along a timeline. Yan et al. (Yan et al., 2011b) designed an evolutionary timeline summarization approach to construct a timeline of a topic by optimizing the relevance, coverage, coherence, and diversity. Lin et al. (Lin et al., 2012) explored the problem of generating storylines from microblogs for user input queries. They first proposed a language model with dynamic pseudo relevance feedback to obtain relevant tweets and then generated storylines via graph optimization.

3 Approach Details

In this section, we first propose a mixture-event-aspect model to detect local/global sub-events based on part-whole relationship with the major event and then present a new method to estimate the bursty of each aspect on a certain date. Afterwards we utilize an optimization method based on local/global aspects to extract the qualified summary.

3.1 Mixture-Event-Aspect Model

The key challenge to our storyline generation task is to detect and distinguish different types of sub-events contained in the article collection. In the collection, each sentence is assigned with a certain date and sentences that are assigned with the same date are grouped into the same sub-collection. Considering the consistency of content between the sub-events and the major event, we model different sub-events into two types: local-sub-event and global-sub-event, and introduce local/global aspects correspondingly. Generally, local aspects which correspond to local-sub-events have distinctive words distribution from each other and sustain for a local context while the global aspects corresponding to global-sub-events have coincident words distribution with the major event. To capture specific words, Titov and McDonald (Titov and McDonald, 2008) proposed a multi-grain topic model, relying on word co-occurrences within short paragraphs and Li et al. (Li et al., 2010) proposed a entity-aspect model based on word co-occurrences within single sentences. Inspired by these ideas, we rely on word co-occurrences within local period context to detect mixed local and global aspects implicated in the whole collection. We name this model as "Mixture-Event-Aspect (MEA)" model which can simultaneously detect local/global aspects and cluster sentences and words into different aspects.

3.1.1 Model Description

Our mixture-event-aspect (MEA) model can be extended from both the Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). We model two distinct types of aspects: global aspects and local aspects, based on their relationship with the major event. The distribution of global aspects is fixed for the collection while the distribution of local aspects is fixed to a local period of sub-collections. That means a sentence is sampled either from the mixture of the global aspects or from the local aspects specific for the local context. Here we take the event "Connecticut school shooting" as an example. For the sentence "On Sunday, President Obama came to Connecticut to give a lecture, expressing his sorrow for ... and calling for an end to such incidents", the words such as "Obama", "lecture", "express" are only occurred for the local period of two days and have no co-occurrence with other neighboring period sentences, so we sample the sentence as a local aspect sentence. But for the sentence "All schools in Newtown, the northeastern U.S. state of Connecticut were in lockdown after a shooting was reported at a local elementary school", the words such as "Connecticut", "shooting", "elementary" have high co-occurrence frequency in the whole collection, so we sample the sentence as a global aspect sentence.

To detect aspects, we first divide words into two types: aspect words and background words. Background words are commonly used in the whole event corpus while aspect words are clearly associat-

ed with the aspects of the sentences they occur in. Stop words are removed using a standard stop word list. In order to get the distribution of local aspects, we implement a mechanism called "Time Window" which covers S_p sequential time-based sub-collections. We associate each time window with a distribution over local aspects and a distribution defining preference of local aspects versus global aspects.

draw $\varphi^B \sim Dir(\beta)$, $\psi_c(v) \sim Dir(\lambda)$, $\pi \sim Dir(\gamma)$
draw $\varphi^{gl} \sim Dir(\beta)$ for A_{gl} times
draw $\varphi^{loc} \sim Dir(\beta)$ for A_{loc} times
choose a distribution of global aspects $\theta^{gl} \sim Dir(\alpha^{gl})$
For each time window v in collection c
    choose $\theta^{loc}_{c,v} \sim Dir(\alpha^{loc})$
    choose $\rho_{c,v} \sim Beta(\alpha^{mix})$
For each sentence s in collection c
    choose window $\nu_{c,s} \sim \psi_c$
    choose $\eta_{c,s} \sim \rho_{c,\nu_{c,s}}$
    if $\eta_{c,s} = gl, z_{c,s} \sim \theta^{gl}$
    if $\eta_{c,s} = loc, z_{c,s} \sim \theta^{loc}_{c,v}$
For each word w of sentence s in collection c
    draw $y_{c,s,n} \sim Multi(\pi)$
    draw $w_{c,s,n} \sim Multi(\varphi^B)$ if $y_{c,s,n} = 1$
    draw $w_{c,s,n} \sim Multi(\varphi^{z_{c,s}})$ if $y_{c,s,n} = 2$

Figure 1: The Collection Generation Process

Formally, let $C = \{C_t | t = 1, 2, 3, ..., T\}$ be T time based sub-collections related to the event subject, C_t represents the collection of sentences which are assigned with the date t. Let v be a time window containing S_p sequential sub-collections, $v = \{C_t | t = i, i+1, ..., i+S_p-1\}$. We draw a background unigram language model which generates words for all sub-collections, and draw A_{gl} global aspect unigram language models for global aspects and A_{loc} word distributions for local aspects. We assume these word distributions have a uniform Dirichlet prior $Dir(\beta)$. There is also a multinomial distribution π that controls in each sentence how often the word occurs as a background word or an aspect word. π has a Dirichlet prior with parameter γ. We assign each window v with an distribution over local aspects and a distribution ρ defining preference for local aspects versus global aspects. ρ has a Beta prior α^{mix}. A sentence can be sampled using any window which is chosen according to a categorical distribution.

In Figure 2 the corresponding graphical model is presented. This model allows for fast approximate inference with collapsed Gibbs sampling.

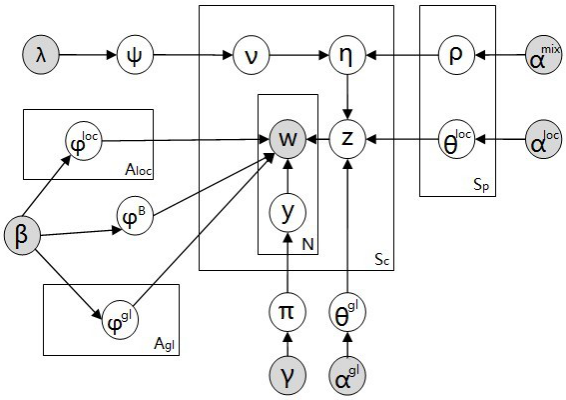


Figure 2: Mixture-Event-Aspect Model

Let S_C denotes the number of sentences in collection C, $N_{c,s}$ denotes the number of words in sentence s of collection c, and $w_{c,s,n}$ denotes the n^{th} word in sentence s. There are two kinds of hidden variables: $z_{c,s}$ for each sentence to indicate the aspect a sentence belongs to, and $y_{c,s,n}$ for each word to indicate whether a word is generated from the background model or the aspect model.

3.1.2 Inference via Gibbs Sampling

In order to estimate the hidden parameters in the model, we try to maximize distribution $p(\mathbf{z},\mathbf{y}|\mathbf{w}; \alpha, \beta, \gamma, \lambda)$, where **z**, **y** and **w** represent the set of all z, y and w variables, respectively. Given a sentence s in the collection c, we apply Gibbs Sampling to estimate the conditional probability for local/global aspects using the following rules:

$$p(v_{c,s} = v_h, \eta_{c,s} = gl, z_{c,s} = a | v', z', y, w) \propto \frac{n^c_{h*}+\lambda}{n^c_{(.)}+S^*_p\lambda} \cdot \frac{n^{c,v_h}_{gl}+\alpha^{mix}_{gl}}{n^{c,v_h}_{(.)}+\sum_{r' \in gl,loc} \alpha^{mix}_{r'}} \cdot \frac{n^c_{gl,a}+\alpha^{gl}}{n^c_{gl}+A_{gl}\alpha^{gl}} \cdot \frac{\prod_{l=1}^{L}\prod_{i=0}^{E(l)-1}(C^a_{(l)}+i+\beta)}{\Pi_{i=0}^{E(.)-1}(C^a_{(.)}+i+L\beta)}$$

$$p(v_{c,s}=v_h,\eta_{c,s}=loc,z_{c,s}=a|v^{'},z^{'},y,w)\propto \frac{n^c_{h^*}+\lambda}{n^c_{(.)}+S^*_p\lambda}\cdot\frac{n^{c,v_h}_{loc}+\alpha^{mix}_{loc}}{n^{c,v_h}_{(.)}+\sum\limits_{r^{'}\in gl,loc}\alpha^{mix}_{r^{'}}}\cdot\frac{n^{c,v_h}_{loc,a}+\alpha^{loc}}{n^{c,v_h}_{loc}+A_{loc}\alpha^{loc}}\cdot\frac{\prod_{l=1}^{L}\prod_{i=0}^{E(l)-1}(C^a_{(l)}+i+\beta)}{\prod_{i=0}^{E(.)-1}(C^a_{(.)}+i+L\beta)}$$

where $n^c_{h^*}$ =#$\{s_i|v_{c,s_i}=v_{i-s_p+h^*+1}\}$, denotes the number of times a sentence s_i in collection c is assigned to its h^{*th} window and for each sentence s_i, we have $h=i-S_p+1+h^*$. S^*_p is the number of windows that contain sentence s. $n^c_{(.)}$ denotes the number of sentences in collection c. S_p represents the number of dates a window covered. n^{c,v_h}_{gl} and n^{c,v_h}_{loc} are the number of sentences in window v_h that are assigned to global or local aspects. $n^{c,v_h}_{(.)}$ is the number of sentences assigned to window v_h. $n^c_{gl,a}$ is the number of sentences in all global aspects that are assigned to aspect a and $n^{c,v_h}_{loc,a}$ is the number of local aspect sentences in the window v_h are assigned to aspect a. A_{gl} is the number of global aspects in collection C while A_{loc} is the number of local aspects. $E(l)$ represents the number of times of word l occurs in the current sentence and is assigned to be an aspect word, while $E(.)$ is the total number of words in the current sentence that are assigned to be an aspect word.

Then we compute the assignments of the considered word. We sample the hidden variables $y_{c,s,n}$ for each word with the following rules:

$$p(y_{c,s,n}=1|z,y^{'})\propto\frac{C^{\pi}_{(1)}+\gamma}{C^{\pi}_{(.)}+2\cdot\gamma}\cdot\frac{C^B_{(w_{c,s,n})}+\beta}{C^B_{(.)}+L\cdot\beta}$$

$$p(y_{c,s,n}=2|z,y^{'})\propto\frac{C^{\pi}_{(2)}+\gamma}{C^{\pi}_{(.)}+2\cdot\gamma}\cdot\frac{C^a_{(w_{c,s,n})}+\beta}{C^a_{(.)}+L\cdot\beta}$$

where $C^{\pi}_{(1)}$ and $C^{\pi}_{(2)}$ are the numbers of words assigned to be background words and aspect words. $C^{\pi}_{(.)}$ is the total number of words. $C^B_{(.)}$ is the total number of background words and $C^a_{(.)}$ is the number of words assigned to aspect a.

By using the samples from Gibbs sampling, we can effectively make the following estimations:

$$\varphi^B_w=\frac{C^B_w+\beta}{C^B_{\cdot}+L\cdot\beta},\ \varphi^a_w=\frac{C^a_w+\beta}{C^a_w+L\cdot\beta},\ \psi_{h^*}=\frac{n^c_{h^*}+\lambda}{n^c_{(.)}+S^*_p\lambda}$$
$$\pi_y=\frac{C^{\pi}_y+\gamma}{C^{\pi}_{\cdot}+2\cdot\gamma},\ \rho^{c,v}_{\eta}=\frac{n^{c,v}_{\eta}+\alpha^{mix}_{\eta}}{n^{c,v}+\sum\limits_{r\in(gl,loc)}\alpha^{mix}_r}$$
$$\theta^{gl}_a=\frac{n^c_{gl,a}+\alpha^{gl}}{n^c_{gl}+A_{gl}\alpha^{gl}},\ \theta^{loc}_a=\frac{n^{c,v}_{loc,a}+\alpha^{loc}}{n^{c,v}_{loc}+A_{loc}\alpha^{loc}}$$

The hyper-parameters like $\alpha,\beta,\gamma,\lambda$ can be estimated using standard methods introduced in (Minka, 2000).

3.2 Bursty Period Detection

We borrow the definition of "bursty" from (Lappas et al., 2009) to measure the popularity of the event on a certain date. Intuitively, each aspect have different bursties on different dates. In this section, we try to obtain the temporal aspect sequences of an event based on the bursty periods of all the aspects. During its bursty period, one aspect should (1)be more popular than other aspects (2) be continuously more popular than other time. Following these intuitions, we design a method to measure the bursty of each aspect and get the bursty period.

Let A_k be the k^{th} aspect obtained from the mixture-event-aspect model, we estimate the bursty of A_k at a certain date t as follows.

$$bursty(A_{k,t})=p(t|A_k)=\frac{p(A_k|t)\cdot p(t)}{\sum\limits_{t^{'}}p(A_k|t^{'})p(t^{'})}$$

where $p(A_k|t)$ is measured by the number of sentences assigned to aspect A_k in date t divided by the total number of sentences in date t. $p(t)$ is estimated by the total number of sentences in aspect A_k divided by the overall number of sentences in the collection C.

After getting the bursty of aspect A_k at each date, we can find the most popular date and expand on both sides to obtain the whole burst period in which the bursties are higher than the neighboring aspects and continuous higher than other dates.

3.3 Optimization-based Storyline Generation

With the methods discussed in previous sections, we can get the local/global aspect sequence. Each aspect contains numbers of sentences and we are aiming to select the most representative ones to compose the final storyline. Considering users' bias and the length requirement, different aspects should have different proportions in the last storyline. For global aspects which correspond more to users' interest, they should share a larger proportion in the final storyline than local aspects. Thus, we use an optimization method to determine if a sentence is selected to be an summary sentence or to be discarded based on the multiple local/global aspects and finally get the optimal storyline. We formalize this problem as selecting a subset of sentences S from the aspect A_k

to minimize the information loss.

$$\arg\min \sum_{A_k \in C, S \in A_k} \sum_{z \in A_k - S, s \in S} O(z, s)$$

where $O(z, s)$ is the cost function which measures the cost of representing sentence z with sentence s. Generally, this is an NP-hard problem (Cheung et al., 2009) but we can use POPSTAR, an implementation of an approximate solution proposed by Resende and Werneck (Resende and Werneck, 2004). To model different costs between global or local aspects and determine the proportions of different aspects in the final storyline, we utilize a function $\zeta(s)$. When sentences z and s are local aspect sentences, $\zeta(s) = \chi$, or, $\zeta(s) = 1 - \chi$. Formally, we incorporate two kinds of decreasing/increasing logistic functions, $\ell_1(x) = 1/(1 + e^x)$ and $\ell_2(x) = e^x/(1 + e^x)$, to define the cost function as

$$O(z, s) = \zeta(s) \cdot \ell_1(S(s)) \cdot \ell_2(S(z)) \cdot D_{KL}(s, z)$$

where $S(s)$ and $S(z)$ are the ranking scores of sentences s and z among the aspect A_k with LexPageRank algorithm. $D_{KL}(s, z)$ is used to measure the similarity between sentence s and z with Kullback-Leibler divergence here.

With this optimization method, we get the representative sentences of each aspect for the given event subject. Combining all the representative sentences together based on the aspect sequence, we finally generate the storyline.

4 Experiments and Evaluation

4.1 Datasets

To evaluate our framework for event storyline generation, we conduct our experiments on the datasets amounting to 12418 articles for 6 event subjects from 6 famous news websites, which provide date edited by professional editors. Each article consists of three parts, title, publish-time and news content. Table 1 and Table 2 give the brief description of the 12418 articles. To generate reference summary, we invite 12 undergraduate students with good English ability to read the sentences, and for each event subject we ask two students to label human storylines.

4.2 Evaluation Metrics

We use the ROUGE[1] (Lin and Hovy, 2003) (Recall Oriented Understudy for Gisting Evaluation) toolkit to evaluate our framework, which has been widely applied for summarization evaluation. It evaluates the quality of a summary by counting the overlapping units between the candidate summary and reference summaries. There are many kinds of ROUGE metrics to measure the system-generated summarization such as ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-U, of which the most important one is ROUGE-N with 3 sub-metrics: precision, recall, and F-score.

$$ROUGE - N = \frac{\sum_{S \in RS} \sum_{N-gram \in S} Count_{match}(N - gram)}{\sum_{S \in RS} \sum_{N-gram \in S} Count(N - gram)}$$

where RS represents the reference summaries. N-gram∈RS in the metrics denotes the N-grams in reference summaries. $Count_{match}(N - gram)$ is the maximum number of N-grams co-occurring in the candidate summary and in the set of reference summaries. $Count(N - gram)$ is the number of N-grams in the reference summaries.

The ROUGE toolkit can report separate scores for 1, 2, 3, and 4-gram. In the experimental results we report three ROUGE F-measure scores: ROUGE-1, ROUGE-2, ROUGE-W metrics. The higher the ROUGE scores, the better the summary is.

4.3 Algorithms for Comparison

Given a collection of news articles, we first decompose them into sentences, and then assign each sentence with a certain date, afterwards stop-words removing and words stemming are performed. We choose the following algorithms as baseline systems. Specifically, baseline 2 and 3 are summarization

Table 1: News sources of the 6 datasets

News Sources	Number of Articles
CNN	2357
Fox News	1936
New York Times	2178
ABC	2113
Washington Post	1405
Xinhua	2429

Table 2: Event subjects of the datasets

Event Subjects	Number of Articles
Connecticut school shooting	1792
The earthquake in Tokyo, Japan	2046
The U.S. presidential election	2573
Sandy hurricane attacked America	1827
American curiosity rover landed on Mars	1651
The 30th London Olympic Games	2529

[1] http://www.isi/edu/licensed-sw/see/rouge/

systems which are similar to our storyline generation system. Then we choose baseline 4, 5 to evaluate the effectiveness of the proposed method. It must be said that all the systems are required to generate the same number of summary words with the human reference. We conduct the same preprocessing for all algorithms for fairness.

• Random : The method selects sentences randomly from the sentence collection.

• LexPageRank (LexRank): This method applies the graph-based multi-document summarization algorithm which first constructs a sentence connectivity graph based on the cosine similarity and then chooses top-ranked sentences with PageRank.

• Chieu : This method was proposed by Chieu (Chieu and Lee, 2004), utilizing interest and burstiness to rank sentences, and choosing the top-ranked query related sentences to construct the timeline.

• LDA+LexPageRank (LDALR) : This method first applies standard LDA to detect latent topics from the collection and clusters sentences to multiple aspects, then utilizes PageRank to generate the most representative component summaries from all the aspects.

• MEA+LexPageRank (MEALR) : This method applies the proposed mixture-event-aspect model to cluster sentences into multiple aspects and then utilizes PageRank to generate the most representative component summaries from all the aspects.

• MEA+Optimization (MEAOp) : This method extracts local/global aspects with the proposed mixture-event-aspect model, and then utilizes the optimization method to get the qualified summary.

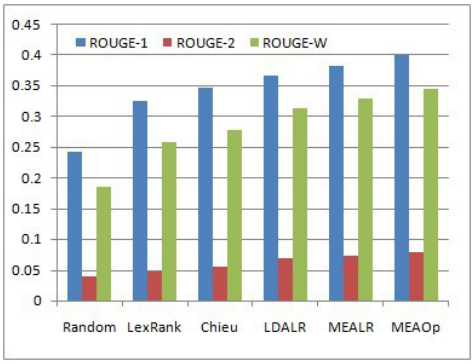


Figure 3: Overall performance for comparison

Table 3: Results of different systems on 6 subjects

	Subject1			Subject2		
Systems	R-1	R-2	R-W	R-1	R-2	R-W
Random	0.234	0.037	0.188	0.242	0.039	0.192
LexRank	0.317	0.045	0.257	0.326	0.051	0.262
Chieu	0.332	0.056	0.277	0.351	0.055	0.283
LDALR	0.356	0.069	0.297	0.369	0.066	0.327
MEALR	0.369	0.072	0.313	0.381	0.076	0.348
MEAOp	0.381	0.075	0.331	0.398	0.081	0.364
	Subject3			Subject4		
Systems	R-1	R-2	R-W	R-1	R-2	R-W
Random	0.258	0.042	0.191	0.234	0.036	0.179
LexRank	0.339	0.049	0.272	0.309	0.043	0.242
Chieu	0.364	0.059	0.296	0.329	0.053	0.264
LDALR	0.383	0.071	0.331	0.347	0.065	0.308
MEALR	0.396	0.076	0.3512	0.368	0.069	0.312
MEAOp	0.419	0.082	0.371	0.376	0.071	0.323
	Subject5			Subject6		
Systems	R-1	R-2	R-W	R-1	R-2	R-W
Random	0.222	0.034	0.166	0.264	0.045	0.195
LexRank	0.309	0.042	0.237	0.349	0.054	0.276
Chieu	0.319	0.049	0.258	0.371	0.062	0.293
LDALR	0.342	0.062	0.291	0.392	0.073	0.325
MEALR	0.372	0.068	0.299	0.406	0.079	0.349
MEAOp	0.384	0.070	0.309	0.427	0.087	0.368

4.4 Overall Performance Comparison

We experiment with all the baselines and our framework on the 6 datasets. We take the average F-score performance in terms of 3 ROUGE-F scores: ROUGE-1, ROUGE-2 and ROUGE-SU4. The overall results are shown in Figure 3 and details are listed in Tables 3.

Figure 3 and Table 3 show the performance of these systems on the same datasets. The local/global optimization balance parameter $\chi = 0.5$. From Figure 3 and Table 3 we have following observations:

• Generally, the Random gets the worst performance;

• The LexRank system outperforms Random algorithm. This is due to the fact that LexRank ranks all the sentences based on eigenvector centrality and the global relationship between sentences, which tends to select the most informative sentences as the summary.

• The results of Chieu (Chieu and Lee, 2004) system are better than those of LexRank. This may be mainly for the reason that Chieu used the date dimension to filter away uninteresting sentences by paraphrasing and defined two different ranking measures: interest and burtiness, to select top-ranked informative sentences.

• The LDALR system outperforms the Chieu sys-

tem. This may be for the fact that Chieu's method is actually based on flat clustering-based summarization, which is not as effective as LDA topic model to extract latent sub-events.

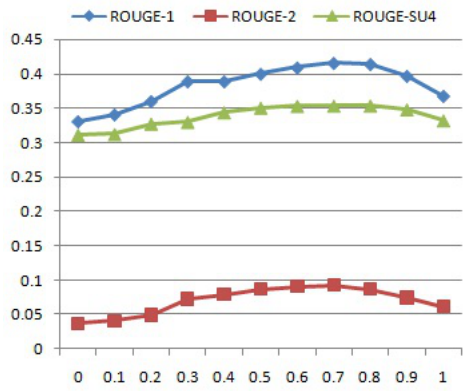


Figure 4: Examine the performance of the balance parameter χ

• The MEALR system outperforms the LDALR system. This may be mainly for the reason that MEALR utilizes the mixture-event-aspect model to detect the more salient sub-events based on the sub-whole relationship, which seems to satisfy users' bias to different sub-events.

• The MEAOp system which utilizes our method outperforms all the baselines, indicating the effectiveness of detecting different types of sub-events with mixture-event-aspect model and the necessity to distinguish different proportions of the component summaries based on local/global aspects.

4.5 Parameter Tuning

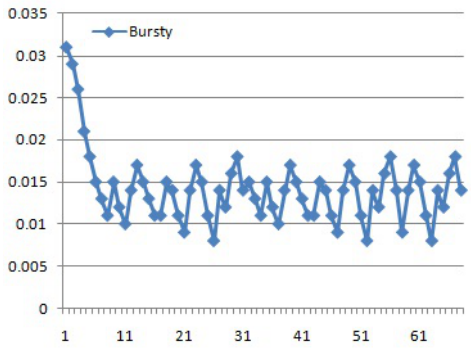


Figure 5: Aspect sequence of event "Connecticut school shooting" (X-axis is the number of days after Dec. 14, 2012. Y-axis is $bursty(A_{k,t})$)

In this section, we compare the performance of the parameters. The hyper-parameters such as α, β, γ, λ can be estimated using standard methods introduced by Minka (Minka, 2000). So we mainly examine the local/global optimization balance parameter χ. We try to evaluate the influence of this parameter on the three kinds of ROUGE measure results respectively. Figure 4 shows the performance of the balance parameters χ. It is obvious that when the balance parameter χ is set to 0.7 this method performs best.

4.6 Sample Output and Case Study

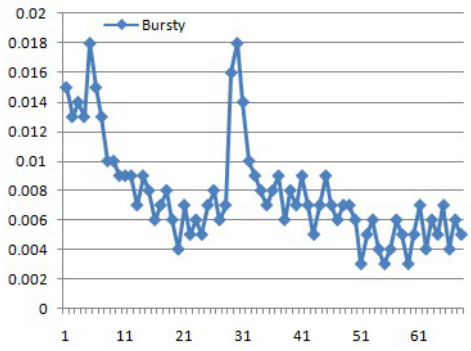


Figure 6: Bursties of sub-event "Gun control debate" (X-axis is the number of days after Dec. 14, 2012. Y-axis is $bursty(A_{k,t})$)

We take the event "Connecticut school shooting" as an example to show the usefulness of our method. Figure 5 shows the aspect sequence based on the bursty periods of all aspects. We select a sub-event "Gun control debate" and Figure 6 shows the bursties of this sub-event on the whole timeline. Table 4 shows part of the storylines for the event "Connecticut school shooting" generated by human and our method. Through observation, we find that the peak of the event "Connecticut school shooting" is around the date when it occurred, and the sub-event "Gun control debate" has two bursty periods around the two peaks. Compared with the human summary, our framework can extract the important sub-events contained in the collection, and satisfy users' interest on different sub-events based on the part-whole relationship with the event subject.

From the sample output and the human storylines, we also get some observations. (1) The component summary of global aspect tend to share larger pro-

Table 4: Selected part of storyline generated by MEAOp and human

Storyline Generated by human:
December 14, 2012 **Global Aspect**
Shooting massacre occurred in a primary school in Connecticut town, Sandy hoot Newton. 20-year-old Adam lanza broke into the primary school after shotting his mother, and in 10 minutes shot more than 100 times, killing twenty children and eight adult, including himself. The youngest death was a children in preschool students.
December 15, 2012 **Global Aspect**
Photos of the teachers and students shoot in the Connecticut massacre are released as well as the shooter's. The shooter was very smart but lonely.
December 16-17, 2012 **Local Aspect**
President Obama arrived in the locality of school shooting, mourned for the victims and made a speech.
December 18-20, 2012 **Local Aspect**
American gun control bill was put on the agenda again.
Storyline Generated by MEAOp:
December 14, 2012 **Global Aspect**
Children and adults gunned down in Connecticut school massacre. 20 children, six adults and the shooter are dead after shooting at Sandy Hoot Elementary School in Newtown, Connecticut. Three law enforcement officials say Adam Lanza, 20, was the shooter, and that he died apparently by his own hand. Suspect's mother, Nancy Lanza, found dead in suspects home in Newtown, law enforcement source says. Ryan Lanza, older brother of Adam Lanza, questioned by police but not labeled a suspect.
December 15-16, 2012 **Global Aspect**
Victims' names released Saturday; all of the slain children were either 6 or 7 years old. Understanding school lockdowns in regards to Connecticut shooting. Connecticut gunman recalled as intelligent but remote.
December 17, 2012 **Local Aspect**
President obama leads interfaith preyer vigil in newtown connecticut. A tearful Obama says "we've endured too many of these tragedies".
December 18-20, 2012 **Local Aspect**
Moderate dems join gun control debate call for commission on us violence gains. Gun debate gains traction as some lawmakers say its time to act.

portion in the final storyline. This is mainly for the reason that when researching for an event subject, users bias more to the information about the global-sub-events that have closely connection and coincident properties with the major event based on the part-whole relationship. So it is really necessary to distinguish different sub-events with distinctive properties. (2) Our system performs better for the persistent event, such as "The U.S. presidential election". This may be for the fact that these events are usually long running and have more global-sub-events than local-sub-events.

5 Conclusion

In this work, we study the task of event storyline generation and present a novel method. We innovatively introduce the properties of different sub-events based on word co-occurrences to determine the part-whole relationship with the major event and develop a mixture-event-aspect (MEA) model to formalize different types of sub-events into local/global aspects. Based on these local/global aspects, we utilize an optimization method to get the optimal component summaries along the aspect sequence. We conduct experiments with our method and various baselines on real web datasets. Through our experiments we notice that our method generates overall better storyline than other baselines. This indicates the effectiveness to detect different types of sub-events with the proposed mixture-event-aspect model and the necessity to distinguish different proportions of the component summaries based on local/global aspects.

Acknowledgments.

We thank the anonymous reviewers for their valuable and constructive comments. This work is financially supported by NSFC under the grant NO.61272340 and 60933004.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Jackie Chi Kit Cheung, Giuseppe Carenini, and Raymond T Ng. 2009. Optimization-based content se-

lection for opinion summarization. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, pages 7–14. ACL.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query based event extraction along a timeline. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 425–432. ACM.

G. Erkan and D.R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*, volume 4.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM.

Theodoros Lappas, Benjamin Arai, Manolis Platakis, Dimitrios Kotsakos, and Dimitrios Gunopulos. 2009. On burstiness-aware search for document sequences. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 477–486. ACM.

Peng Li, Jing Jiang, and Yinglin Wang. 2010. Generating templates of entity summaries with an entity-aspect model and pattern mining. In *Proceedings of the 48th annual meeting of the Association for Computational Linguistics*, pages 640–649. ACL.

C.Y. Lin and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. ACL.

Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. 2012. Generating event storylines from microblogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 175–184. ACM.

Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. 2004. Simple semantics in topic detection and tracking. *Information Retrieval*, 7(3-4):347–368.

Qiaozhu Mei and ChengXiang Zhai. 2005. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 198–207. ACM.

Q. Mei, J. Guo, and D. Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1009–1018. ACM.

Thomas Minka. 2000. Estimating a dirichlet distribution.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: bringing order to the web.

Yulong Pei, Wenpeng Yin, et al. 2012. Generic multi-document summarization using topic-oriented information. In *PRICAI 2012: Trends in Artificial Intelligence*, pages 435–446. Springer.

D.R. Radev, H. Jing, M. Styś, and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Mauricio GC Resende and Renato F Werneck. 2004. A hybrid heuristic for the p-median problem. *Journal of heuristics*, 10(1):59–88.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM.

X. Wan. 2008. Document-based hits model for multi-document summarization. *PRICAI 2008: Trends in Artificial Intelligence*, pages 454–465.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 784–793. ACM.

Xiang Wang, Kai Zhang, Xiaoming Jin, and Dou Shen. 2009. Mining common topics from multiple asynchronous text streams. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 192–201. ACM.

Dingding Wang, Tao Li, and Mitsunori Ogihara. 2012. Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs. *Proceddings of AAAI 2012*.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011a. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 433–443. ACL.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 745–754. ACM.

Automatically Determining a Proper Length for Multi-document Summarization: A Bayesian Nonparametric Approach

Tengfei Ma and Hiroshi Nakagawa
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
{matf@r., nakagawa@}dl.itc.u-tokyo.ac.jp

Abstract

Document summarization is an important task in the area of natural language processing, which aims to extract the most important information from a single document or a cluster of documents. In various summarization tasks, the summary length is manually defined. However, how to find the proper summary length is quite a problem; and keeping all summaries restricted to the same length is not always a good choice. It is obviously improper to generate summaries with the same length for two clusters of documents which contain quite different quantity of information. In this paper, we propose a Bayesian nonparametric model for multi-document summarization in order to automatically determine the proper lengths of summaries. Assuming that an original document can be reconstructed from its summary, we describe the "reconstruction" by a Bayesian framework which selects sentences to form a good summary. Experimental results on DUC2004 data sets and some expanded data demonstrate the good quality of our summaries and the rationality of the length determination.

1 Introduction

Text summarization is the process of generating a short version of a given text to indicate its main topics. As the number of documents on the web exponentially increases, text summarization has attracted increasing attention, because it can help people get the most important information within a short time.

In most of the existing summarization systems, people need to first define a constant length to restrict all the output summaries. However, in many cases it is improper to require all summaries are of the same length. Take the multi-document summarization as an example, generating the summaries of the same length for a 5-document cluster and a 50-document cluster is intuitively improper. More specifically, consider two different clusters of documents: one cluster contains very similar articles which all focus on the same event at the same time; the other contains different steps of the event but each step has its own topics. The former cluster may need only one or two sentences to explain its information, while the latter needs to include more.

Research on summary length dates back in the late 90s. Goldstein et al. (1999) studied the characteristics of a good summary (single-document summarization for news) and showed an empirical distribution of summary length over document size. However, the length problem has been gradually ignored later, since researchers need to fix the length so as to estimate different summarization models conveniently. A typical instance is the Document Understanding Conferences (DUC)[1], which provide authoritative evaluation for summarization systems. The DUC conferences collect news aritcles as the input data and define various summarization tasks, such as generic multi-document summarization, query-focused summarization and update summarization. In all the DUC tasks, the output is restricted within a length. Then human-generated

[1] After 2007, the DUC tasks are incorporated into the Text Analysis Conference (TAC).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 736–746,
Seattle, Washington, USA, 18-21 October 2013.

summaries are provided to evaluate the results of different summarization systems. Limiting the length of summaries contributed a lot to the development of summarization techniques, but as we discussed before, in many cases keeping the summaries of the same size is not a good choice.

Moreover, even in constant-length summarization, how to define a proper size of summaries for the summarization tasks is quite a problem. Why does DUC2007 main task require 250 words while Update task require 100 words? Is it reasonable? A short summary may sacrifice the coverage, while a long summary may cause redundance. Automatically determining the best size of summaries according to the input documents is valuable, and it may deepen our understanding of summarization.

In this work, we aim to find the proper length for document summarization automatically and generate varying-length summaries based on the document itself. The varying-length summarization is more robust for unbalanced clusters. It can also provide a recommended size as the predefined summary length for general constant-length summarization systems. We advance a Bayesian nonparametric model of extractive multi-document summarization to achieve this goal. As far as we are concerned, it is the first model that can learn appropriate lengths of summaries.

Bayesian nonparametric (BNP) methods are powerful tools to determine the size of latent variables (Gershman and Blei, 2011). They let the data "speak for itself" and allow the dimension of latent variables to grow with the data. In order to integrate the BNP methods into document summarization, we follow the assumption that the original documents should be recovered from the reconstruction of summaries (Ma and Wan, 2010; He et al., 2012). We use the Beta process as a prior to generate binary vectors for selecting active sentences that reconstruct the original documents. Then we construct a Bayesian framework for summarization and use the variational approximation for inference. Experimental results on DUC2004 dataset demonstrate the effectiveness of our model. Besides, we reorganize the original documents to generate some new datasets, and examine how the summary length changes on the new data. The results prove that our summary length determination is rational and necessary on unbalanced data.

2 Related Work

2.1 Research on Summary Length

Summary length is an important aspect for generating and evaluating summaries. Early research on summary length (Goldstein et al., 1999) focused on discovering the properties of human-generated summaries and analyzing the effect of compression ratio. It demonstrated that an evaluation of summarization systems must take into account both the compression ratios and the characteristics of the documents. Radev and Fan (2000) compared the readability and speedup in reading time of 10% summaries and 20% summaries[2] for topic sets with different number of documents. Sweeney et al. (2008) developed an incremental summary containing additional sentences that provide context. Kaisser et al. (2008) studied the impact of query types on summary length of search results. Other than the content of original documents, there are also some other factors affecting summary length especially in specific applications. For example, Sweeney and Crestani (2006) studied the relation between screen size and summary length on mobile platforms. The conclusion of their work is the optimal summary size always falls into the shorter one regardless of the screen size.

In sum, the previous works on summary length mostly put their attention on the empirical study of the phenomenon, factors and impacts of summary length. None of them automatically find the best length, which is our main task in this paper. Nevertheless, they demonstrated the importance of summary length in summarization and the reasonability of determining summary length based on content of news documents (Goldstein et al., 1999) or search results (Kaisser et al., 2008). As our model is mainly applied for generic summarization of news articles, we do not consider the factor of screen size in mobile applications.

2.2 BNP Methods in Document Summarization

Bayesian nonparametric methods provide a Bayesian framework for model selection and adaptation using nonparametric models (Gershman

[2] 10% and 20% are the compression rates, and the documents are from search results in information retrieval systems.

and Blei, 2011). A BNP model uses an infinite-dimensional parameter space, but invokes only a finite subset of the available parameters on any given finite data set. This subset generally grows with the data set. Thus BNP models address the problem of choosing the number of mixture components or latent factors. For example, the hierarchical Dirichlet process (HDP) can be used to infer the number of topics in topic models or the number of states in the infinite Hidden Markov model (Teh et al., 2006).

Recently, some BNP models are also involved in document summarization approaches (Celikyilmaz and Hakkani-Tür, 2010; Chang et al., 2011; Darling and Song, 2011). BNP priors such as the nested Chinese restaurant process (nCRP) are associated with topic analysis in these models. Then the topic distributions are used to get the sentence scores and rank sentences. BNP here only impacts the number and the structure of the latent topics, but the summarization framework is still constant-length. Our BNP summarization model differs from the previous models. Besides using the HDP for topic analysis, our approach further integrates the beta process into sentence selection. The BNP method in our model are directly used to determine the number of summary sentences but not latent topics.

3 BNP Summarization

In this section, we first introduce the BNP priors which will be used in our model. Then we propose our model called BNP summarization.

3.1 The Beta Process and the Bernoulli process

The beta process(BP) (Thibaux and Jordan, 2007; Paisley and Carin, 2009) and the related Indian buffet process(IBP) (Griffiths and Ghahramani, 2005) are widely applied to factor/feature analysis. By defining the infinite dimensional priors, these factor analysis models need not to specify the number of latent factors but automatically determine it.

Definition of BP (Paisley et al., 2010): Let B_0 be a continuous measure on a space Θ and $B_0(\Theta) = \gamma$. If B_k is defined as follows,

$$\begin{aligned} B_k &= \sum_{k=1}^{N} \pi_k \delta_{\theta_k}, \\ \pi_k &\sim Beta(\frac{\alpha\gamma}{N}, \alpha(1-\frac{\gamma}{N})) \\ \theta_k &\sim \frac{1}{\gamma} B_0 \end{aligned} \quad (1)$$

(where δ_{θ_k} is the atom at the location θ_k; and α is a positive scalar), then as $N \to \infty$, $B_k \to B$ and B is a beta process: $B \sim BP(\alpha B_0)$.

Finite Approximation: The beta process is defined on an infinite parameter space, but sometimes we can also use its finite approximation by simply setting N to a large number (Paisley and Carin, 2009).

Bernoulli Process: The beta process is conjugate to a class of Bernoulli processes, denoted by $X \sim Bep(B)$. If B is discrete, of the form in (1), then $X = \sum_k b_k \delta_{\theta_k}$ where the b_k are independent Bernoulli variables with the probability $p(b_k = 1) = \pi_k$. Due to the conjugation between the beta process priors and Bernoulli process, the posterior of B given M samples $X_1, X_2, ...X_M$ where $X_i \sim Bep(B) for i = 1, , , M.$ is also a beta process which has updated parameters:

$$\begin{aligned} &B|X_1, X_2, ..., X_M \\ &\sim BP(\alpha + M, \tfrac{\alpha}{\alpha+M} B_0 + \tfrac{1}{c+M} \textstyle\sum_i X_i) \end{aligned} \quad (2)$$

Application of BP: Furthermore, marginalizing over the beta process measure B and taking $\alpha = 1$, provides a predictive distribution on indicators known as the Indian buffet process (IBP) (Thibaux and Jordan, 2007). The beta process or the IBP is often used in a feature analysis model to generate infinite vectors of binary indicator variables(Paisley and Carin, 2009), which indicates whether a feature is used to represent a sample. In this paper, we use the beta process as the prior to select sentences.

3.2 Framework of BNP Summarization

Most existing approaches for generic extractive summarization are based on sentence ranking. However, these methods suffer from a severe problem that they cannot make a good trade-off between the coverage and minimum redundancy (He et al.,

2012). Some global optimization algorithms are developed, instead of greedy search, to select the best overall summaries (Nenkova and McKeown, 2012). One approach to global optimization of summarization is to regard the summarization as a reconstruction process (Ma and Wan, 2010; He et al., 2012) . Considering a good summary must catch most of the important information in original documents, the original documents are assumed able to be recovered from summaries with some information loss. Then the summarization problem is turned into finding the sentences that cause the least reconstruction error (or information loss). In this paper, we follow the assumption and formulate summarization as a Bayesian framework.

First we review the models of (Ma and Wan, 2010) and (He et al., 2012). Given a cluster of M documents $x_1, x_2, ..., x_M$ and the sentence set contained in the documents as $S = [s_1, s_2, ..., s_N]$, we denote all corresponding summary sentences as $V = [v_1, ..., v_n]$, where n is the number of summary sentences and N is the number of all sentences in the cluster. A document x_i and a sentence v_i or s_i here are all represented by weighted term frequency vectors in the space $\mathbb{R}^d$, where d is the number of total terms (words).

Following the reconstruction assumption, a candidate sentence v_i can be approximated by the linear combination of summary sentences: $s_i \simeq \sum_{j=1}^{n} w'_j v_j$, where w'_j is the weight for summary sentence v_j. Thus the document can also be approximately represented by a linear combination of summary sentences (because it is the sum of the sentences).

$$x_i \simeq \sum_{j=1}^{n} w_j v_j. \tag{3}$$

Then the work in (He et al., 2012) aims to find the summary sentence set that can minimize the reconstruction error $\sum_{i=1}^{N} ||s_i - \sum_{j=1}^{n} w'_j v_j||^2$; while the work in (Ma and Wan, 2010) defines the problem as finding the sentences that minimize the distortion between documents and its reconstruction $dis(x_i, \sum_{j=1}^{n} w_j v_j)$ where this distortion function can also be a squared error function.

Now we consider the reconstruction for each document, if we see the document x_i as the dependent variable, and the summary sentence set S as the independent variable, the problem to minimize the reconstruction error can be seen as a linear regression model. The model can be easily changed to a Bayesian regression model by adding a zero-mean Gaussian noise ϵ (Bishop, 2006), as follows.

$$x_i = \sum_{j=1}^{n} w_j v_j + \epsilon_i \tag{4}$$

where the weights w_j are also assigned a Gaussian prior.

The next step is sentence selection. As our system is an extractive summarization model, all the summary sentences are from the original document cluster. So we can use a binary vector $z_i =< z_{i1}, ..., z_{iN} >^T$ to choose the active sentences V (i.e. summary sentences) from the original sentence set S. The Equation (4) is turned into $x_i = \sum_{j=1}^{N} \phi_{ij} * z_{ij} s_j + \epsilon_i$. Using a beta process as a prior for the binary vector z_i, we can automatically infer the number of active component associated with z_i. As to the weights of the sentences, we use a random vector ϕ_i which has the multivariate normal distribution because of the conjugacy. $\phi_i \in \mathbb{R}^N$ is an extension to the weights $\{w_1, ...w_n\}$ in (4).

Integrating the *linear reconstruction* (4) and the *beta process*[3] (1), we get the complete process of summary sentence selection as follows.

$$\begin{aligned} x_i &= S(\phi_i \circ z_i) + \epsilon_i \\ S &= [s_1, s_2, ..., s_N] \\ z_{ij} &\sim Bernoulli(\pi_j) \\ \pi_j &\sim Beta(\frac{\alpha\gamma}{N}, \alpha(1 - \frac{\gamma}{N})) \\ \phi_i &\sim \mathcal{N}(0, \sigma_\phi^2 I) \\ \epsilon_i &\sim \mathcal{N}(0, \sigma_\epsilon^2 I) \end{aligned} \tag{5}$$

where N is the number of sentences in the whole document cluster. The symbol $\circ$ represents the elementwise multiplication of two vectors.

One problem of the reconstruction model is that the word vector representation of the sentences are sparse, which dramatically increase the reconstruction error. So we bring in topic models to reduce the

[3] We use the finite approximation because the number of sentences is large but finite

dimension of the data. We use a HDP-LDA (Teh et al., 2006) to get topic distributions for each sentence, and we represent the sentences and documents as the topic weight vectors instead of word weight vectors. Finally x_i is a K-dimensional vector and S is a $K * N$ matrix, where K is the number of topics in topic models.

4 Variational Inference

In this section, we derive a variational Bayesian algorithm for fast inference of our sentence selection model. Variational inference (Bishop, 2006) is a framework for approximating the true posterior with the best from a set of distributions $Q : q* = \arg\min_{q\in Q} KL(q(Z)|p(Z|X))$. Suppose $q(Z)$ can be partitioned into disjoint groups denoted by Z_j, and the q distribution factorizes with respect to these groups: $q(Z) = \prod_{j=1}^{M} q(Z_j)$. We can obtain a general expression for the optimal solution $q_j^*(Z_j)$ given by

$$\ln q_j^*(Z_j) = \mathbb{E}_{i\neq j}[\ln p(X, Z)] + const. \quad (6)$$

where $\mathbb{E}_{i\neq j}[\ln p(X, Z)]$ is the expectation of the logarithm of the joint probability of the data and latent variables, taken over all variables not in the partition. We will therefore seek a consistent solution by first initializing all of the factors $q_j(Z_j)$ appropriately and then cycling through the factors and replacing each in turn with a revised estimate given by (6) evaluated using the current estimates for all of the other factors.

Update for Z

$$p(z_{ij}|\pi_j, x_i, S, \phi_i) \propto p(x_i|z_{ij}, s_j, \phi_i)p(z_{ij}|\pi_j)$$

We use $q(z_{ij})$ to approximate the posterior:

$$\begin{aligned}
&q(z_{ij}) \\
&\propto \exp\{\mathrm{E}[\ln(p(x_i|z_{ij}, z_i^{-j}, S, \phi_i)) + \ln(p(z_{ij}|\pi))]\} \\
&\propto \exp\{\mathrm{E}[\ln(\pi_j)]\}* \\
&\exp\{\mathrm{E}[-\frac{1}{2\sigma_\epsilon^2}\left(x_i^{-j} - s_j z_{ij}\phi_{ij}\right)^T\left(x_i^{-j} - s_j z_{ij}\phi_{ij}\right)]\} \\
&\propto \exp\{\overline{\ln(\pi_j)}\}* \\
&\exp\{-\frac{\left(\overline{\phi_{ij}^2} * \overline{z_{ij}^2} * \overline{s_j^T s_j} - 2\overline{\phi_{ij}} * \overline{z_{ij}} * \overline{s_j}^T * \overline{x_i^{-j}}\right)}{2\sigma_\epsilon^2}\}
\end{aligned} \quad (7)$$

where $x_i^{-j} = x_i - S^{-j}(\phi_i^{-j} \circ z_i^{-j})$, and the symbol $^-$ indicates the expectation value. The $\overline{\phi_{ij}^2}$ can be extended to this form:

$$\overline{\phi_{ij}^2} = \overline{\phi_{ij}}^2 + \Delta_i^j \quad (8)$$

where Δ_i^j means the j^{th} diagonal element of Δ_i which is defined by Equation 13.

As z_i is a binary vector, we only calculate the probability of $z_{ij} = 1$ and $z_{ij} = 0$.

$$\begin{aligned}
&q(z_{ij} = 1) \propto \exp\{\overline{\ln(\pi_j)}\} * \\
&\exp\{-\frac{1}{2\sigma_\epsilon^2}\left(\overline{\phi_{ij}^2} * \overline{s_j^T s_j} - 2\overline{\phi_{ij}} * \overline{s_j}^T * \overline{x_i^{-j}}\right)\} \\
&q(z_{ij} = 0) \propto \exp\{\overline{\ln(1 - \pi_j)}\}
\end{aligned} \quad (9)$$

The expectations can be calculated as

$$\overline{\ln(\pi_j)} = \varphi(\frac{\alpha\gamma}{N} + \overline{n_j}) - \varphi(\alpha + M) \quad (10)$$

$$\overline{\ln(1 - \pi_j)} = \varphi(\alpha(1 - \frac{\gamma}{N}) + M - \overline{n_j}) - \varphi(\alpha + M) \quad (11)$$

where $\overline{n_j} = \sum_{i=1}^{M} z_{ij}$.

Update for π

$$p(\pi_j|Z) \propto p(\pi_j|\alpha, \gamma, N)p(Z|\pi_j)$$

Because of the conjugacy of the beta to Bernoulli distribution, the posterior of π is still a beta distribution:

$$\pi_j \sim Beta(\frac{\alpha\gamma}{N} + \overline{n_j}, \alpha(1 - \frac{\gamma}{N}) + M - \overline{n_j}) \quad (12)$$

Update for Φ

$$p(\phi_i|x_i, Z, S) \propto p(x_i|\phi_i, Z, S)p(\phi_i|\sigma_\phi^2)$$

The posterior is also a normal distribution with mean μ_i and covariance Δ_i.

$$\Delta_i = \left(\frac{1}{\sigma_\epsilon^2}\overline{\tilde{S}_i^T \tilde{S}_i} + \frac{1}{\sigma_\phi^2} I\right)^{-1} \quad (13)$$

$$\mu_i = \Delta_i \left(\frac{1}{\sigma_\epsilon^2}\overline{\tilde{S}_i}^T x_i\right) \quad (14)$$

Here $\tilde{S}_i \equiv S \circ \tilde{z}_i$ and $\tilde{z}_i \equiv [z_i, ..., z_i]^T$ is a $K \times N$ matrix with the vector z_i repeated K(the number of the latent topics) times.

$$\overline{\tilde{S}_i} = S * \overline{\tilde{z}_i} \quad (15)$$

$$\overline{\tilde{S}_i^T \tilde{S}_i} = (S^T S) \circ (\overline{z_i} * \overline{z_i}^T + Bcov_i) \quad (16)$$

$$Bcov_i = \text{diag}[\overline{z_{i1}}(1 - \overline{z_{i1}}), ..., \overline{z_{iN}}(1 - \overline{z_{iN}})] \quad (17)$$

Update for σ_ϵ^2

$$p(\sigma_\epsilon^2|\Phi, X, Z, S) \propto p(X|\Phi, Z, S, \sigma_\epsilon^2)p(\sigma_\epsilon^2)$$

By using a conjugate prior, inverse gamma prior $InvGamma(u, v)$, the posterior can be calculated as a new inverse gamma distribution with parameters

$$\begin{aligned} u' &= u + MK/2 \\ v' &= v + \frac{1}{2}\sum_{i=1}^{M}(||x_i - S(\overline{z_i} \circ \overline{\phi_i})|| + \xi_i) \end{aligned} \quad (18)$$

where

$$\begin{aligned} \xi_i = \textstyle\sum_{j=1}^{N}(\overline{z_{ij}^2} * \overline{\phi_{ij}^2} * s_j^T s_j - \overline{z_{ij}}^2 * \overline{\phi_{ij}}^2 * s_j^T s_j) \\ + \textstyle\sum_{j \neq l} \overline{z_{ij}} * \overline{z_{il}} * \Delta_{i,jl} * s_j^T s_l \end{aligned}$$

Update for σ_ϕ^2

$$p(\sigma_\phi^2|\Phi) \propto p(\Phi|\sigma_\phi^2)p(\sigma_\phi^2)$$

By using a conjugate prior, inverse gamma prior $InvGamma(e, f)$, the posterior can be calculated as a new inverse gamma distribution with parameters

$$\begin{aligned} e' &= e + MN/2 \\ f' &= f + \frac{1}{2}\sum_{i=1}^{M}\left((\overline{\Phi})^T\overline{\Phi} + trace(\Delta_i')\right) \end{aligned} \quad (19)$$

5 Experiments

To test the capability of our BNP summarization systems, we design a series of experiments. The aim of the experiments mainly includes three aspects:

1. To demonstrate the summaries extracted by our model have good qualities and the summary length determined by the model is reasonable.
2. To give examples where varying summary length is necessary.
3. To observe the distribution of summary length.

We evaluate the performance on the dataset of DUC2004 task2. The data contains 50 document clusters, with 10 news articles in each cluster. Besides, we construct three new datasets from the DUC2004 dataset to further prove the advantage of variable-length summarization. We separate each cluster in the original dataset into two parts where each has 5 documents, hence getting the *Separate* Dataset; Then we randomly combine two original clusters in the DUC2004 dataset, and get two datasets called *Combined1* and *Combined2*. Thus each of the clusters in the combined datasets include 20 documents with two different themes.

5.1 Evaluation of Summary Qualities

First, we implement our BNP summarization model on the DUC2004 dataset, with summary length not limited. At the topic analysis step, we use the HDP model and follow the inference in (Teh et al., 2006). For the sentence selection step, we use the variational inference described in Section 4, where the parameters in the beta process (5) are set as $\gamma = 1, \alpha = 1$. The summaries that we finally generate have an average length of 164 words. We design several popular unsupervised summarization systems and compare them with our model.

- The *Random* model selects sentences randomly for each document cluster.
- The *MMR* (Carbonell and Goldstein, 1998) strives to reduce redundancy while maintaining relevance. For generic summarization, we replace the query relevance with the relevance to documents.
- The *Lexrank* model (Erkan and Radev, 2004) is a graph-based method which choose sentences based on the concept of eigenvector centrality.
- The *Linear* Representation model (Ma and Wan, 2010) has the same assumption as ours and it can be seen as an approximation of the constant-length version of our model.

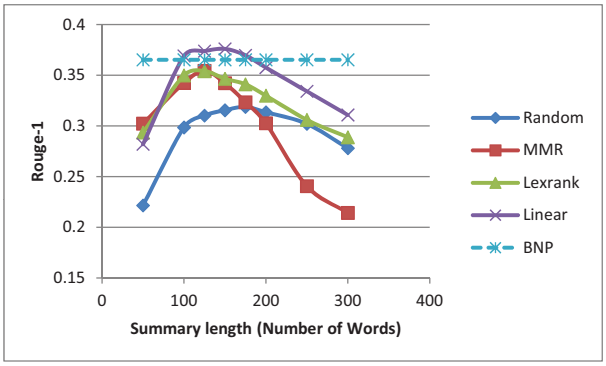


Figure 1: Rouge-1 values on DUC2004 dataset.

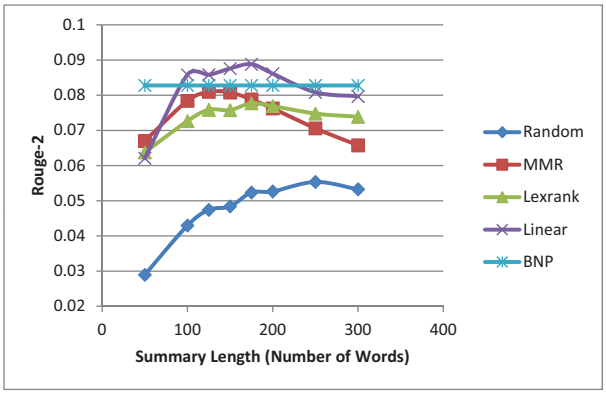


Figure 2: Rouge-2 values on DUC2004 dataset.

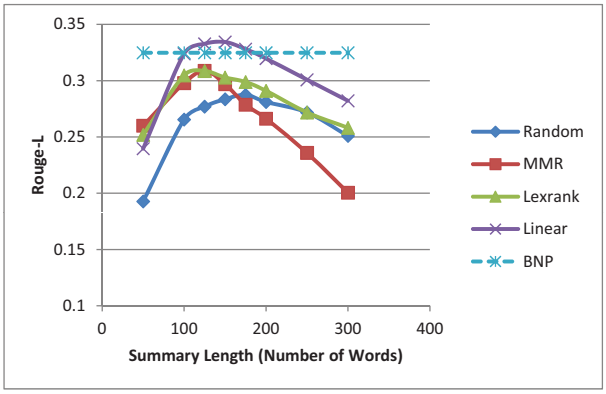


Figure 3: Rouge-L values on DUC2004 dataset.

All the compared systems are implemented at different predefined lengths from 50 to 300 words. Then we evaluate the summaries with ROUGE[4] tools (Lin and Hovy, 2003) in terms of the f-measure scores of Rouge-1 Rouge-2, and Rouge-L. The metric of Rouge f-measure takes into consideration the summary length in evaluation, so it is proper for our experiments. From Fig.1, Fig.2 and Fig.3, we can see that the result of BNP summarization (the dashed line) gets the second best value among all systems. It is only defeated by the *Linear* model but the result is comparable to the best in Fig.1 and Fig.3; while it exceeds other systems at all lengths. This proves the good qualities of our BNP summaries. The reason that the *Linear* system gets a little better result may be its weights for linear combination of summary sentences are guaranteed non-negative while in our model the weights are zero-mean Gaussian variables. This may lead to less redundance in sentence selection for the *Linear* Representation model.

Turn to the length determination. We take advantage of the *Linear* Representation model to approximate the constant-length version of our model. Comparing the summaries generated at different predefined lengths, Fig.4 shows the the model gets the best performance (Rouge values) at the length around 164 words, the length learned by our BNP model. This result partly demonstrates our length determination is rational and it can be used as the recommended length for some constant-length summarization systems, such as the *Linear* .

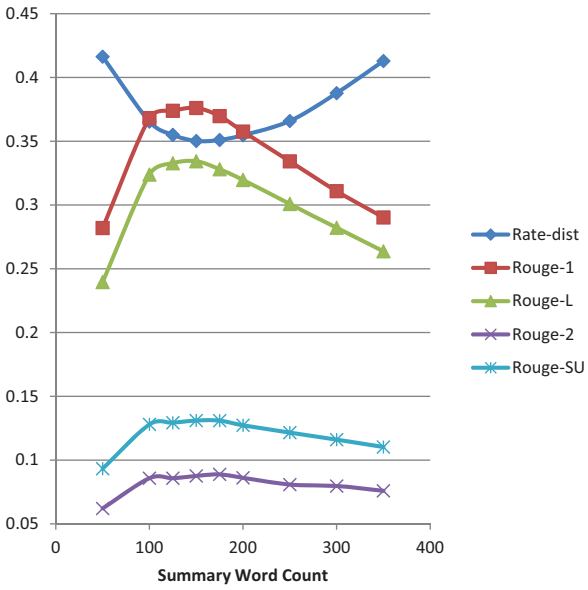


Figure 4: Rate-dist value V.S. summary word length.

[4]we use ROUGE1.5.5 in this work.

5.2 A New Evaluation Metric

The Rouge evaluation requires golden standard summaries as the base. However, in many cases we cannot get the reference summaries. For example, when we implement experiments on our expanded datasets (the separate and combined clusters of documents), we do not have exact reference summaries. Louis and Nenkova (2009) advanced an automatic summary evaluation without human models. They used the Jensen-Shannon divergence(JSD) between the input documents and the summaries as a feature, and got high correlation with human evaluations and the rouge metric. Unfortunately, it was designed for comparison at a constant-length, which cannot meet our needs. To extend the JSD evaluation to compare varying-length summaries, we propose a new measure based on information theory, the rate-distortion (Cover and Thomas, 2006).

Rate-Distortion: The distortion function $d(x, \hat{x})$ is a measure of the cost of representing the symbol x to a new symbol $\hat{x}$; and the rate can indicate how much compression can be achieved. The problem of finding the minimum rate can be solved by minimizing the functional

$$\mathcal{F}[p(\hat{x}|x)] = I(X;\hat{X}) + \beta\mathbb{E}(d(x,\hat{x})). \qquad (20)$$

where $I(X;\hat{X})$ denotes the mutual information. The rate-distortion theory is a fundamental theory for lossy data compression. Recently, it has also been successfully employed for text clustering (Slonim, 2002) and document summarization (Ma and Wan, 2010). Slonim (2002) claims that the mutual information $I(X;\hat{X})$ measures the compactness of the new representation. Thus the rate-distortion function is a trade-off between the compactness of new representation and the expected distortion. Specifically in summarization, the summaries can be seen as the new representation $\hat{X}$ of original documents X. A good summary balances the compression ratio and the information loss, thus minimizing the function (20). So we use the function (20)(we set $\beta = 1$) to compare which summary is a better compression. The JS-divergence (JSD), which has been proved to have high correlation with manual evaluation (Louis and Nenkova, 2009) for constant-length summary evaluation, is utilized as the distortion in the function. In the following sections, we simply call the values of the function (20) *rate-dist*. In fact, the rate-dist values can be seen as the JSD measure with length regularization.

To check the effectiveness of rate-dist measure, we evaluate all summaries generated in Section 5.1 with the new measure (the lower the better). Fig. 5 shows that the results accord with the ones in Fig. 1 and Fig. 3. Moreover, in Fig. 4, the curve of rate-dist values has a inverse tendency of Rouge measures (Rouge-1, Rouge-2, Rouge-L and Rouge-SU4 are all listed here), and the best performance also occurs around the summary length of 164 words. This even more clearly reveals that the BNP summarization achieves a perfect tradeoff between compactness and informativeness. Due to the accordance with rouge measures, it is promising to be regarded as an alternative to the rouge measures in case we do not have reference summaries.

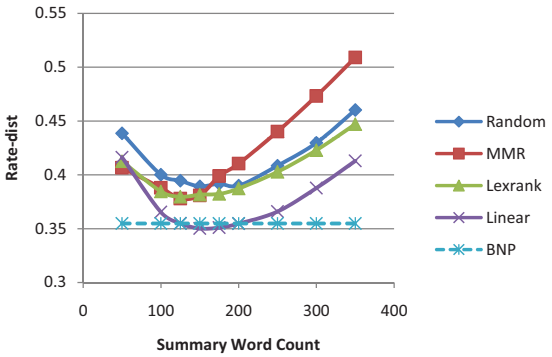


Figure 5: Comparison of BNP Summarization with other systems using rate-dist measure.

5.3 Necessity of Varying Summary Length

In this section, we discuss the necessity of length determination and how summary length changes according to the input data. As explained before, we generate three new datasets from the original DUC2004 dataset. Now we use them to indicate varying summary length is necessary when the input data varies a lot.

Table 1 shows the average summary length of different data sets. The results satisfy the intuitive expectation of summary length change. When we split a 10-document cluster into two 5-document parts, we expect the average summary length of the new clusters to be a little smaller than the original cluster but much larger than half of the original length,

because all the documents concentrate on the same themes. When we combine two clusters into one, the summary length should be smaller than the sum of the summary lengths of two original clusters due to some unavoidable common background information but much larger than the summary length of original clusters.

Original	Separate	Combined1	Combined2
164	115	250	231

Table 1: Average summary length (number of words) on different datasets

We also run the Linear Representation system at different lengths on the new datasets and evaluate the qualities. As we do not have golden standard for the new datasets, so we only use the rate-dist measure here. Results in Table 2,3,4 show the summaries which do not change the predefined length [5] perform significantly worse than the BNP summarization. All the comparison is statistically significant. So varying summary length is necessary when the input changes a lot, and our model can just give a good match to the new data. This characteristic also can be used to give recommended summary length for extractive summarization systems when given unknown data.

	Predefined	Unchanged	BNP
Length	665 bytes	164 words	115 words
Rate-dist	0.4130	0.4404	0.4007

Table 2: Comparison of summary lengths on Separate Dataset.

	Predefined	Unchanged	BNP
Length	665 bytes	164 words	250 words
Rate-dist	0.3768	0.3450	0.3238

Table 3: Comparison of summary lengths on Combined1 Dataset.

Then we observe the summary length distributions and compression ratios according to document size(the length of the whole documents in a cluster). The average summary length increases (Fig. 6),

[5]665 bytes is the DUC2004 requirement and 164 words is the best length on original data

	Predefined	Unchanged	BNP
Length	665 bytes	164 words	231 words
Rate-dist	0.3739	0.3464	0.3326

Table 4: Comparison of summary lengths on Combined2 Dataset.

while the compression ratios decreases (Fig. 7) as document size grows. The rule of the compression ratio here agrees with the rule in (Goldstein et al., 1999), although that work is done for single-document summarization.

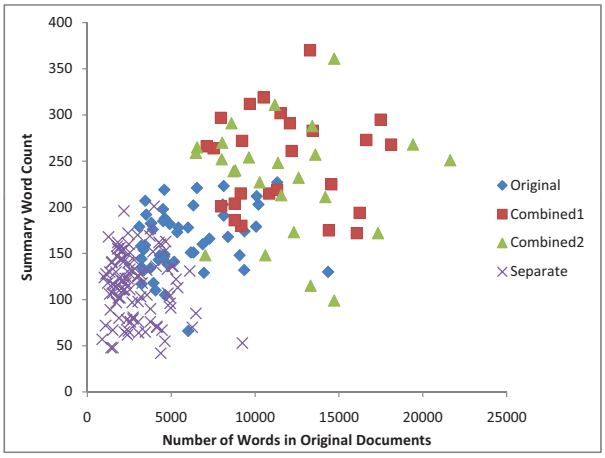


Figure 6: The distribution of summary word length.

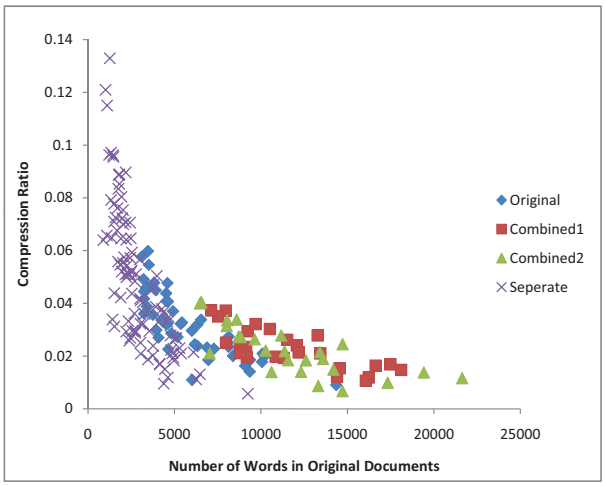


Figure 7: Compression ratio versus document word length.

6 Conclusion and Future Work

In this paper, we present a new problem of finding a proper summary length for multi-document summarization based on the document content. A Bayesian nonparametric model is proposed to solve this problem. We use the beta process as the prior to construct a Bayesian framework for summary sentence selection. Experimental results are shown on DUC2004 dataset, as well as some expanded datasets. We demonstrate the summaries we extract have good qualities and the length determination of our system is rational.

However, there is still much work to do for variable-length summarization. First, Our system is extractive-base summarization, which cannot achieve the perfect coherence and readability. A system which can determine the best length even for abstractive summarization will be better. Moreover, in this work we only consider the aspect of data compression and evaluate the performance using an information-theoretic measure. In future we may consider more human factors, and prove the summary length determined by our system agrees with human preference. In addition, in the experiments, we only use the imbalanced datasets as the example that intuitively needs varying the summary length. However, the data type is also important to impact the summary length. In future, we may extend the work by studying more cases that need varying summary length.

References

Christopher M. Bishop. 2006. *Pattern recognition and machine learning*. . Vol. 4. No. 4. New York: springer.

Jaime Carbonell, and Jade Goldstein. 1998. The Use Of Mmr, Diversity-Based Reranking For Reordering Documents And Producing Summaries. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1998.*

Asli Celikyilmaz and Dilek Hakkani-Tür. 2010. A Hybrid Hierarchical Model for Multi-Document Summarization. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 815-824.

Ying-Lan Chang, Jui-Jung Hung and Jen-Tzung Chien 2011. Bayesian Nonparametric Modeling Of Hierarchical Topics And Sentences. *IEEE International Workshop on Machine Learning for Signal Processing*, September 18-21, 2011, Beijing, China.

Thomas M. Cover, and Joy A. Thomas. 2006. *Elements of information theory*. Wiley-interscience, 2006.

William M. Darling and Fei Song. 2011. PathSum: A Summarization Framework Based on Hierarchical Topics. *Canadian AI Workshop on Text Summarization*, St. John's, Newfoundland.

Samuel J. Gershman and David M. Blei. 2011. A Tutorial On Bayesian Nonparametric Models. *Journal of Mathematical Psychology(2011).*

Thomas L. Griffiths and Zoubin Ghahramani. 2005. Infinite Latent Feature Models and the Indian Buffet Process. *Advances in Neural Information Processing Systems 18.*

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal and Jaime Carbonelly. 1999. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. *Proceedings of SIGIR'99* , pages 121-128.

Zhanying He, Chun Chen, Jiajun Bu, CanWang, Lijun Zhang, Deng Cai and Xiaofei He. 2012. Document Summarization Based on Data Reconstruction. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Michael Kaisser, Marti A. Hearst, John B. Lowe. 2008. Improving Search Results Quality by Customizing Summary Lengths. *Proceedings of ACL-08: HLT*, pages 701-709.

Chin-Yew Lin, Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. 2006. An Information-Theoretic Approach to Automatic Evaluation of Summaries. *Proceedings of NAACL2006*, pages 463-470.

Chin-Yew Lin, and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. *Proceedings of NAACL2003.*

Annie Louis and Ani Nenkova. 2009. Automatically Evaluating Content Selection in Summarization without Human Models. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 306-314. Singapore, 6-7 August 2009.

Tengfei Ma and Xiaojun Wan. 2010. Multi-document Summarization Using Minimum Distortion. *IEEE 10th International Conference on Data Mining (ICDM).*

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. *Mining Text Data, Chapter 3, Springer Science+Business Media, LLC* (2012).

John Paisley and Lawrence Carin. 2009. Nonparametric Factor Analysis with Beta Process Priors. *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada.

John Paisley, Aimee Zaas, Christopher W. Woods, Geoffrey S. Ginsburg and Lawrence Carin. 2010. A Stick-Breaking Construction of the Beta Process. *Proceedings of the 27 th International Confer- ence on Machine Learning*, Haifa, Israel, 2010.

Dragomir R. Radev and Weiguo Fan. 2000. Effective search results summary size and device screen size: Is there a relationship. *Proceedings of the ACL-2000 workshop on Recent advances in natural language processing and information retrieval*

Günes Erkan, and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22 (2004) 457-479.

Noam Slonim. 2002. The Information Bottleneck: Theory and Applications. *PHD Thesis of the Hebrew University* .

Simon Sweeney and Fabio Crestani. 2006. Effective search results summary size and device screen size: Is there a relationship. *Information Processing and Management* 42 (2006) 1056-1074.

Simon Sweeney, Fabio Crestani and David E. Losada. 2008. 'Show me more': Incremental length summarisation using novelty detection. *Information Processing and Management* 44 (2008) 663-686.

Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. 2007. Stick-breaking Construction for the Indian Buffet Process. *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Y.W. Teh, M.I. Jordan, M.J. Beal and D.M. Blei. 2006. Hierarchical Dirichlet Processes. *JASA* , 101(476):1566-1581.

Romain Thibaux and Michael I. Jordan. 2009. Hierarchical Beta Processes and the Indian Buffet Process. *AISTATS2007*.

A discourse-driven content model for summarising scientific articles evaluated in a complex question answering task

Maria Liakata
University of Warwick/
EMBL-EBI, UK
M.Liakata@warwick.ac.uk

Simon Dobnik
University of Gothenburg, Sweden
simon.dobnik@gu.se

Shyamasree Saha
EMBL-EBI, UK
saha@ebi.ac.uk

Colin Batchelor
Royal Society of Chemistry, UK
batchelorc@rsc.org

Dietrich Rebholz-Schuhmann
University of Zurich, Switzerland/
EMBL-EBI, UK
rebholz@ebi.ac.uk

Abstract

We present a method which exploits automatically generated scientific discourse annotations to create a content model for the summarisation of scientific articles. Full papers are first automatically annotated using the CoreSC scheme, which captures 11 content-based concepts such as Hypothesis, Result, Conclusion etc at the sentence level. A content model which follows the sequence of CoreSC categories observed in abstracts is used to provide the skeleton of the summary, making a distinction between dependent and independent categories. Summary creation is also guided by the distribution of CoreSC categories found in the full articles, in order to adequately represent the article content. Finally, we demonstrate the usefulness of the summaries by evaluating them in a complex question answering task. Results are very encouraging as summaries of papers from automatically obtained CoreSCs enable experts to answer 66% of complex content-related questions designed on the basis of paper abstracts. The questions were answered with a precision of 75%, where the upper bound for human summaries (abstracts) was 95%.

1 Introduction

The publication boom of the last few years, especially in the life sciences, has highlighted the need to facilitate automatic access to the information content of articles. Researchers, curators, reviewers all need to process a continuously expanding flow of articles whether the purpose is to follow the state of the art, curate large knowledge bases or have a good working knowledge of their own and related disciplines to assess progress in research. While a lot of effort has concentrated on information extraction of particular types of entities and relations from the scientific literature (Cohen and Hersh, 2005; Kim et al., 2009; Ananiadou et al., 2010; Kim et al., 2011), with a view to support scientists in obtaining relevant information from scientific articles and abstracts, less work has focussed on automatically combining such information in the form of a cohesive summary which preserves the context. Researchers rely to a great extent on author-written abstracts, but the latter suffer from a number of problems; they are less structured, vary significantly in terms of length, are often not self-contained and have been written independently of the main document (Teufel, 2010, p.83).

Teufel (2001; 2010), (Teufel and Moens, 2002) identify argumentative zones within scientific articles and use them to create use-targeted extractive summaries. Argumentative zones are annotations which designate the type of knowledge claim and rhetorical status for a sentence and how these relate to the communicative function of the entire paper. A selection of various combinations of argumentative zones are chosen for the use-targeted extractive summaries (rhetorical extracts), each of which fulfills a different role. For instance, purpose-oriented extracts less than 10 sentences long are generated containing a predetermined number of AIM, SOLUTION and BACKGROUND zones. As the emphasis of this approach was the identification of the argumentative zones, less attention was given to the sentence selection criteria for the extractive summaries.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 747–757,
Seattle, Washington, USA, 18-21 October 2013.

The sentences chosen for the rhetorical extracts were either all sentences of a particular category (in the case of rare categories) (Teufel and Moens, 2002), selected according to a classifier trained on a relevance gold standard (Teufel and Moens, 2002), manually or randomly selected (Teufel, 2010, p.60).

More recently Contractor et al. (2012) have used automatically annotated argumentative zones (Guo et al., 2011) to guide the creation of extractive summaries of scientific articles. Here argumentative zones are used as features for the summariser, along with verbs, tf-idf values and sentence location. They use a standard approach to summarisation, with a binary classification recognising candidate sentences which are then fed into a clustering mechanism. Extracts can be created to summarise the entire paper or focus on specific user-specified aspects. The number of sentences to include in the summary is prespecified (either directly or using a compression ratio).

Our approach also makes use of the scientific discourse for summarisation purposes. We use the scientific discourse to create a content model for extractive summarisation, with a focus on representing the content of the full paper, while keeping the cohesion of the narrative. We first automatically annotate the articles with a scheme which captures fine-grained aspects of the content and conceptual structure of the papers, namely the Core Scientific Concepts (CoreSC) scheme (Liakata et al., 2010; Liakata et al., 2012). The CoreSC scheme is "uniquely suited to recovering common types of scientific arguments about hypotheses, explanations, and evidence" (White et al., 2011), which are not readily identifiable by other annotation schemes. Also, when compared to argumentative zoning and more specifically its extension for chemistry papers, AZ-II (Teufel et al., 2009), it was shown to provide a greater level of detail in terms of categories denoting objectives, methods and outcomes whereas AZ-II focusses on the attribution of knowledge claims and the relation with previous work (Liakata et al., 2010).

We then use the distribution of CoreSC categories observed in abstracts to create a content model which provides a skeleton for extractive summaries. The reasoning behind this is to try to preserve cohesion within the summaries and we hypothesise that the sequence of CoreSC categories is a good proxy for cohesion (see section 3.1). In creating the summary, instantiating the content model, we identify independent categories and dependent categories, and we argue that in order to preserve the cohesion of the text the independent categories should be determined first (see section 3.2). We also preserve in the summary the distribution of CoreSC categories found in the corresponding full paper.

Finally, we evaluate the extractive summaries in a complex real world question-answering task, in which we assess the usefulness of the summaries as well as to what extent the generated CoreSC summaries represent the content of the original article. Experts are presented with different types of summaries and are asked to answer article-specific questions on the basis of the summaries (see section 4.1). Our results show that automatically generated CoreSC summaries can answer 66% of complex questions with 75% precision, outperforming a baseline of microsoft autosummarise summaries (See section 4.2).

We have also peformed an intrinsic evaluation of the summaries using ROUGE and automatic measures for summary informativeness, such as the Jensen-Shannon divergence, yielding positive results (See section 4.2). However, as such measures have not yet reached maturity and are harder to interpret, we consider the user-based evaluation to be a more reliable measure of summary quality.

Code for generating the summaries can be obtained by contacting the first author and/or visiting http://www.sapientaproject.com/software.

2 Related work

The Core Scientific Concepts (CoreSC) Scheme: The CoreSC scheme consists of three layers; the first layer corresponds to eleven concepts (Background (BAC), Hypothesis (HYP), Motivation (MOT), Goal (GOA), Object (OBJ), Method (MET), Model (MOD), Experiment (EXP), Observation (OBS), Result (RES) and Conclusion (CON)); the second layer corresponds to properties of the concepts (e.g. New/Old) and the third layer provides identifiers which link instances of the same category. Liakata et al. (2010) created a corpus of 265 full scientific articles from chemistry and biochemistry annotated

with this scheme and trained classifiers using SVMs and CRFs in (Liakata et al., 2012), with an accuracy of >51% across the 11 concepts. Their data and CoreSC classification system are available online and can provide a good benchmark for comparison. Louis & Nenkova (2012) have successfully used the CoreSC corpus for evaluating syntax-based coherence models, which indicates the strong connection between coherence and discourse structure.

Summarisation for scientific articles: A lot of the work on summarising scientific articles has focussed on citation-based summaries. Qazvinian & Radev (2008) use sentences from papers citing the article to be summarised. Sentences are clustered together creating a topic, with the combination of clusters forming a citation summary network. Qazvinian & Radev (2010), (Qazvinian et al., 2010) also make use of citation sentences in other scientific papers to summarize the contributions of a paper. The drawback of citation summaries is that a paper must be already cited, so this type of summary will not be useful to a paper reviewer. Also, citations of articles will have been influenced by other citations rather than the paper itself.

Document models for summarisation: Our content model has some similarities with content modelling using global sentence ordering (Barzilay and Lee, 2004; Chen et al., 2009). In (Barzilay and Lee, 2004) unsupervised methods are used to create HMM topic sequence models for newswire text articles. Topics are assigned to texts according to the content model and extracts of fixed length are created by selecting the topics most likely to occur in summaries. While we use supervised methods to annotate papers with a fixed set of topics (CoreSCs) in scientific papers, our summary content model for extracts shares similar principles such as global ordering of sentences and non-recurrence. However, their evaluation involved newspaper articles and extracts which are a lot shorter (15 and 6 sentences, respectively).

It is not clear whether unsupervised topic modelling such as (Chen et al., 2009) can be applied to scientific articles (over 100 sentences long), which by nature include repetition of topics. It would be interesting to make comparisons with summaries using content models learnt from our data automatically, following a similar approach to (Sauper et al., 2010) which learns a content model jointly with a particular supervised task in web-based documents.

3 Extractive Summarisation using CoreSCs

In this section we describe how we use CoreSC discourse categories annotated at the sentence level to create extractive summaries of full papers, which we subsequently evaluate in a question answering task in section 4.

To generate summaries we follow classic text extraction techniques while making use of a document content model based on CoreSCs. Our aim is for the content model to reflect both the distribution of CoreSCs in the paper as well as the discourse model of human summaries, as the latter is indicated by the generic ordering of CoreSC categories in abstracts encountered in a corpus of 265 annotated full papers (Liakata and Soldatova, 2009; Liakata et al., 2012). While we do not consider abstracts to be adequate summaries, we at least consider them to be coherent summaries, which is why the content model reflects the distribution of CoreSCs in the abstracts.

To create our summaries, we employed automatically generated CoreSC annotations, which are the output of the classifiers described in (Liakata et al., 2012). These classifiers assign CoreSC categories to sentences on the basis of features local to a sentence, such as significant n-grams, verbs and word triples, as well as global features such as the position of the sentence within the document and within a paragraph and section headers. The following subsections give details about the creation of extractive summaries from CoreSC categories.

3.1 A content model for CoreSC extractive summaries

Building an extractive summary using a computational model of document structure is an idea shared by many previous approaches, whether the model is hand-crafted, based on rhetorical elements (McKeown, 1985; Teufel and Moens, 2002) or rhetorical relations (Marcu, 1998b; Marcu, 1998a) or whether it is a content model, learnt automatically from text as in (Barzilay and Lee, 2004), focussing on the local content or a combination of the local content and global structure (Sauper et al., 2010).

Our document content model is primarily based on the global discourse of the article as provided by the type and number of CoreSC categories. However, unlike (Teufel and Moens, 2002), who take a fixed number of AZ categories of specific type to create rhetorical extracts, the number of categories used from each CoreSC category depends on their distribution in the original article. Any and all types of CoreSC category could potentially appear in a summary, as our summaries are meant to be representative of the entire content of the paper. Also, the ordering of the categories in the summary is learnt to reflect the ordering of categories observed in abstracts of papers from the same domain.

Our model also caters for local discourse dependencies. For example, the selection of a particular 'Method' sentence for inclusion in the summary should influence the choice of 'Experiment' sentences, which refers to particular experimental procedures performed. This is not an issue of concern to (Teufel and Moens, 2002), but relates to the notion of NUCLEUS and SATELLITE clauses, which form the foundation of Rhetorical Structure Theory (Mann and Thompson, 1998), and guides the summarisation paradigm of (Marcu, 1998a; Marcu, 1998b). However, the difference here is that we define a-priori certain categories to be independent (have the property of playing the role of nucleus in the discourse) and specify their relation with particular types of dependent categories. Thus, nuclearity becomes a property of the CoreSC category, which is indirectly inherited by the sentence.

Therefore, when creating the CoreSC content model for summaries we addressed the following issues: (i) summary length; (ii) number of sentences from each CoreSC, (iii) the ordering in which sentences from each CoreSC category should appear and (iv) the extraction of sentences according to independent and dependent categories.

- **Summary length:** While the literature (Teufel, 2010, p.45) suggests that 20–30% of the original document is required for an adequately informative summary, (Teufel, 2010, p.55) assumes this is too long for scientific papers. For this reason and to allow better comparison between papers of varying lengths, we fixed our summary length to 20 sentences. This is reasonable considering we have 11 CoreSCs, any and all of which can appear in both abstracts and full papers.
- **Number of sentences from each category:** To reflect the content of the paper, the distribution of the CoreSC categories in the extract follows the distribution of CoreSCs in the full paper.

  For each CoreSC we determine the number of sentences to be selected ($n(selected(C))$) by multiplying the ratio of that category in the paper by 20. A difficulty arises if the ratio of a particular concept in the paper is very low (≤ 0.05) in which case we prefer to include one sentence. If a particular concept is not at all present in the paper, the number of selected sentences for that category will be 0.
- **Ordering of CoreSC categories in the summary:** According to a study of empirical summaries (Liddy, 1991), sentences of a particular textual type appear in a particular order. Since paper abstracts were the closest approximation of human summaries available to us, CoreSC category transitions found in abstracts have been adopted in our content model for extracts. The transitions were derived semi-empirically. First, we extracted initial, medium and final bi-grams of categories from paper abstracts together with transition probabilities.

  Using this information we manually constructed transitions of the CoreSC categories that best fit the observed frequencies and our own intuitions. This gave us the following sequence: MOT $>$ (HYP) $>$ OBJ $>$ GOA $>$ BAC $>$ MOD $>$ MET $>$ EXP $>$ OBS $>$ (HYP) $>$ RES $>$ CON. HYP appears twice in the sequence as annotators had distinguished two types of hypotheses, global hypotheses (stated together with other objectives) and hypotheses about particular observations. The model provides an amalgamated representation of CoreSC concepts in abstracts. Interestingly, our semi-empirically derived model closely follows the content model for abstracts described in (Liddy, 1991). It would be interesting to see how this compares to a Markov model of CoreSC categories learnt from the annotated abstracts.

3.2 Sentence extraction based on independent and dependent categories

Sentence extraction involves selecting the most relevant sentences to include in a summary. Typically, this entails ranking the sentences according to some measure of salience and selecting the top n-best sentences. For example, a sentence will be represented by a number of features associated with it, such as whether it contains certain high frequency words or cue phrases, its location in the document, location in a paragraph (Brandow et al., 1995; Kupiec et al., 1995). Other methods include clustering based on sentence similarity and choosing the centroids (Erkan and Radev, 2004) or choosing the best connected sentences (Mihalcea and Tarau, 2004).

When sentences are classified according to CoreSC categories features such as the ones described above for text extraction are taken into account. Liakata et al. (2012) report that the most salient features for classifying CoreSC categories are overall n-grams, verbs and direct objects whereas other features such as the location of the sentence, the neighbouring section headings and whether a sentence contains citations play an important role for some of the categories. Thus, classification into CoreSC categories already provides a selection bias for sentence extraction.

As explained in section 3.1, the number of CoreSC categories in the summaries is determined according to their distribution in the paper and the order of the categories is specified in the content model. Salience for sentence extraction in this case is determined by the need to select the most representative sentences for a category. There isn't much point, for example, in identifying that we need to include a Method sentence (MET) and that this should be followed by an Experiment sentence (EXP), if we are not sure that those are indeed the categories of the sentences we are about to select.

We therefore rank sentences according to the classifier confidence score (probability) with which they were assigned a CoreSC category in (Liakata et al., 2012). The intuition behind this is that sentences with high classifier confidence will be less noisy, high precision cases and more representative of a particular category. Indeed, (Liakata et al., 2012) report statistical significance for the correlation between high classifier confidence and agreement between manual and automatic classification

However, as mentioned in section 3.1, there is inter-dependence between sentences in the text, which is in turn inherited by the categories assigned to them. For example, the highest ranking MET sentence will be related to an Experiment (EXP) or Background (BAC) sentence, which may not be the ones with the highest confidence score in their category.

In order to preserve discourse cohesion it is important to select related sentences from different categories. We resolve this by distinguising the CoreSCs into independent categories, which by definition are expected to show nucleus behaviour, and dependent categories. We also specify the relation between independent and dependent categories. The independent categories include the categories with the lowest percentage of sentences in scientific articles as reported in (Liakata et al., 2012), namely: Motivation (MOT) (1%), Goal (GOA)(1%), Hypothesis (HYP)(2%), Object (OBJ)(3%), Model (MOD)(9%), Conclusion (CON)(9%) and Method (MET)(11%). Categories whose sentence selection semantically depends on the former are Experiment (EXP)(10%), Background (BAC)(19%), Result (RES)(21%) and Observation (OBS)(14%). The independent categories also have higher precision than recall, in contrast to the dependent categories. While MET and EXP are almost equally represented in the CoreSC corpus, EXP by definition provides the detailed steps of an experimental method and thus it is semantically dependent on some MET category. More specifically, the dependencies are considered to be as follows: EXP, BAC depend on MET, RES depends on CON and OBS depends on RES (OBS is double-dependent).

Sentence extraction is driven by first identifying the independent categories based on classifier confidence scores and then choosing the corresponding dependent categories on the basis of both relatedness to the independent categories and classifier confidence. We use sentence proximity (defined below) as a measure for relatedness and combine it with classifier confidence during sentence extraction.

The mechanism to select sentences for inclusion in the summary, which considers category dependencies, proceeds as follows:

- For an independent category CatI, order sentences by decreasing order of confidence score. The confidence score is the average confidence score of the SVM and CRF classifiers reported in (Liakata et al., 2012) for a sentence.
- For a dependent category Cat, for which we need n sentences, given the selected sentences m from the corresponding independent category CatI we do the following:
  - If $m = 0$, then treat Cat as independent category for this case.
  - Otherwise, for each selected sentence t_i in CatI, calculate its proximity score to every sentence c_j of the dependent category Cat. $Proximity$ is defined as $1 - Distance$ where $Distance$ is an absolute difference in sentence ids between c_j and t_i normalised by the maximum absolute distance found between all c_j and t_i pairs.
  - The classifier prediction score for each c_j is multiplied by the $Proximity(c_j, t_i)$ score and the sentences are re-ranked according to the new scores, where only the n highest ranking c_js are kept. The last two steps result in an $m \times n$ matrix.
  - If $m = 1$, then the choice for the n sentences for Cat is straightforward.
  - Otherwise, we pick the n highest ranking c_js, proceeding row-wise. Thus, the highest ranking c_js for the highest ranking independent sentences t_i are given priority and any c_j is chosen at most once.

Once the sentence ids are selected for each independent and each dependent category we plug them into the content model. Sentence order is preserved within each CoreSC category. For example, if two Result sentences are selected, the order in which they appear in the paper will be preserved in the summary.

4 Summary evaluation via question answering

4.1 Task Description and experimental setup

We evaluate the extractive CoreSC summaries in terms of how well they enable 12 chemistry experts/evaluators (with at least a Masters degree in chemistry) to answer complex questions about the papers. Our test corpus consists of 28 papers held out from the ART/CoreSC corpus, roughly 1/9, which were annotated automatically with the SVM and CRF classifiers described in (Liakata et al., 2012) trained on the remaining 8/9 of the corpus. For each of the 28 papers in the test corpus, we generated CoreSC summaries automatically using the method described in section 3. We compare the performance of the experts on a question answering (Q-A) task when given the CoreSC summaries and two other types of summary, amounting to a total of three experimental conditions (A,B,C). The other two types of summary are the original paper abstracts (summaries A), in the absence of human summaries, and summaries generated by Microsoft Office Word 2007 AutoSummarize (summaries B).

Microsoft Office Word 2007 AutoSummarize (MA) is a widely available commercial system with reportedly good results (Garcia-Hernandez et al., 2009) and performance equivalent to TextRank (Mihalcea and Tarau, 2004). MA works by assigning a score to each word in a sentence depending on its frequency in the document and sentences are ranked and extracted according to the combination of scores of the words they contain. MA therefore follows classic lexicalised text extraction techniques, is domain independent and is completely agnostic of the discourse. For the latter reason, we considered MA to be a suitable baseline the comparison with which would illustrate the effect of using CoreSC categories on the summary and the merits of having a discourse based model for summarisation.

Neither the paper title nor section headings were available to any of the summarising systems as our extractive system does not make direct use of them and we were not sure how they would influence MA.

To ensure that each evaluator considered only one type of summary per paper, so as to avoid bias from previous stimuli, and to make sure all experts were exposed to all papers and all types of summary, the 12 experts were assigned to four groups (G1-G4) and were allocated 28 summaries each according to the Latin Square design in Table 1.[1].

The experimental setup follows the paradigm of (Teufel, 2001). However, while (Teufel, 2001) developed a Q-A task to evaluate summaries showing the contribution of a scientific article in relation to previous work, the purpose of the Q-A task at hand

[1]Initially we had four experimental conditions but one was dropped, so is not presented in this context

is to show the usefulness of the extracted summaries in answering questions on the paper, and how they compare to a discourse-agnostic baseline. In the case of (Teufel, 2001) the task consists of a fixed set of five questions, the same for all articles tuned particularly to the relation of current and previous work. By contrast, the current Q-A task aims to show how well the summaries represent the content of the entire paper, which means that questions are individual to each paper and required domain knowledge to create.

Each of the 12 experts answered three content-based questions per summary, where the questions were individual to each paper. An example of the questions and the corresponding answers for a given paper can be found below.

Example 4.1.1

- **Q:What do DNJ imino sugars inhibit the action of?**
  A: They inhibit glycosidases and ceramide glucosyltransferases.
- **Q:What methods do the authors use to study the conformation of N-benzyl-DNJ?**
  A: They use resonant two-photon ionization (R2PI), ultraviolet–ultraviolet (UV–UV) hole burning, and infrared (IR) ion-dip spectroscopies in conjunction with electronic structure theory calculations.
- **Q:What is the conformation of the exocyclic hydroxymethyl group?**
  A: The exocyclic hydroxymethyl group is axial to the piperidine ring (gauche- to the ring nitrogen).

As one can see, the questions are complex wh-questions and correspond to answers with multiple components. Questions were complex, to minimise the likelihood of correct random answers. They were designed by a senior chemistry expert with knowledge of linguistics, so that they could be answered based on the abstracts (A). For this purpose, the senior expert chose abstracts that were at least three sentences long. Ideally, the questions and answers should have been set on the basis of the entire paper, but this was not possible given our timeframe for the experiment.The underlying assumption is that a good summary should cover most of the main points of the paper. One of the merits of setting the questions on the basis of the abstracts was that the answers to be identified were deemed sufficiently important to be expressed in the humanly created abstract. However, automatic summaries created in the way proposed here could potentially answer questions beyond the scope of the abstract and in cases of very short abstracts be much more informative.

Experts were told that summaries were automatically generated with no details about different types of summary; it is assumed that none of them is completely familiar with the work mentioned in the 28 papers.

On average, it took experts less than 10 minutes to read a summary and answer the three content-based questions.

	Papers (28)			
Evaluator groups	1–7	8–14	15–21	22-28
G1	A	B	-	C
G2	C	A	B	-
G3	-	C	A	B
G4	B	-	C	A

Table 1: Distribution of summaries to evaluators

4.2 Results and Discussion

We compared each evaluator's answers obtained after reading a summary against the model answers set by the senior expert, the author of the questions, based on the abstract (A) of the corresponding paper. If an evaluator's answer is identical to a model answer, then this counts as "matched".

For instance in example 4.1.1 above, "axial to the piperidine ring", "gauche- to the ring nitrogen" and "The OH6 group is axial (Gauche) to the ring nitrogen" were all considered correct, fully matched answers to the question "What is the conformation of the exocyclic hydroxymethyl group?". In the case of the second question in the same example all of the following were considered correct and fully matched: "Resonant two-photon ionization (R2PI), UV/UV hole-burn, and IR ion-dip spectroscopies in conjunction with electronic structure theory calculations", "R2PI UV/UV hole-burn IR ion-dip e- structure theory calculations" and "a combination of resonant two-photon ionization (R2PI), UV/UV hole-burn, and IR ion-dip spectroscopies in conjunction with electronic structure theory calculations".

If the answer requires listing more than one item (as is the case with questions one and two of example 4.1.1), all of the items have to be matched. Partially matched answers are counted as "partially matched". Non-matching answers can be of two

types. If an un-matched answer coincided with the answer the senior expert would have given after reading that particular summary, then it was marked as "un-matched:justified": Such answers were correct given the particular summary, but are not necessarily correct with respect to the paper and do not count as alternative answers. If the answer was un-matched and also unjustified given the content of the summary, then it was marked as "un-matched:unjustified" . These are cases of evaluator error. Similarly, cases where the evaluator gave "N/A" as an answer were marked as "justified" or "unjustified" according to whether the senior expert could find the answer in the summary or not. The results from marking answers are shown in Table 2.

Number of	A	B	C
Matched	240	126	135
Partially matched	0	4	3
Un-matched:justified	0	25	15
N/A:justified	0	71	71
Un-matched:unjustified	5	11	17
N/A:unjustified	7	15	11
All answers	252	252	252

Table 2: Matches between summary-based answers and model answers

	Micro-AVG			Macro-AVG		
S. types	R	P	F	R	P	F
A	1	0.95	0.98	1	0.95	0.97
B	0.64	0.70	0.67	**0.64**	0.64	0.60
C	**0.66**	**0.75**	**0.70**	**0.64**	**0.70**	**0.65**

Table 3: Precision, Recall and F-score for answering questions using the four types of summary. A: abstracts, B: autosummarize, C:automatic CoreSC summaries.

We report Precision, Recall and F-score (P-R-F) for answering questions given each type of summary (Table 3). To calculate these we define TP as matched answers, FN as N/A:justified and FP everything else (partially matched + un-matched:justified + un-matched:unjustified + N/A:unjustified). Here, the standard definition of recall (TP/(TP+FN)) demonstrates how many questions can be answered using the summary (summary coverage) and Precision (TP/(TP+FP)) how well the questions are answered (summary clarity).

We consider the F-measure to be an overall indicator of the summary usefulness. Micro-averaging is obtained by adding all answers from all papers to calculate TP, FN and FP whereas macro-averaging calculates P-R-F first per paper and then averages over all papers.

The rankings remain consistent regardless of the averaging method. Condition A (abstracts) shows perfect Recall (the evaluators are able to answer all the questions) whereas Precision is affected by unjustified failed matches (Table 2). The perfect recall is hardly surprising as the questions are designed on the basis of the abstract but provides a sanity check for the experiment. The precision sets an upper bound for precision with automatic summaries. Summaries of condition C provide answers to more questions (Recall) and with greater accuracy (Precision) than summaries B. When macro-averaging, the Recall score of summaries C is tied with that for summaries B but Precision is 6% higher.

To verify the statistical significance for the difference in precision and recall for summaries B and C respectively, we performed Monte Carlo sampling 10000 times, for the populations of answers for summaries B and C. During each iteration of sampling, precision and recall were calculated, creating populations of 10000 recalls and 10000 precisions propagated to be representative of the original population of answers. A t-test performed on the population of precision and the population of recalls showed statistical significance at 95% in both cases, with summaries C having a precision of 5% higher and a recall of 1.4-1.6% higher than summaries B (see Table 4). Therefore, we can say that CoreSC summaries C are overall better for answering questions than summaries B.

Comparison between B and C (B-C)	
precision	recall
t = -105.90 df = 19959.79 p-value $<$ 2.2e-16	t = -32.52 df = 19994.40 p-value $<$ 2.2e-16
alternative hypothesis: true difference in means $\neq$ 0	
95% confidence interval: -0.051 -0.049 sample estimates: mean of x mean of y 0.696 0.746	95% confidence interval: -0.016 -0.014 sample estimates: mean of x mean of y 0.639 0.655

Table 4: Test for statistical significance betwen summaries B (microsoft) and C (CoreSC)

The difference in precision between summaries B and C shows the advantage of having a con-

tent model: summaries C are significantly clearer. We had also expected CoreSC summaries to have a much higher coverage than summaries B, and therefore significantly higher recall. However, this difference was less pronounced perhaps because autosummarize favours shorter sentences, which are more likely to be found in the abstracts. We expect that a refinement in the sentence selection criterion, which would also take sentence length into account, will help to showcase further the benefits of using a CoreSC-based content model.

Analysis using ROUGE showed that while summaries C had a slightly higher ROUGE-1 measure than summaries B (0.75 vs 0.73), with respect to abstracts, ROUGE-L was the same for the two (0.70).

In table 5 we also report measurements on summary informativeness based on divergence (Kullback Leibler (KL) divergence and Jensen Shannon (JS) divergence), as in (Louis and Nenkova, 2013). KL divergence is asymmetric and reflects the average number of bits wasted by coding samples of a distribution P using another distribution Q. JS divergence is an information-theoretic measure, reflecting the average distance of the KL divergence between summary and input (the full paper in our case) from the mean vocabulary distributions. Compared to other measures, JS divergence has been found to produce the best predictions of summary quality (Louis and Nenkova, 2013). In practice, what JS divergence tells us is how 'different'/divergent the summary is from the original paper. Low divergence scores are indicative of greater overlap between the summaries and the original paper and are considered positive in terms of the summary information content.

type	KLI-S	KLS-I	UnJSD	SJSD
B	1.66	0.70	0.21	0.19
C	1.40	0.62	0.18	0.17
random	1.61	0.79	0.21	0.19

Table 5: Macro-averaged divergence scores for the 28 test summaries. B: Autosummarize, C: CoreSC, random: random summaries each 20 sentences long for each paper. KLI-S: Average Kullback Leibler divergence between input and summary. KLS-I: Kullback Leibler divergence between summary and input, since KL divergence is not symmetric. UnJSD: Jensen Shannon divergence between input and summary. No smoothing. SJSD:A version with smoothing.

One can see the that CoreSC summaries have consistently lower divergence (both KL and JS) than microsoft autosummarise summaries and random summaries of the same length. This is a positive outcome but since such automatic measures of summary quality have not yet reached maturity and are harder to interpret, we consider the manual evaluation a more reliable indicator of summary informativeness and usefulness. Note that it is not appropriate to use divergence to assess the abstracts as this measure is influenced by the length of a text, which varies dramatically in the case of abstracts.

5 Conclusions and future work

We have shown how a content model based on the scientific discourse as annotated by the CoreSC scheme can be used to produce extractive summaries. These summaries can be generated as alternatives to abstracts. Since they preserve the distribution of CoreSCs in the paper and are not produced independently of it, as is the case with many abstracts, they are potentially more representative of abstracts than the full article. We have tested the usefulness CoreSC based summaries in answering complex questions relating to the content of scientific papers. Extracts from automated CoreSCs are informative, outperform microsoft autosummarise summaries, in both intrinsic and extrinsic evaluation, and enable experts to answer 66% of complex questions with a precision of 75%.

In the future we would like to experiment further with refining the sentence selection method so as to consider criteria for local cohesion, such as lexical chains. We would also like to perform comparisons with automatically induced content models and check their viability for scientific articles. We also would like to perform a human based evaluation of coherence and explore the full potential of these summaries as alternatives to author-written abstracts. This work constitutes a very important step in producing automatic summaries of scientific papers and enabling experts to extract information from the papers, a major requirement for resource curation, which is dependent on constant reviewing of the literature.

Acknowledgements

This work has been funded by an Early Career Leverhulme Trust Fellowship to Dr Liakata and by EMBL-EBI, UK. The authors would like to thank Annie Louis, Yufan Guo, Simone Teufel, Stephen Clark and the anonymous reviewers for their valuable comments. We would also like to thank Mo Abrahams for the python version of the summarisation code and the cafe summary toolkit.

References

S. Ananiadou, Pyysalo S., and J. Tsujii. 2010. Event extraction for systems biology by text mining the literature. *Trends in Biotechnology*, 28(7):381–390.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120. Best paper award.

R. Brandow, K. Mitze, and L. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31:675–685.

Harr Chen, S. R. K. Branavan, Regina Barzilay, and David R. Karger. 2009. Content modeling using latent permutations. *J. Artif. Int. Res.*, 36:129–163, September.

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

Danish Contractor, Yufan Guo, and Anna Korhonen. 2012. Using argumentative zones for extractive summarization of scientific articles. In *COLING*, pages 663–678.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:2004.

R.A. Garcia-Hernandez, Y. Ledeneva, G.M. Mendoza, A.H. Dominguez, J. Chavez, A. Gelbukh, and J.L.T. Fabela. 2009. Comparing commercial tools and state-of-the-art methods for generating text summaries. In *Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on*, pages 92–96, November.

Yufan Guo, Anna Korhonen, and Thierry Poibeau. 2011. A weakly-supervised approach to argumentative zoning of scientific documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 273–283. Association for Computational Linguistics.

T. Kim, J.and Ohta, S. Pyysalo, Y. Kano, and J. Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on BioNLP: Shared Task*, pages 1–9, Boulder, Colorado.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 1–6, Portland, Oregon, USA, June. Association for Computational Linguistics.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 68–73, New York, NY, USA. ACM.

M. Liakata and L.N. Soldatova. 2009. The ART Corpus. Technical report, Aberystwyth University.

M. Liakata, S. Teufel, A. Siddharthan, and C. Batchelor. 2010. Corpora for the conceptualisation and zoning of scientific papers. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Valetta,Malta.

M. Liakata, S. Saha, S. Dobnik, C. Batchelor, and Rebholz-Schuhmann D. 2012. Automatic recognition of conceptualisation zones in scientific articles and two life science applications. *Bioinformatics*, 28:991–1000.

Elizabeth DuRoss Liddy. 1991. The discourse-level structure of empirical abstracts: an exploratory study. *Inf. Process. Manage.*, 27:55–81, February.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1157–1168. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

W. C. Mann and S. A. Thompson. 1998. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 1998a. Improving summarization through rhetorical parsing tuning. In *Proceedings of The Sixth Workshop on Very Large Corpora*, pages 206–215, Montreal,Canada.

Daniel C. Marcu. 1998b. *The rhetorical parsing, summarization, and generation of natural language texts*. Ph.D. thesis, Toronto, Ont., Canada, Canada. AAINQ35238.

Kathleen R. McKeown. 1985. *Text generation: using discourse strategies and focus constraints to generate natural language text*. Cambridge University Press, New York, NY, USA.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 689–696, Morristown, NJ, USA. Association for Computational Linguistics.

Vahed Qazvinian and Dragomir R Radev. 2010. Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 555–564. Association for Computational Linguistics.

Vahed Qazvinian, Dragomir R Radev, and Arzucan Özgür. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 895–903. Association for Computational Linguistics.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2010. Incorporating content structure into text analysis applications. In *EMNLP'10*, pages 377–387.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Comput. Linguist.*, 28:409–445, December.

Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2009. Towards discipline-independent argumentative zoning: Evidence from chemistry and computational linguistics. In *Proceedings of EMNLP-09*, Singapore.

Simone Teufel. 2001. Task-based evaluation of summary quality: Describing relationships between scientific papersworkshop "automatic summarization", naacl-2001. In *NAACL-01 Workshop "Automatic Text Summarisation"*, Pittsburgh, PA.

Simone Teufel. 2010. *The Structure of Scientific Articles: Applications to Citation Indexing and Summarization*. CSLI Studies in Computational Linguistics. Center for the Study of Language and Information, Stanford, California.

Elizabeth White, K. Bretonnel Cohen, and Larry Hunter. 2011. Hypothesis and evidence extraction from full-text scientific journal articles. In *Proceedings of BioNLP 2011 Workshop*, pages 134–135, Portland, Oregon, USA, June. Association for Computational Linguistics.

Optimal Incremental Parsing via Best-First Dynamic Programming*

Kai Zhao[1] **James Cross**[1] **Liang Huang**[1,2]

[1]Graduate Center
City University of New York
365 Fifth Avenue, New York, NY 10016
{kzhao,jcross}@gc.cuny.edu

[2]Queens College
City University of New York
6530 Kissena Blvd, Queens, NY 11367
huang@cs.qc.cuny.edu

Abstract

We present the first provably optimal polynomial time dynamic programming (DP) algorithm for best-first shift-reduce parsing, which applies the DP idea of Huang and Sagae (2010) to the best-first parser of Sagae and Lavie (2006) in a non-trivial way, reducing the complexity of the latter from exponential to polynomial. We prove the correctness of our algorithm rigorously. Experiments confirm that DP leads to a significant speedup on a probablistic best-first shift-reduce parser, and makes exact search under such a model tractable for the first time.

1 Introduction

Best-first parsing, such as A* parsing, makes constituent parsing efficient, especially for bottom-up CKY style parsing (Caraballo and Charniak, 1998; Klein and Manning, 2003; Pauls and Klein, 2009). Traditional CKY parsing performs cubic time exact search over an exponentially large space. Best-first parsing significantly speeds up by always preferring to explore states with higher probabilities.

In terms of incremental parsing, Sagae and Lavie (2006) is the first work to extend best-first search to shift-reduce constituent parsing. Unlike other very fast greedy parsers that produce suboptimal results, this best-first parser still guarantees optimality but requires exponential time for very long sentences in the worst case, which is intractable in practice. Because it needs to explore an exponentially large space in the worst case, a bounded priority queue becomes necessary to ensure limited parsing time.

*This work is mainly supported by DARPA FA8750-13-2-0041 (DEFT), a Google Faculty Research Award, and a PSC-CUNY Award. In addition, we thank Kenji Sagae and the anonymous reviewers for their constructive comments.

On the other hand, Huang and Sagae (2010) explore the idea of dynamic programming, which is originated in bottom-up constituent parsing algorithms like Earley (1970), but in a beam-based non best-first parser. In each beam step, they enable state merging in a style similar to the dynamic programming in bottom-up constituent parsing, based on an equivalence relation defined upon feature values. Although in theory they successfully reduced the underlying deductive system to polynomial time complexity, their merging method is limited in that the state merging is only between two states in the same beam step. This significantly reduces the number of possible merges, because: 1) there are only a very limited number of states in the beam at the same time; 2) a lot of states in the beam with different steps cannot be merged.

We instead propose to combine the idea of dynamic programming with the best-first search framework, and apply it in shift-reduce dependency parsing. We merge states with the same features set globally to further reduce the number of possible states in the search graph. Thus, our DP best-first algorithm is significantly faster than non-DP best-first parsing, and, more importantly, it has a polynomial time complexity even in the worst case.

We make the following contributions:

- *theoretically*, we formally prove that our DP best-first parsing reaches optimality with polynomial time complexity. This is the first time that exact search under such a probabilistic model becomes tractable.

- more interestingly, we reveal that our dynamic programming over shift-reduce parsing is in parallel with the bottom-up parsers, except that we have an extra order constraint given by the shift action to enforce left to right generation of

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 758–768,
Seattle, Washington, USA, 18-21 October 2013.

$$\text{input} \quad w_0 \dots w_{n-1}$$

$$\text{axiom} \quad 0 : \langle 0,\ \epsilon \rangle : 0$$

$$\textsf{sh} \quad \frac{\ell : \langle j,\ S \rangle : c}{\ell + 1 : \langle j+1,\ S|w_j \rangle : c + sc_{\textsf{sh}}(j, S)} \ j < n$$

$$\textsf{re}_{\curvearrowleft} \quad \frac{\ell : \langle j,\ S|s_1|s_0 \rangle : c}{\ell + 1 : \langle j,\ S|s_1{}^{\curvearrowleft}s_0 \rangle : c + sc_{\textsf{re}_{\curvearrowleft}}(j, S|s_1|s_0)}$$

$$\textsf{re}_{\curvearrowright} \quad \frac{\ell : \langle j,\ S|s_1|s_0 \rangle : c}{\ell + 1 : \langle j,\ S|s_1{}^{\curvearrowright}s_0 \rangle : c + sc_{\textsf{re}_{\curvearrowright}}(j, S|s_1|s_0)}$$

Figure 1: Deductive system of basic non-DP shift-reduce parsing. Here ℓ is the step index (for beam search), S is the stack, c is the score of the precedent, and $sc_a(x)$ is the score of action a from derivation x. See Figure 2 for the DP version.

partial trees, which is analogous to Earley.

- *practically*, our DP best-first parser is only ∼2 times slower than a pure greedy parser, but is guaranteed to reach optimality. In particular, it is ∼20 times faster than a non-DP best-first parser. With inexact search of bounded priority queue size, DP best-first search can reach optimality with a significantly smaller priority queue size bound, compared to non-DP best-first parser.

Our system is based on a MaxEnt model to meet the requirement from best-first search. We observe that this locally trained model is not as strong as global models like structured perceptron. With that being said, our algorithm shows its own merits in both theory and practice. To find a better model for best-first search would be an interesting topic for future work.

2 Shift-Reduce and Best-First Parsing

In this section we review the basics of shift-reduce parsing, beam search, and the best-first shift-reduce parsing algorithm of Sagae and Lavie (2006).

2.1 Shift-Reduce Parsing and Beam Search

Due to space constraints we will assume some basic familiarity with shift-reduce parsing; see Nivre (2008) for details. Basically, shift-reduce parsing (Aho and Ullman, 1972) performs a left-to-right scan of the input sentence, and at each step, chooses either to *shift* the next word onto the stack, or to *reduce*, i.e., combine the top two trees on stack, either with left as the root or right as the root. This scheme is often called "arc-standard" in the literature (Nivre, 2008), and is the basis of several state-of-the-art parsers, e.g. Huang and Sagae (2010). See Figure 1 for the deductive system of shift-reduce dependency parsing.

To improve on strictly greedy search, shift-reduce parsing is often enhanced with beam search (Zhang and Clark, 2008), where b derivations develop in parallel. At each step we extend the derivations in the current beam by applying each of the three actions, and then choose the best b resulting derivations for the next step.

2.2 Best-First Shift-Reduce Parsing

Sagae and Lavie (2006) present the parsing problem as a search problem over a DAG, in which each parser derivation is denoted as a node, and an edge from node x to node y exists if and only if the corresponding derivation y can be generated from derivation x by applying one action.

The best-first parsing algorithm is an application of the Dijkstra algorithm over the DAG above, where the score of each derivation is the priority. Dijkstra algorithm requires the priority to satisfy the *superiority* property, which means a descendant derivation should never have a higher score than its ancestors. This requirement can be easily satisfied if we use a generative scoring model like PCFG. However, in practice we use a MaxEnt model. And we use the negative log probability as the score to satisfy the superiority:

$$x \prec y \Leftrightarrow x.score < y.score,$$

where the order $x \prec y$ means derivation x has a higher priority than y.[1]

The vanilla best-first parsing algorithm inherits the optimality directly from Dijkstra algorithm. However, it explores exponentially many derivations to reach the goal configuration in the worst case. We propose a new method that has polynomial time complexity even in the worst case.

[1]For simplicity we ignore the case when two derivations have the same score. In practice we can choose either one of the two derivations when they have the same score.

3 Dynamic Programming for Best-First Shift-Reduce Parsing

3.1 Dynamic Programming Notations

The key innovation of this paper is to extend best-first parsing with the "state-merging" method of dynamic programming described in Huang and Sagae (2010). We start with describing a parsing configuration as a non-DP *derivation*:

$$\langle i, j, ...s_2 s_1 s_0 \rangle,$$

where $...s_2 s_1 s_0$ is the stack of partial trees, $[i..j]$ is the span of the top tree s_0, and $s_1 s_2...$ are the remainder of the trees on the stack.

The notation $\mathbf{f}_k(s_k)$ is used to indicate the features used by the parser from the tree s_k on the stack. Note that the parser only extracts features from the top $d+1$ trees on the stack.

Following Huang and Sagae (2010), $\widetilde{\mathbf{f}}(x)$ of a derivation x is called *atomic features*, defined as the *smallest set* of features s.t.

$$\begin{aligned} &\widetilde{\mathbf{f}}(i, j, ...s_2 s_1 s_0) = \widetilde{\mathbf{f}}(i, j, ...s'_2 s'_1 s'_0) \\ \Leftrightarrow\ &\mathbf{f}_k(s_k) = \mathbf{f}_k(s'_k), \forall k \in [0, d]. \end{aligned}$$

The atomic feature function $\widetilde{\mathbf{f}}(\cdot)$ defines an equivalence relation $\sim$ in the space of derivations $\mathcal{D}$:

$$\begin{aligned} &\langle i, j, ...s_2 s_1 s_0 \rangle \sim \langle i, j, ...s'_2 s'_1 s'_0 \rangle \\ \Leftrightarrow\ &\widetilde{\mathbf{f}}(i, j, ...s_2 s_1 s_0) = \widetilde{\mathbf{f}}(i, j, ...s'_2 s'_1 s'_0) \end{aligned}$$

This implies that any derivations with the same atomic features are in the same *equivalence class*, and their behaviors are similar in shift and reduce. We call each equivalence class a DP *state*. More formally we define the space of all states $\mathcal{S}$ as:

$$\mathcal{S} \triangleq \mathcal{D}/\sim.$$

Since only the top $d+1$ trees on the stack are used in atomic features, we only need to remember the necessary information and write the state as:

$$\langle i, j, s_d...s_0 \rangle.$$

We denote a derivation x's state as $[x]_\sim$. In the rest of this paper, we always denote derivations with letters x, y, and z, and denote states with letters p, q, and r.

The deductive system for dynamic programming best-first parsing is adapted from Huang and Sagae (2010). (See the left of Figure 2.) The difference is that we do not distinguish the step index of a state.

This deductive system describes transitions between states. However, in practice we use one state's best derivation found so far to represent the state. For each state p, we calculate the prefix score, $p.pre$, which is the score of the derivation to reach this state, and the inside score, $p.ins$, which is the score of p's top tree $p.s_0$. In addition we denote the shift score of state p as $p.sh \triangleq sc_{\mathsf{sh}}(p)$, and the reduce score of state p as $p.re \triangleq sc_{\mathsf{re}}(p)$. Similarly we have the prefix score, inside score, shift score, and reduce score for a derivation.

With this deductive system we extend the concept of reducible states with the following definitions:

The set of all states with which a state p can legally reduce from the right is denoted $\mathcal{L}(p)$, or *left states*. (see Figure 3 (a)) We call any state $q \in \mathcal{L}(p)$ a *left state* of p. Thus each element of this set would have the following form:

$$\begin{aligned} \mathcal{L}(\langle i, j, s_d...s_0 \rangle) \triangleq \{ &\langle h, i, s'_d...s'_0 \rangle \mid \\ &\mathbf{f}_k(s'_{k-1}) = \mathbf{f}_k(s_k),\ \forall k \in [1, d] \} \end{aligned} \quad (1)$$

in which the span of the "left" state's top tree ends where that of the "right" state's top tree begins, and $\mathbf{f}_k(s_k) = \mathbf{f}_k(s'_{k-1})$ for all $k \in [1, d]$.

Similarly, the set of all states with which a state p can legally reduce from the left is denoted $\mathcal{R}(p)$, or *right states*. (see Figure 3 (a)) For two states p, q,

$$p \in \mathcal{L}(q) \Leftrightarrow q \in \mathcal{R}(p)$$

3.2 Algorithm 1

We constrain the searching time with a polynomial bound by transforming the original search graph with exponentially many derivations into a graph with polynomial number of states.

In Algorithm 1, we maintain a chart C and a priority queue Q, both of which are based on hash tables.

Chart C can be formally defined as a function mapping from the space of states to the space of derivations:

$$C : \mathcal{S} \rightarrow \mathcal{D}.$$

In practice, we use the atomic features $\widetilde{\mathbf{f}}(p)$ as the signature of state p, since all derivations in the same state share the same atomic features.

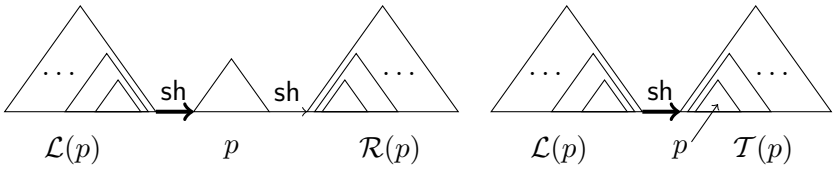
$$\textsf{sh} \quad \frac{\overset{\text{state } p\text{:}}{\langle \_, j, s_d ... s_0 \rangle : (c, \_)}}{\langle j, j+1, s_{d-1} ... s_0, w_j \rangle : (c + \xi, 0)} \; j < n$$

$$\textsc{Pred} \quad \frac{\langle \_, j, A \to \alpha . B \beta \rangle : (c, \_)}{\langle j, j, B \to . \gamma \rangle : (c{+}s,\ s)} \; (B \to \gamma) \in G$$

$$\textsf{re}_{\curvearrowright} \; \frac{\overset{\text{state } q\text{:}}{\langle k, i, s'_d ... s'_0 \rangle : (c', v')} \quad \overset{\text{state } p\text{:}}{\langle i, j, s_d ... s_0 \rangle : (\_, v)}}{\langle k, j, s'_d ... s'_1, s'_0 \frown s_0 \rangle : (c'{+}v{+}\delta,\ v'{+}v{+}\delta)} \; q \in \mathcal{L}(p)$$

$$\textsc{Comp} \; \frac{\langle k, i, A \to \alpha . B \beta \rangle : (c', v') \quad \langle i, j, B \rangle : (\_, v)}{\langle k, j, A \to \alpha B . \beta \rangle : (c'{+}v,\ v'{+}v)}$$

Figure 2: Deductive systems for dynamic programming shift-reduce parsing (Huang and Sagae, 2010) (left, omitting $\textsf{re}_{\curvearrowright}$ case), compared to weighted Earley parsing (Stolcke, 1995) (right). Here $\xi = sc_{\textsf{sh}}(p)$, $\delta = sc_{\textsf{sh}}(q) + sc_{\textsf{re}_{\curvearrowright}}(p)$, $s = sc(B \to \gamma)$, G is the set of CFG rules, $\langle i, j, B \rangle$ is a surrogate for any $\langle i, j, B \to \gamma . \rangle$, and $\_$ is a wildcard that matches anything.



(a) $\mathcal{L}(p)$ and $\mathcal{R}(p)$ (b) $\mathcal{T}(p) = \mathcal{R}(\mathcal{L}(p))$

Figure 3: Illustrations of left states $\mathcal{L}(p)$, right states $\mathcal{R}(p)$, and left corner states $\mathcal{T}(p)$. (a) Left states $\mathcal{L}(p)$ is the set of states that can be reduced with p so that $p.s_0$ will be the right child of the top tree of the result state. Right states $\mathcal{R}(p)$ is the set of states that can be reduced with p so that $p.s_0$ will be the left child of the top tree of the result state. (b) Left corner states $\mathcal{T}(p)$ is the set of states that have the same reducibility as shifted state p, i.e., $\forall p' \in \mathcal{L}(p)$, we have $\forall q \in \mathcal{T}(p), q \in \mathcal{R}(p')$. In both (a) and (b), thick sh arrow means shifts from multiple states; thin sh arrow means shift from a single state.

We use $C[p]$ to retrieve the derivation in C that is associated with state p. We sometimes abuse this notation to say $C[x]$ to retrieve the derivation associated with signature $\widetilde{\mathbf{f}}(x)$ for derivation x. This is fine since we know derivation x's state immediately from the signature. We say state $p \in C$ if $\widetilde{\mathbf{f}}(p)$ is associated with some derivation in C. A derivation $x \in C$ if $C[x] = x$. Chart C supports operation PUSH, denoted as $C[x] \leftarrow x$, which associate a signature $\widetilde{\mathbf{f}}(x)$ with derivation x.

Priority queue Q is defined similarly as C, except that it supports the operation POP that pops the highest priority item.

Following Stolcke (1995) and Nederhof (2003), we use the prefix score and the inside score as the priority in Q:

$$x \prec y \Leftrightarrow x.pre < y.pre \text{ or } (x.pre = y.pre \text{ and } x.ins < y.ins), \quad (2)$$

Note that, for simplicity, we again ignore the special case when two derivations have the same prefix score and inside score. In practice for this case we can pick either one of them. This will not affect the correctness of our optimality proof in Section 5.1.

In the DP best-first parsing algorithm, once a derivation x is popped from the priority queue Q, as usual we try to expand it with shift and reduce. Note that both left and right reduces are between the derivation x of state $p = [x]_\sim$ and an in-chart derivation y of left state $q = [y]_\sim \in \mathcal{L}(p)$ (Line 10 of Algorithm 1), as shown in the deductive system (Figure 2). We call this kind of reduction *left expansion*.

We further expand derivation x of state p with some in-chart derivation z of state r s.t. $p \in \mathcal{L}(r)$, i.e., $r \in \mathcal{R}(p)$ as in Figure 3 (a). (see Line 11 of Algorithm 1.) Derivation z is in the chart because it is the descendant of some other derivation that has been explored before x. We call this kind of reduction *right expansion*.

Our reduction with $\mathcal{L}$ and $\mathcal{R}$ is inspired by Nederhof (2003) and Knuth (1977) algorithm, which will be discussed in Section 4.

Algorithm 1 Best-First DP Shift-Reduce Parsing.

Let $\mathcal{L}_C(x) \triangleq C[\mathcal{L}([x]_\sim)]$ be in-chart derivations of $[x]_\sim$'s left states

Let $\mathcal{R}_C(x) \triangleq C[\mathcal{R}(p)]$ be in-chart derivations of $[x]_\sim$'s right states

1: **function** PARSE($w_0 \dots w_{n-1}$)
2: $C \leftarrow \emptyset$ ▷ empty chart
3: $Q \leftarrow \{\text{INIT}\}$ ▷ initial priority queue
4: **while** $Q \neq \emptyset$ **do**
5: $x \leftarrow \text{POP}(Q)$
6: **if** GOAL(x) **then return** x ▷ found best parse
7: **if** $[x]_\sim \notin C$ **then**
8: $C[x] \leftarrow x$ ▷ add x to chart
9: SHIFT(x, Q)
10: REDUCE($\mathcal{L}_C(x), \{x\}, Q$) ▷ left expansion
11: REDUCE($\{x\}, \mathcal{R}_C(x), Q$) ▷ right expansion
12: **procedure** SHIFT(x, Q)
13: TRYADD($\mathsf{sh}(x)$, Q) ▷ shift
14: **procedure** REDUCE(A, B, Q)
15: **for** $(x, y) \in A \times B$ **do** ▷ try all possible pairs
16: TRYADD($\mathsf{re}_{\curvearrowleft}(x, y)$, Q) ▷ left reduce
17: TRYADD($\mathsf{re}_{\curvearrowright}(x, y)$, Q) ▷ right reduce
18: **function** TRYADD(x, Q)
19: **if** $[x]_\sim \notin Q$ **or** $x \prec Q[x]$ **then**
20: $Q[x] \leftarrow x$ ▷ insert x into Q or update $Q[x]$

3.3 Algorithm 2: Lazy Expansion

We further improve DP best-first parsing with lazy expansion.

In Algorithm 2 we only show the parts that are different from Algorithm 1.

Assume a *shifted* derivation x of state p is a direct descendant from derivation x' of state p', then $p \in \mathcal{R}(p')$, and we have:

$$\forall ys.t.\ [y]_\sim = q \in \text{REDUCE}(\{p'\}, \mathcal{R}(p')), x \prec y$$

which is proved in Section 5.1.

More formally, we can conclude that

$$\forall ys.t.\ [y]_\sim = q \in \text{REDUCE}(\mathcal{L}(p), \mathcal{T}(p)), x \prec y$$

where $\mathcal{T}(p)$ is the *left corner states* of shifted state p, defined as

$$\mathcal{T}(\langle i, i{+}1, s_d ... s_0 \rangle) \triangleq \{\langle i, h, s'_d ... s'_0 \rangle \mid \mathbf{f}_k(s'_k) = \mathbf{f}_k(s_k),\ \forall k \in [1, d]\}$$

which represents the set of all states that have the same reducibility as a shifted state p. In other words,

$$\mathcal{T}(p) = \mathcal{R}(\mathcal{L}(p)),$$

Algorithm 2 Lazy Expansion of Algorithm 1.

Let $\mathcal{T}_C(x) \triangleq C[\mathcal{T}([x]_\sim)]$ be in-chart derivations of $[x]_\sim$'s left-corner states

1: **function** PARSE($w_0 \dots w_{n-1}$)
2: $C \leftarrow \emptyset$ ▷ empty chart
3: $Q \leftarrow \{\text{INIT}\}$ ▷ initial priority queue
4: **while** $Q \neq \emptyset$ **do**
5: $x \leftarrow \text{POP}(Q)$
6: **if** GOAL(x) **then return** x ▷ found best parse
7: **if** $[x]_\sim \notin C$ **then**
8: $C[x] \leftarrow x$ ▷ add x to chart
9: SHIFT(x, Q)
10: REDUCE($x.\mathit{lefts}, \{x\}, Q$) ▷ left expansion
11: **else if** $x.\mathit{action}$ **is** sh **then**
12: REDUCE($x.\mathit{lefts}, \mathcal{T}_C(x), Q$) ▷ right expan.
13: **procedure** SHIFT(x, Q)
14: $y \leftarrow \mathsf{sh}(x)$
15: $y.\mathit{lefts} \leftarrow \{x\}$ ▷ initialize *lefts*
16: TRYADD(y, Q)
17: **function** TRYADD(x, Q)
18: **if** $[x]_\sim \in Q$ **then**
19: **if** $x.\mathit{action}$ **is** sh **then** ▷ maintain *lefts*
20: $y \leftarrow Q[x]$
21: **if** $x \prec y$ **then** $Q[x] \leftarrow x$
22: $Q[x].\mathit{lefts} \leftarrow y.\mathit{lefts} \cup x.\mathit{lefts}$
23: **else if** $x \prec Q[x]$ **then**
24: $Q[x] \leftarrow x$
25: **else** ▷ $x \notin Q$
26: $Q[x] \leftarrow x$

which is illustrated in Figure 3 (a). Intuitively, $\mathcal{T}(p)$ is the set of states that have p's top tree, $p.s_0$, which contains only one node, as the left corner.

Based on this observation, we can safely delay the REDUCE($\{x\}, \mathcal{R}_C(x)$) operation (Line 11 in Algorithm 1), until the derivation x of a shifted state is popped out from Q. This helps us eliminate unnecessary right expansion.

We can delay even more derivations by extending the concept of left corner states to reduced states. Note that for any two states p, q, if q's top tree $q.s_0$ has p's top tree $p.s_0$ as left corner, and p, q share the same left states, then derivations of p should always have higher priority than derivations of q. We can further delay the generation of q's derivations until p's derivations are popped out.[2]

[2]We did not implement this idea in experiments due to its complexity.

4 Comparison with Best-First CKY and Best-First Earley

4.1 Best-First CKY and Knuth Algorithm

Vanilla CKY parsing can be viewed as searching over a hypergraph(Klein and Manning, 2005), where a hyperedge points from two nodes x, y to one node z, if x, y can form a new partial tree represented by z. Best-first CKY performs best-first search over the hypergraph, which is a special application of the Knuth Algorithm (Knuth, 1977).

Non-DP best-first shift-reduce parsing can be viewed as searching over a graph. In this graph, a node represents a derivation. A node points to all its possible descendants generated from shift and left and right reduces. This graph is actually a tree with exponentially many nodes.

DP best-first parsing enables state merging on the previous graph. Now the nodes in the hypergraph are not derivations, but equivalence classes of derivations, i.e., states. The number of nodes in the hypergraph is no longer always exponentially many, but depends on the equivalence function, which is the atomic feature function $\widetilde{\mathbf{f}}(\cdot)$ in our algorithms.

DP best-first shift-reduce parsing is still a special case of the Knuth algorithm. However, it is more difficult than best-first CKY parsing, because of the extra topological order constraints from shift actions.

4.2 Best-First Earley

DP best-first shift-reduce parsing is analogous to weighted Earley (Earley, 1970; Stolcke, 1995), because: 1) in Earley the PRED rule generates states similar to shifted states in shift-reduce parsing; and, 2) a newly completed state also needs to check all possible left expansions and right expansions, similar to a state popped from the priority queue in Algorithm 1. (see Figure 2)

Our Algorithm 2 exploits lazy expansion, which reduces unnecessary expansions, and should be more efficient than pure Earley.

5 Optimality and Polynomial Complexity

5.1 Proof of Optimality

We define a *best derivation* of state $[x]_\sim$ as a derivation x such that $\forall y \in [x]_\sim$, $x \preceq y$.

Note that each state has a unique feature signature. We want to prove that Algorithm 1 actually fills the chart by assigning a best derivation to its state. Without loss of generality, we assume Algorithm 1 fills C with derivations in the following order:

$$x_0, x_1, x_2, \ldots, x_m$$

where x_0 is the initial derivation, x_m is the first goal derivation in the sequence, and $C[x_i] = x_i, 0 \leq i \leq m$. Denote the status of chart right after x_k being filled as C_k. Specially, we define $C_{-1} = \emptyset$

However, we do not have superiority as in non-DP best-first parsing. Because we use a pair of prefix score and inside score, (pre, ins), as priority (Equation 2) in the deductive system (Figure 2). We have the following property as an alternative for superiority:

Lemma 1. *After derivation x_k has been filled into chart, $\forall x$ s.t. $x \in Q$, and x is a best derivation of state $[x]_\sim$, then x's descendants can not have a higher priority than x_k.*

Proof. Note that when x_k pops out, x is still in Q, so $x_k \preceq x$. Assume z is x's direct descendant.

- If $z = \mathsf{sh}(x)$ or $z = \mathsf{re}(x, \_)$, based on the deductive system, $x \prec z$, so $x_k \preceq x \prec z$.
- If $z = \mathsf{re}(y, x)$, $y \in \mathcal{L}(x)$, assume $z \prec x_k$.

  $$z.pre = y.pre + y.sh + x.ins + x.re$$

  We can construct a new derivation $x' \sim x$ by appending x's top tree, $x.s_0$ to y's stack, and

  $$x'.pre = y.pre + y.sh + x.ins < z.pre$$

  So $x' \prec z \prec x_k \preceq x$, which contradicts that x is a best derivation of its state.

With induction we can easily show that any descendants of x can not have a higher priority than x_k. □

We can now derive:

Theorem 1 (Stepwise Completeness and Optimality)**.** *For any k, $0 \leq k \leq m$, we have the following two properties:*

$$\forall x \prec x_k, [x]_\sim \in C_{k-1} \quad \text{(Stepwise Completeness)}$$
$$\forall x \sim x_k, x_k \preceq x \quad \text{(Stepwise Optimality)}$$

Proof. We prove by induction on k.

1. For $k = 0$, these two properties trivially hold.

2. Assume this theorem holds for $k = 2, ..., i-1$. For $k = i$, we have:

a) *[Proof for Stepwise Completeness]*

(Proof by Contradiction) Assume $\exists x \prec x_i$ s.t. $[x]_\sim \notin C_{i-1}$.Without loss of generality we take a best derivation of state $[x]_\sim$ as x. x must be derived from other best derivations only. Consider this derivation transition hypergraph, which starts at initial derivation $x_0 \in C_{i-1}$, and ends at $x \notin C_{i-1}$.

There must be a best derivation x' in this transition hypergraph, s.t. all best parent derivation(s) of x' are in C_{i-1}, but not x'.

If x' is a reduced derivation, assume x''s best parent derivations are $y \in C_{i-1}, z \in C_{i-1}$. Because y and z are best derivations, and they are in C_{i-1}, from Stepwise Optimality on $k = 1, ..., i-1$, $y, z \in \{x_0, x_1, \ldots, x_{i-1}\}$. From Line 7-11 in Algorithm 1, x' must have been pushed into Q when the latter of y, z is popped.

If x' is a shifted derivation, similarly x' must have been pushed into Q.

As $x' \notin C_{i-1}$, x' must still be in Q when x_i is popped. However, from Lemma 1, none of x''s descendants can have a higher priority than x_i, which contradicts $x \prec x_i$.

b) *[Proof for Stepwise Optimality]*

(Proof by Contradiction) Assume $\exists x \sim x_i$ s.t. $x \prec x_i$. From Stepwise Completeness on $k = 1, ..., i$, $x \in C_{i-1}$, which means the state $[x_i]_\sim$ has already been assigned to x, contradicting the premise that x_i is pushed into chart.

□

Both of the two properties have very intuitive meanings. Stepwise Optimality means Algorithm 1 only fills chart with a best derivation for each state. Stepwise Completeness means every state that has its best derivation better than best derivation p_i must have been filled before p_i, this guarantees that the global best goal derivation is captured by Algorithm 1.

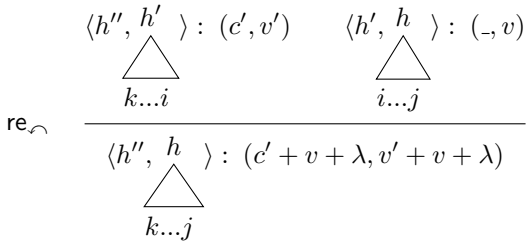


Figure 4: Example of shift-reduce with dynamic programming: simulating an edge-factored model. GSS is implicit here, and $\mathsf{re}_{\curvearrowleft}$ case omitted. Here $\lambda = sc_{\mathsf{sh}}(h'', h') + sc_{\mathsf{re}_{\curvearrowleft}}(h', h)$.

More formally we have:

Theorem 2 (Optimality of Algorithm 1)**.** *The first goal derivation popped off the priority queue is the optimal parse.*

Proof. (Proof by Contradiction.) Assume $\exists x$, x is the a goal derivation and $x \prec x_m$. Based on Stepwise Completeness of Theorem 1, $x \in C_{m-1}$, thus x has already been popped out, which contradicts that x_m is the first popped out goal derivation. □

Furthermore, we can see our lazy expansion version, i.e., Algorithm 2, is also optimal. The key observation is that we delay the reduction of derivation x' and a derivation of right states $\mathcal{R}([x']_\sim)$ (Line 11 of Algorithm 1), until shifted derivation, $x = \mathsf{sh}(x')$, is popped out (Line 11 of Algorithm 2). However, this delayed reduction will not generate any derivation y, s.t. $y \prec x$, because, based on our deductive system (Figure 2), for any such kind of reduced derivations y, $y.pre = x'.pre + x'.sh + y.re + y.ins$, while $x.pre = x'.pre + x'.sh$.

5.2 Analysis of Time and Space Complexity

Following Huang and Sagae (2010) we present the complexity analysis for our DP best-first parsing.

Theorem 3. *Dynamic programming best-first parsing runs in worst-case polynomial time and space, as long as the atomic features function satisfies:*

- **bounded***: $\forall$ derivation x, $|\widetilde{\mathbf{f}}(x)|$ is bounded by a constant.*
- **monotonic***:*

– **horizontal**: $\forall k,\ \mathbf{f}_k(s) = \mathbf{f}_k(t) \Rightarrow \mathbf{f}_{k+1}(s) = \mathbf{f}_{k+1}(t)$, *for all possible trees* s, t.
– **vertical**: $\forall k,\ \mathbf{f}_k(s^\frown s') = \mathbf{f}_k(t^\frown t') \Rightarrow \mathbf{f}_k(s) = \mathbf{f}_k(t)$ *and* $\mathbf{f}_k(s^\frown s') = \mathbf{f}_k(t^\frown t') \Rightarrow \mathbf{f}_k(s') = \mathbf{f}_k(t')$, *for all possible trees* s, s', t, t'.

In the above theorem, boundness means we can only extract finite information from a derivation, so that the atomic feature function $\widetilde{\mathbf{f}}(\cdot)$ can only distinguish a finite number of different states. Monotonicity requires the feature representation $\mathbf{f}_k$ subsumes $\mathbf{f}_{k+1}$. This is necessary because we use the features as signature to match all possible left states and right states (Equation 1). Note that we add the vertical monotonicity condition following the suggestion from Kuhlmann et al. (2011), which fixes a flaw in the original theorem of Huang and Sagae (2010).

We use the edge-factored model (Eisner, 1996; McDonald et al., 2005) with dynamic programming described in Figure 4 as a concrete example for complexity analysis. In the edge-factored model the feature set consists of only combinations of information from the roots of the two top trees s_1, s_0, and the queue. So the atomic feature function is

$$\widetilde{\mathbf{f}}(p) = (i, j, h(p.s_1), h(p.s_0))$$

where $h(s)$ returns the head word index of tree s.

The deductive system for the edge-factored model is in Figure 4. The time complexity for this deductive system is $O(n^6)$, because we have three head indexes and three span indexes as free variables in the exploration. Compared to the work of Huang and Sagae (2010), we reduce the time complexity from $O(n^7)$ to $O(n^6)$ because we do not need to keep track of the number of the steps for a state.

6 Experiments

In experiments we compare our DP best-first parsing with non-DP best-first parsing, pure greedy parsing, and beam parser of Huang and Sagae (2010).

Our underlying MaxEnt model is trained on the Penn Treebank (PTB) following the standard split: Sections 02-21 as the training set and Section 22 as the held-out set. We collect gold actions at different parsing configurations as positive examples from gold parses in PTB to train the MaxEnt model. We use the feature set of Huang and Sagae (2010).

	model score	accuracy	# states	time
greedy	-1.4303	90.08%	125.8	0.0055
beam*	-1.3302	90.60%	869.6	0.0331
non-DP	-1.3269	90.70%	$4,194.4$	0.2622
DP	-1.3269	90.70%	243.2	0.0132

Table 1: Dynamic programming best-first parsing reach optimality faster. *: for beam search we use beam size of 8. (All above results are averaged over the held-out set.)

Furthermore, we reimplemented the beam parser with DP of Huang and Sagae (2010) for comparison. The result of our implementation is consistent with theirs. We reach 92.39% accuracy with structured perceptron. However, in experiments we still use MaxEnt to make the comparison fair.

To compare the performance we measure two sets of criteria: 1) the internal criteria consist of the model score of the parsing result, and the number of states explored; 2) the external criteria consist of the unlabeled accuracy of the parsing result, and the parsing time.

We perform our experiments on a computer with two 3.1GHz 8-core CPUs (16 processors in total) and 64GB RAM. Our implementation is in Python.

6.1 Search Quality & Speed

We first compare DP best-first parsing algorithm with pure greedy parsing and non-DP best-first parsing without any extra constraints.

The results are shown in Table 1. Best-first parsing reaches an accuracy of 90.70% in the held-out set. Since that the MaxEnt model is locally trained, this accuracy is not as high as the best shift-reduce parsers available now. However, this is sufficient for our comparison, because we aim at improving the search quality and efficiency of parsing.

Compared to greedy parsing, DP best-first parsing reaches a significantly higher accuracy, with $\sim$2 times more parsing time. Given the extra time in maintaining priority queue, this is consistent with the internal criteria: DP best-first parsing reaches a significantly higher model score, which is actually optimal, exploring twice as many as states.

On the other hand, non-DP best-first parsing also achieves the optimal model score and accuracy.

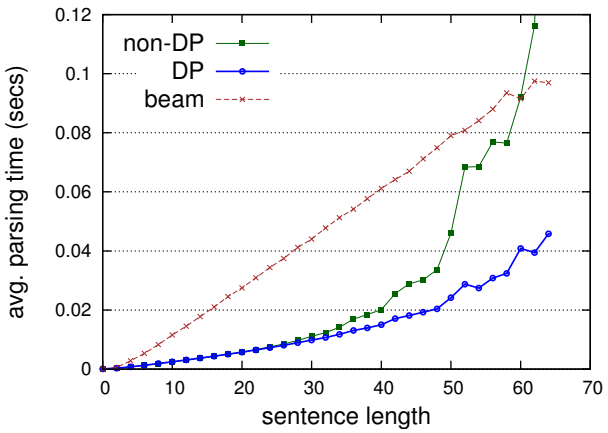


Figure 5: DP best-first significantly reduces parsing time. Beam parser (beam size 8) guarantees linear parsing time. Non-DP best-first parser is fast for short sentences, but the time grows exponentially with sentence length. DP best-first parser is as fast as non-DP for short sentences, but the time grows significantly slower.

However, it explores $\sim$17 times more states than DP, with an unbearable average time.

Furthermore, on average our DP best-first parsing is significantly faster than the beam parser, because most sentences are short.

Figure 5 explains the inefficiency of non-DP best-first parsing. As the time complexity grows exponentially with the sentence length, non-DP best-first parsing takes an extremely long time for long sentences. DP best-first search has a polynomial time bound, which grows significantly slower.

In general DP best-first parsing manages to reach optimality in tractable time with exact search. To further investigate the potential of this DP best-first parsing, we perform inexact search experiments with bounded priority queue.

6.2 Parsing with Bounded Priority Queue

Bounded priority queue is a very practical choice when we want to parse with only limited memory.

We bound the priority queue size at 1, 2, 5, 10, 20, 50, 100, 500, and 1000, and once the priority queue size exceeds the bound, we discard the worst one in the priority queue. The performances of non-DP best-first parsing and DP best-first parsing are illustrated in Figure 6 (a) (b).

Firstly, in Figure 6 (a), our DP best-first parsing reaches the optimal model score with bound 50, while non-DP best-first parsing fails even with bound 1000. Also, in average with bound 1000, compared to non-DP, DP best-first only needs to explore less than half of the number of states.

Secondly, for external criteria in Figure 6 (b), both algorithms reach accuracy of 90.70% in the end. In speed, with bound 1000, DP best-first takes $\sim 1/3$ time of non-DP to parse a sentence in average.

Lastly, we also compare to beam parser with beam size 1, 2, 4, 8. Figure 6 (a) shows that beam parser fails to reach the optimality, while exploring significantly more states. On the other hand, beam parser also fails to reach an accuracy as high as best-first parsers. (see Figure 6 (b))

6.3 Simulating the Edge-Factored Model

We further explore the potential of DP best-first parsing with the edge-factored model.

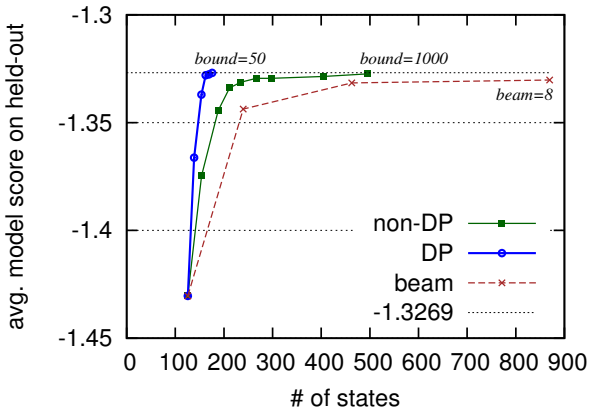
The simplified feature set of the edge-factored model reduces the number of possible states, which means more state-merging in the search graph. We expect more significant improvement from our DP best-first parsing in speed and number of explored states.

Experiment results confirms this. In Figure 6 (c) (d), curves of DP best-first diverge from non-DP faster than standard model (Figure 6 (a) (b)).

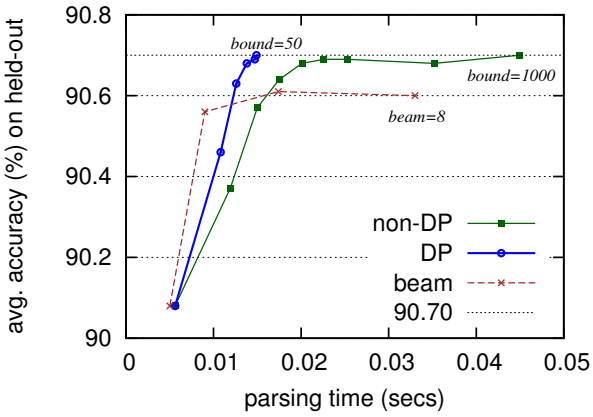
7 Conclusions and Future Work

We have presented a dynamic programming algorithm for best-first shift-reduce parsing which is guaranteed to return the optimal solution in polynomial time. This algorithm is related to best-first Earley parsing, and is more sophisticated than best-first CKY. Experiments have shown convincingly that our algorithm leads to significant speedup over the non-dynamic programming baseline, and makes exact search tractable for the first-time under this model.

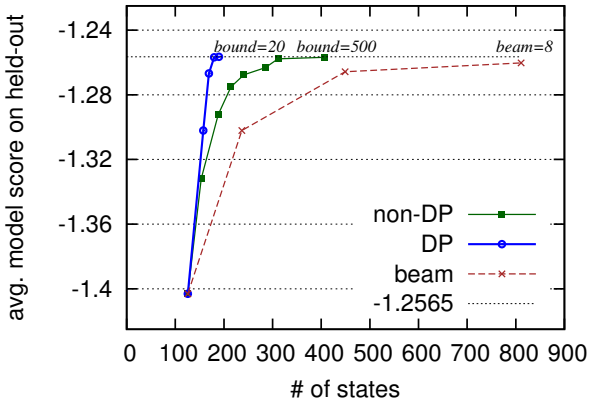
For future work we would like to improve the performance of the probabilistic models that is required by the best-first search. We are also interested in exploring A* heuristics to further speed up our DP best-first parsing.



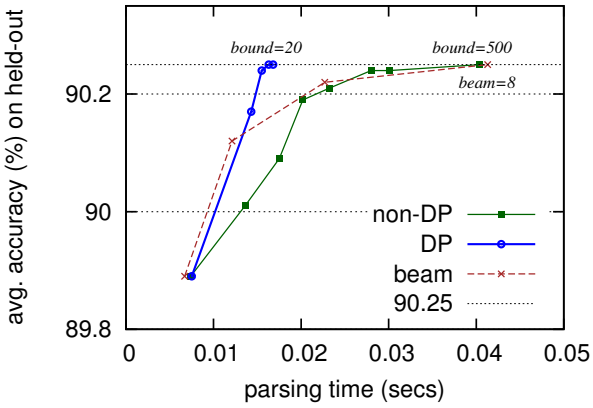
(a) search quality vs. # of states



(b) parsing accuracy vs. time



(c) search quality vs. # of states (edge-factored)



(d) parsing accuracy vs. time (edge-factored)

Figure 6: Parsing performance comparison between DP and non-DP. (a) (b) Standard model with features of Huang and Sagae (2010). (c) (d) Simulating edge-factored model with reduced feature set based on McDonald et al. (2005). Note that to implement bounded priority queue we use two priority queues to keep track of the worst elements, which introduces extra overhead, so that our bounded parser is slower than the unbounded version for large priority queue size bound.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey.

Sharon A Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*.

Dan Klein and Christopher D Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of HLT-NAACL*.

Dan Klein and Christopher D Manning. 2005. Parsing and hypergraphs. In *New developments in parsing technology*, pages 351–372. Springer.

Donald Knuth. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1).

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, pages 135–143.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Adam Pauls and Dan Klein. 2009. Hierarchical search for parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 557–565. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of ACL*.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*.

Exploiting language models for visual recognition

Dieu-Thu Le
DISI, University of Trento
Povo, 38123, Italy
dle@disi.unitn.it

Jasper Uijlings
DISI, University of Trento
Povo, 38123, Italy
jrr@disi.unitn.it

Raffaella Bernardi
DISI, University of Trento
Povo, 38123, Italy
bernardi@disi.unitn.it

Abstract

The problem of learning language models from large text corpora has been widely studied within the computational linguistic community. However, little is known about the performance of these language models when applied to the computer vision domain. In this work, we compare representative models: a window-based model, a topic model, a distributional memory and a commonsense knowledge database, ConceptNet, in two visual recognition scenarios: human action recognition and object prediction. We examine whether the knowledge extracted from texts through these models are compatible to the knowledge represented in images. We determine the usefulness of different language models in aiding the two visual recognition tasks. The study shows that the language models built from general text corpora can be used instead of expensive annotated images and even outperform the image model when testing on a big general dataset.

1 Introduction

Computational linguistics have created many tools for automatic knowledge acquisition which have been successfully applied in many tasks inside the language domain, such as question answering, machine translation, semantic web, etc. In this paper we ask whether such knowledge generalizes to the observed reality outside the language domain, where we use well-known image datasets as a proxy for observed reality.

In particular, we aim to determine which language model yields knowledge that is most suitable for use in Computer Vision. Therefore we test a variety of language models and a linguistically mined knowledge base within two computer vision scenarios:

Human action recognition : Recognizing <subject, verb, object> triples based on objects (e.g., car, horse) and scenes (the place that the actions occur, e.g., countryside, forest, office) recognized in images. In this scenario, we only consider images with human actions so the "human" subject is always present.

Objects in context : Predicting the most likely identity of an object given its context as expressed in terms of co-occurring objects.

Computer vision can greatly benefit from natural language processing as learning from images requires a prohibitively expensive annotation effort. A major goal of natural language processing is to obtain general knowledge from text and in this paper we test which model provides the best knowledge for use in the visual domain.

Within the two visual scenarios, we compare three state-of-the-art language models and a knowledge base: (1) A window-based model, which counts co-occurrence frequencies within a fixed window; (2) R-LDA (Séaghdha, 2010), an extension of LDA that enables generation of joint probabilities; (3) TypeDM (Baroni and Lenci, 2010), a strong Distributional Memory model; (4) ConceptNet (Speer and Havasi, 2013), an automatically generated semantic graph containing concepts with their relations.

We test the language models in two ways: (1) We directly compare the statistics of the linguistic models with statistics extracted from the visual domain.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 769–779,
Seattle, Washington, USA, 18-21 October 2013.

(2) We compare the linguistic models inside the two computer vision applications, leading to a direct estimation of their usefulness.

To summarize, our main research questions are: (1) Is the knowledge from language compatible with the knowledge from vision? (2) Can the knowledge extracted from language help in computer vision scenarios?

2 Related Work

Using high level knowledge to aid image understanding has become a recent interest in the computer vision community. Objects, actions and scenes are detected and localized in images using low-level features. This detection and localization process is guided by reasoning and knowledge. Such knowledge is employed to disambiguate locations between objects in (Gupta and Davis, 2008). From the defined relationships between nouns (e.g., above, below, brighter, smaller), the system constrains which region in an image corresponds to which object/noun. Similarly, (Srikanth et al., 2005) exploit ontologies extracted from WordNet to associate words and images and image regions. (Yu et al., 2011) employ relations between scenes and objects introducing an active model to recognize scenes through objects. The reasoning knowledge limits the detector to search for an object within a particular region rather than on the whole image.

Language models have also been employed to generate descriptive sentences for images. (Ushiku et al., 2012) introduce an online learning method for multi-keyphrase estimation to generate a sentence using a grammar model to describe an image. Similarly, from objects and scenes detected in an image, (Yang et al., 2011) estimated a sentence structure to generate a sentence description composed of a noun, verb, scene and preposition.

The studies most similar to ours are (Teo et al., 2012) and (Lampert et al., 2009). In (Teo et al., 2012), the Gigaword corpus is used to extract relationships between tools and actions (e.g., knife - cut, cup - drink) by counting their co-occurences. These relationships are used to constrain and select the most plausible actions within a predefined set of actions in cooking videos. Instead of using this knowledge as a guidance during recognition, we compare different language models and build a general framework that is able to detect unseen actions through their components (verb - object - scene), hence our method does not limit the number of actions in images. (Lampert et al., 2009) use attributes of nouns (e.g., an animal: white, eat fish, water, etc.). They can detect animals without having seen training examples by manually defining the attributes of the target animal. In this work, rather than relying on manual definitions, our aim is to find the best language models built automatically from available corpora to extract relations from natural language.

Currently, human action recognition is popular and mostly studied in video using the Bag-of-Visual-Words method (Delaitre et al., 2010; Everts et al., 2013; Kuehne et al., 2012; Reddy and Shah, 2012; Wang et al., 2013). In this method one extracts small local visual patches of, say, 24 by 24 pixels by 10 frames at every 12^{th} pixel at every 5^{th} frame. For each patch local gradients or local movement (optical flow) histograms are calculated. Then these local visual features are mapped to abstract, predefined "visual words", previously obtained using k-means clustering on a set of random features. While results are good, there are two main drawbacks with this approach. First of all, human actions are semantic and more naturally recognized through their components (human, objects, scene) rather than through a bag of local gradient/motion patterns. Hence we use a component-based method for human action recognition. Second, the number of possible human actions is huge (the number of objects times the number of verbs). Obtaining annotated visual examples for each action is therefore prohibitively expensive. So we learn from language models how components combine into human actions.

3 Two Visual Recognition Scenarios

We now describe the two computer vision scenarios: human action recognition and objects in context.

3.1 Human Action Recognition

We want to identify a human action, defined as a <subject, verb, object> triple. We do this by recognizing the human, the object, and the scene and then determine the most likely verb based on these components. Scenes are only used here as features for

predicting/disambiguating the human action and the final task is to define the human action triple. As in most work in human action recognition, we simplify the problem by considering only images in which human actions occur. This means that a human is always present, leaving the problem of predicting the verb given the object and the scene. While this may seem like a strong assumption, the possibility of having no action in the image at all is largely unexplored in computer vision due to its difficulty.

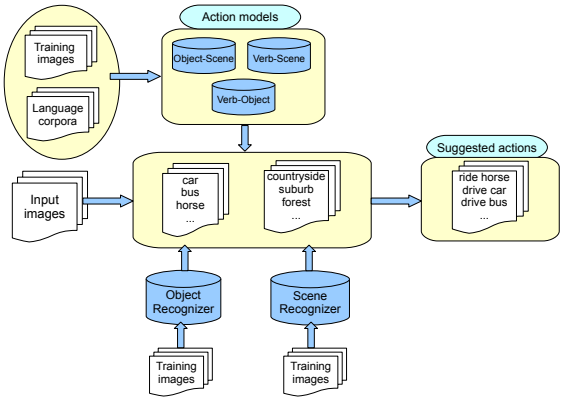


Figure 1: Human action suggestion: based on the objects and scenes recognized in an image, the system suggests the most plausible actions. The action models provide the relationships between objects - scenes - verbs

3.1.1 Human action recognition framework

Our general action recognition framework is presented in Figure 1. Given an image, an object recognizer will predict the probability of each object (e.g, bike, horse) presented in that image. Furthermore, a scene recognizer will provide the probabilities of each scene (e.g., countryside, suburb, forest) given the image. The action model is composed of the conditional probabilities that relate verbs, objects and scenes, which have been learned from training images or language corpora. Given the object and scene probabilities recognized in the image, the action model will guide the action prediction process and finally, the system will suggest the most proper actions (e.g., ride horse, drive car). We will now describe each component in detail:

- **Object and scene recognizers:** To train the object recognizer, we use a set of images where objects have been annotated with bounding boxes (to specify objects' locations). We follow the state-of-the-art method of (Uijlings et al., 2013). The method is based on multiple hierarchical segmentations to sample a limited set of high quality object locations in terms of bounding boxes. A Bag-of-Visual-Words method (Uijlings et al., 2010) is applied to these boxes to localize and recognize objects. For the scene recognizer, we trained the same Bag-of-Visual-Words method on complete images on a dataset annotated with 15 scenes. In both cases we use Support Vector Machines to learn the object/scene models. We use Platt's sigmoid function to obtain the final conditional probabilities $P(o_j|I)$ and $P(s_k|I)$.

- **Action models:** The model captures the relationship between Object - Scene, Verb - Scene and Verb - Object: the probability of an object given a scene $P(o_j|s_k)$, a verb given an object $P(v_i|o_j)$, and a verb given a scene $P(v_i|s_k)$. In one experiment we learn the probabilities from the training images, where each image has been annotated with an object, a verb (of an action) and a scene. All three probabilities are computed using frequency counts in the training set, for example:

$$P(o_j|s_k) = \frac{\#\text{images having } o_j, s_k}{\#\text{images having } s_k} \quad (1)$$

  We aim to replace this learning from annotated training images, which are expensive to obtain, with learning from language corpora. The details of how to extract the probability distributions from language models are explained in section 4.

3.1.2 Component integration

To combine these components in the framework, we use an energy-based model (Lecun et al., 2006) visualized in Figure 2, which includes the image I (an observed variable) and object O, scene S, and verb V. This energy-based formulation allows us to set different weights for energies which come from disparate sources (i.e. language and vision) using Gibbs measure.

Now given an image I, we can compute the score function $\mathbb{S}(a_{ij}; I)$ of an action a_{ij} as:

$$\mathbb{S}(a_{ij}; I) = \mathbb{S}(v_i, o_j; I) = \frac{1}{Z} \exp\left(-\sum_{F \in \mathcal{F}} E_F^{ij}\right),$$

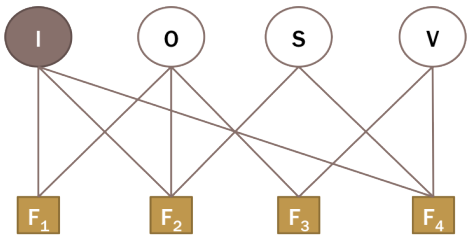


Figure 2: An energy-based model for action recognition

where we define each energy function E_F^{ij} to give lower energies to correct answers and higher energies to incorrect ones, with $\mathcal{S}$ is the set of all scenes:

$$E_{F_1}^{ij}(O, I) = -w_{F_1} \log P(o_j|I) \quad (2)$$

$$E_{F_2}^{ij}(S, I) = -w_{F_2} \log \sum_{S \in \mathcal{S}} P(o_j|S) \times P(S|I) \quad (3)$$

$$E_{F_3}^{ij}(V, O) = -w_{F_3} \log P(v_i|o_j) \quad (4)$$

$$E_{F_4}^{ij}(V, S) = -w_{F_4} \log \sum_{S \in \mathcal{S}} P(v_i|S) \times P(S|I) \quad (5)$$

Let P_i be the position of the correct action in the ranked list of predicted actions for a certain image I_i. The ranked list is sorted in the order of the score $\mathbb{S}$. We evaluate human action recognition in terms of this position average over all images, which we call Average Ranking (AR). Therefore we use Average Ranking as our loss-function:

$$\mathcal{L}(w_F) = AR_N = \frac{1}{N} \sum_{i=0}^{N} P_i. \quad (6)$$

Training the energy model involves finding the factors w_F^* that minimizes the loss:

$$w_F^* = \underset{w_F}{\operatorname{argmin}}\, \mathcal{L}(w_F) \quad (7)$$

As we have only four parameters to learn in our energy model, we do this by performing an exhaustive search and cross validation. We require $w_F \in \{0.0, 0.1, 0.2, ..., 0.9\}$ and set the constraint $\sum_{F \in \mathcal{F}} w_F = 1$. We note that the factor graph formulation of our framework would allow us to use more advanced learning algorithms. We plan to look into this once the model becomes more complex by adding, for example, information about the position of the objects and the human.

3.1.3 Dataset

Recently, researchers have released many image action datasets such as the 7 everyday actions (Delaitre et al., 2010), the Stanford 40 action dataset (Yao et al., 2011), the PASCAL action classification competition (Everingham et al., 2012), and the 89 action dataset (Le et al., 2013). The 89 action dataset was originally created for the recognition of 20 objects. Afterwards also actions were annotated. Therefore, the actions occurring with these objects are mostly unbiased, unlike in other action datasets. Hence we choose to use the 89 action dataset.

In the 89 action dataset, every image has been annotated with human actions, where each action is composed of a verb and an object. We additionally annotated every image with one of the 15 scenes from the 15 scene dataset (Lazebnik et al., 2006).

3.2 Objects in Context

Our other computer vision scenario is about objects in context. Context is useful in visual recognition for two reasons: Firstly, context can significantly reduce the number of possible object categories simplifying the problem. Secondly, when the object appearance is inconclusive for its identity, context can be used for disambiguation. For example, a grey rectangle on a desk may be recognized as a pen, while a grey rectangle on a table may be recognized as a knife. As the recognition systems are not always reliable, the use of context can greatly improve results.

For this scenario we choose a theoretical setting in which we want to predict the identity of one object given that the identities of all other objects in the image are known. We believe that our main conclusions on the linguistic models will transfer to a practical computer vision application where visual recognition systems predict the object identities.

Formally, we can describe this scenario as follows: Given an image I with N objects $\mathcal{O} = \{o_1, o_2, \cdots, o_N\}$, we want to predict the identity of object o_i given all other objects $\mathcal{O} \setminus o_i$. In this paper we use a Naive Bayes assumption, leading to:

$$P(o_i|\mathcal{O} \setminus o_i) = \frac{P(\mathcal{O} \setminus o_i|o_i) \times P(o_i)}{P(\mathcal{O} \setminus o_i)} \approx P(o_i) \times \prod_{o_j \in \mathcal{O} \setminus o_i} P(o_j|o_i). \quad (8)$$

In this scenario, we need conditional relations $P(o_j|o_i)$ and priors. We obtain these from language data or from images directly.

3.2.1 Dataset

For the objects in context scenario, we use the SUN object dataset (Xiao et al., 2010), which contains more than 16 thousand images, more than 79,000 objects whose locations are annotated using polygons. The dataset has been annotated by various people who could choose their own object categories, leading to duplicate categories such as "building" and "buildings", "person" and "person walking". Furthermore, for some images large parts are not annotated leading to an incomplete context. We therefore cleaned the object categories (mapping from around 7,500 objects to over 700 unique object categories) and considered only images whose content was sufficiently annotated.

In our experiments, we used the predefined training and testing parts of the SUN dataset and obtained around 4,500 images for learning the object relations and 10,600 images for testing the object prediction. We obtain conditional probabilities $P(o_j|o_i)$ from frequency counts.

4 Language models & distribution extraction

To extract probability distributions from texts, we use ConceptNet, the Window2 and 20 model, TypeDM and R-LDA. We will now describe how we estimate the four conditional probabilities $P(V|O), P(V|S), P(O|S), P(O|O)$ needed in the two visual scenarios for each language model.

4.1 ConceptNet

ConceptNet (Speer and Havasi, 2013) is a large semantic graph containing concepts and relations between them. It includes everyday basic, cultural and scientific knowledge, which have been automatically extracted from Internet using predefined rules. In this work we use the most current version, ConceptNet 5. As it was mined from free text using rules, the database has uncontrolled vocabulary and contains many false/nonesense statements.

To extract relations from ConceptNet5, we first examine all relations in the database and define those that are relevant to our scenarios (Figure 3). For

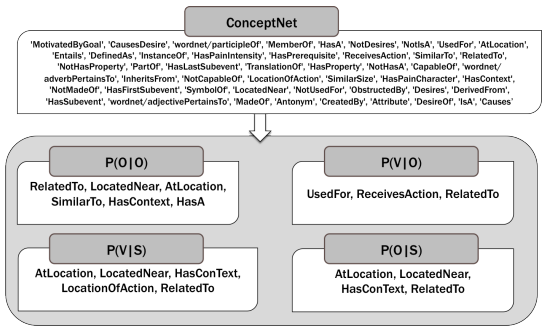


Figure 3: List of relations in ConceptNet

example, for the conditional probability of objects given scenes, relations such as "At Location", "Located Near" are extracted. For the human action recognition scenario, we used a list of 19 objects, 15 scenes and around 5 thousand verbs for computing $P(V|O), P(O|S), P(V|S)$. For the objects in context scenario, we used 700 objects for computing $P(O|O)$. Examples of relations extracted from ConceptNet are illustrated in Table 1, such as: Oil - Located near - Car, Horse - Related to - Zebra. From these relations, we define the four conditional probabilities using their frequency counts. For example, to compute the conditional probability of an object given a scene $P(o_i|s_j)$, we extract all triples having the form <object, rel, scene>, where "rel" can be "AtLocation", "LocatedNear", etc.

$$P(o_i|s_j) = \frac{freq(< o_i, rel, s_j >)}{\sum_{o_m \in \mathcal{O}} freq< o_m, rel, s_j >} \quad (9)$$

4.2 Window model

One of the most famous and basic statistical model is based on counting co-occurrences within a window of fixed width, which follows the tradition of hyperspace analogue to language (Lund and Burgess, 1996). We took the Window2, 20 models which have been built in (Bruni et al., 2012) using the ukWaC (1.9B tokens) and Wackypedia (820M tokens). As the Window2 model only looks at 2 words on the left and right of the current one, it reflects the relationships between words occurring near each other, while the Window20 searches for a broader view of how words are related to each other. The weights of each pairs of words are calculated using the Local Mutual Information (LMI). To compute the conditional probabilities, we use the LMI scor-

LocatedNear		RelatedTo		UsedFor		AtLocation	
oil car	seatbelt car	horse zebra	plant garden	bottle store_liquid	horse race	bus city	car city
chair your_bottom	chair school	horse pony	sheep baa	boat fish	table eat_off_of	bike street	dog city
plant everywhere	muzzle dog	plant green	sheep cloud	dog companionship	chair rest	bird countryside	dog street
trailer car	dog bark_bone	boat ship	cow bull	horse riding	bus travel	car street	chair city
salt table	horse cowboy	chair table	horse riding	chair sitting	table eat_meal	cat store	bus city
stool table	carriage horse	dog wolf	sheep farm	chair sit_on	boat travel	car street	chair store
pasture cow	horse fence	dog cat	cow milk	car transportation	bottle hold_liquid	car street	bicycle store
cat dog	whisker cat	sheep lamb	table desk	sheep wool	boat float_on_water	bird forest	chair store
horse zebra	desk chair	sheep wool	cat feline	table put_thing_on	table eat_at	car city	bottle store
cat household	train railroad	dog a wolf	dog canine	boat travel_on_water	cat catch_mouse	table kitchen	chair office
horsehair horse	sheep wool	cat dog	plant flower	chair sit	cow milk	chair office	chair city

Table 1: Examples of relations extracted from ConceptNet 5

ing function provided by the models, for example:

$$P(v_i|o_j) = \frac{LMI_{v_i,o_j}}{\sum_{v_m \in \mathcal{V}} LMI_{v_m,o_j}} \quad (10)$$

4.3 Distributional Memory

Distributional Memory (Baroni and Lenci, 2010) (DM) is a multi-purpose framework for semantic modeling. This model is more complex than the Window models because it exploits different degrees of lexicalization for each relation. Distributional information is extracted as a set of weighted <word-link-word> tuples obtained from a dependency parse of corpora. In the Window model the relation between each word pair is decided by their co-occurences within a sliding window, while in DM this relation is defined by distributional properties of the two words. These distributional properties are based on a syntactic relation or lexico-syntactic pattern that links the two words. For example, the tuple <marine, use, bomb> encodes that marine co-occurs with bomb in the corpus, and the word "use" specifies the type of the syntagmatic link.

Distributional Memory contains three different models, corresponding to different ways to construct the weighted structure through the "link". The first model, LexDM is the most heavily lexicalized model with the most variety of links, whereas the DepDM has the minimum degree of lexicalization, thus having the smallest number of links. TypeDM, which was reported to achieve the best performance in different tasks including selectional preferences, is laying somewhere in the middle of the other two models. It shares the same lexical information as in LexDM but use a different scoring function, which focuses on the variety of surface forms, rather than the frequency of a link. Hence we choose the best model, TypeDM, to learn the relationships between verbs, objects and scenes. As in the window model, we compute conditional probabilities using the LMI scores provided by the model (Equation 10).

4.4 R-LDA

To model the relationships between verbs, objects and scenes, we adapt the R-LDA model (Séaghdha, 2010) (ROOTH-LDA), which has been used for the selectional preference task in order to obtain conditional probabilities of two words. Each relation m of $< w_1, w_2 >$ is generated by picking up a distribution over topics, then both elements of the relation m share the same topic assignment z_m, which keep two different w_1-topic and w_2-topic distributions sharing the same topic (Figure 4). The models are estimated by Gibbs sampling following (Heinrich, 2004). It is also noted that these models are generative, hence they also predict the probabilities of tuples that do not occur in the corpus.

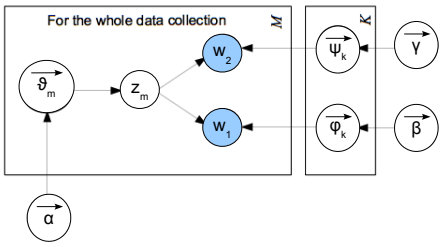


Figure 4: Generative graphical model of R-LDA: modeling the relations between two words

To model the relations between objects and verbs, we follow the data preparation in (Le et al., 2013), using the British National Corpus (BNC) which has been preprocessed and parsed using TreeTagger and Maltparser. Verbs are heads of sentences while objects are either direct or indirect objects related to those verbs by the parser. For the relations between verbs and scenes, we consider also verbs as heads

Topic 8:				Topic 14:				Topic 0:				Topic 54:			
Noun		Verb		Noun		Verb		Noun		Noun		Noun		Noun	
people	0.0208	have	0.157	year	0.0154	win	0.109	attention	0.0172	study	0.01	decision	0.02	case	.0176
job	0.0167	work	0.108	Cup	0.0093	have	0.099	model	0.0147	research	0.0123	view	0.0244	fact	.0096
work	0.0156	make	0.0262	team	0.0086	beat	0	study	0.014	work	0.0111	question	0.018	question	.0096
class	0.014	take	0.0244	race	0.00772	take	0.025	role	0.0139	chapter	0.0085	issue	0.0124	law	.0096
worker	0.0123	find	0.0194	season	0.0062	lose	0.0211	account	0.013	problem	0.0075	evidence	0.0104	decision	.0092
staff	0.0111	pay	0.0156	time	0.006	run	0.0152	analysis	0.0123	issue	0.006	point	0.0099	time	.0067
group	0.0089	say	0.0146	world	0.0058	finish	0.0135	aspect	0.012	system	0.0065	reason	0.0096	issue	.0062
way	0.0086	get	0.0136	game	0.0055	make	0.0127	problem	0.0106	area	0.006	statement	0.0086	evidence	.00617
service	0.008	leave	0.0114	champion	0.0053	lead	0.0122	effect	0.0105	process	0.006	doubt	0.008	interest	.0059
company	0.0076	run	0.0102	seat	0.0049	follow	0.0098	pattern	0.0103	policy	0.0055	attention	0.0076	point	.0058
day	0.0069	come	0.0101	match	0.0049	qualify	0.008	issue	0.0102	theory	0.00551	matter	0.00738	judge	.0055
number	0.00615	help	0.0088	place	0.0047	compete	0.0074	range	0.0095	way	0.0053	policy	0.006	statement	.005

Table 2: Random R-LDA topics with the relations between Noun-Verb (first 2 columns) and between Noun-Noun (last 2 columns)

of sentences while scenes are all nouns occurring in the same sentence. For the relations between objects and scenes as well as objects and objects, we use all nouns to capture a general model[1]. The statistics of the BNC corpus with their corresponding relations are reported in Table 3.

	#Relations	#Tokens
Verb - Object	3.3M	6.7M
Noun - Noun	19.8M	39.7M
Verb - Noun	83.4M	166.8M

Table 3: The statistics of the dataset used for estimating R-LDA models for each relation type

Samples of topics extracted through R-LDA are illustrated in Table 3. It shows that Noun and Object share many similar terms in the same topic while Noun and Verb sharing the same topics tend to go often together (e.g., win, cup, beat, race).

5 Experiments

In this section we want to answer our two main research questions: (1) Is knowledge from language compatible with knowledge from vision? (2) Can we use knowledge extracted from language in computer vision scenarios?

5.1 Is knowledge from language and vision compatible?

In this section we compare statistics mined from texts with those mined from visual sources. Ideally, we want statistics from the language models to follow those of the image model, even though not all statistics from images can be reliably measured due to insufficient data. Therefore, we measure how well the estimated language models fit the estimated visual distributions using the the χ^2-distance:

$$\mathcal{X}^2 = \sum_{i=1}^{N} \frac{({P_I}^i - {P_L}^i)^2}{{P_I}^i} \tag{11}$$

where P_I and P_L are the probability distribution obtained from the image data and language models respectively.

For the conditional probabilities $P(V|O)$, $P(V|S)$, and $P(O|S)$ we compare language models with image statistics extracted from the 89 human action dataset. Table 4 shows the results. For the relations between verb and scene $P(V|S)$, there is not much fluctuation among different language models. For objects and scenes $P(O|S)$, R-LDA is closest to the image model. This is because R-LDA is good at measuring contextual and indirect relations by design, which is the case for object-scene relations. This also explains why TypeDM and Window20 are

Chi statistics	P(V|O)	P(V|S)	P(O|S)
R-LDA	17.8	11.6	**11.9**
Window2	11.6	**11.4**	32.6
Window20	11.7	**11.4**	23.7
TypeDM	**11.5**	13.3	23.2
ConceptNet	17.5	11.5	34.4

Table 4: $\mathcal{X}^2$ distance for relations between verbs, objects, scenes from different language models to image data

[1]Different from objects and verbs, which can be defined explicitly from the parsed corpora, scenes can only be defined from more restrained rules (e.g., followed by some prepositions), so here we take all nouns to have the most general model.

further away from the image model, followed by the Window2 model. Instead, human actions are found in language as the relation between verbs and their direct linguistic objects. Indeed, TypeDM is closest to the image model for $P(V|O)$ as it makes explicit use of this linguistic link. The Window2 and 20 models are almost as close to the image model for $P(V|O)$, while R-LDA is considerably further away due to its contextual nature. Finally, ConceptNet is the furthest away from the image model. To conclude, TypeDM is best for modelling direct verb-object relations, while R-LDA is better at capturing the more contextual object-scene relations.

To look closer at the difference between the statistics obtained from the image and language data, we give an example of the conditional probabilities of an object given a scene $P(O|S)$ in Figure 5. We see that the distribution extracted from language (TypeDM) is much smoother and contains more relations than the image model since it has been trained on general and large text corpora. The distribution from image data on the other hand is more sparse and tailored to this specific dataset. For example, given a "store", the probability that there is a "table" is 1, given "highway", the probability of a "car" is also 1 in the image dataset, while the highest conditional probability of the language model is only less than 60%.

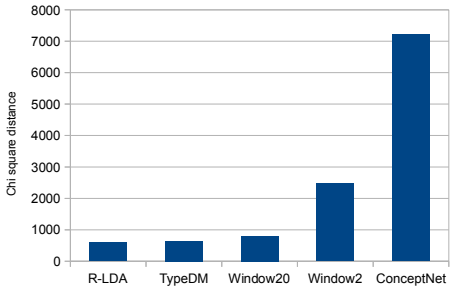


Figure 6: χ^2-distances between the tested language models and the image model for conditional probabilities of objects $P(O|O)$.

For the relations between objects and objects, we use the SUN dataset, which is much bigger and more general than the action dataset. As shown in Figure 6, R-LDA is most similar to the image model, closely followed by TypeDM and the Window20 model. All these three models are good at capturing broad contextual relations. The Window2 model has a significantly larger distance to the image model as it captures a narrow context of 2 words, which is apparently not enough to find co-occurences of objects. ConceptNet is the most inconsistent with this image data since not enough objects and their relations are extracted from it. To sum up, R-LDA achieves the best performance in modeling the relations between objects and objects among all language models.

5.2 Language Models for Visual Recognition

To measure the performance of the two visual recognition scenarios, we use the position p^i of the correct action found in the ranked list for each image i.

We report the average ranking over all images (AR_I) and over all objects or actions (AR_O, AR_A):

$$AR_I = \frac{\sum_{i=0}^{N} p^i}{N}; AR_O = \frac{\sum_{j=0}^{N_o} p_o^j}{N_o} \quad (12)$$

where N is the number of images, N_o is the number of objects and p_o^j is the average rank of all images having object j. The average rank over all actions AR_A is defined similarly to AR_O. The average rank over all image measures the performance over the image dataset, but infrequent objects/events have little impact on this performance. The average rank over objects or actions gives more weight to rare examples.

5.2.1 Human Action Recognition

We evaluate the performance of human action recognition in images based on objects and scenes individually, and then study the integration of them. The training set contains 1,104 images (for training the image relations) and the test set has 710 images. First, we test how the model predicts an action knowing the actual object and/or scene appearing in an image (given object/scene gold standard), i.e., O_{gs}, S_{gs} and $O_{gs}S_{gs}$ in the settings. After that, we test a complete model which is based on the output of our object recognizer and our scene recognizer (O_{rec}, S_{rec}, $O_{rec}S_{rec}$).

For each setting, we try different action models, either learnt from the training images (Image), or from each of the language models (TypeDM, R-LDA, Window2, Window20, ConceptNet).

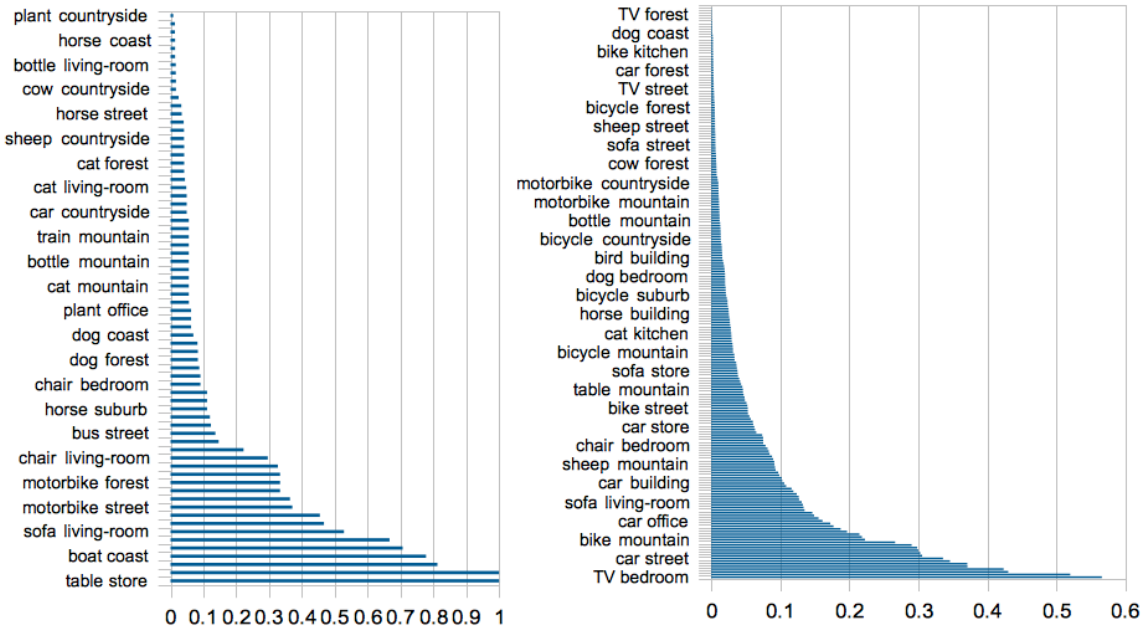


Figure 5: Probability distributions of scene over object extracted from: (left) image dataset; (right) TypeDM model (as there are many <object - scene> relations, only a few are shown on the Y-axises). The number of relations in the TypeDM is much bigger than in the image model, which shows a more general model than the image one.

Table 5 presents the average ranking over all images. Results show that the action model learnt directly from the training images achieves the best performance in all settings, even if we give more weight to infrequent actions by taking the average ranking over all actions, as presented in Table 6. One explanation may be that the action dataset has a limited domain of only 19 objects, while the language models were learnt from broad knowledge (See Figure 5). Another possibility is that verbs used for describing actions in images are more specific than verbs used in language. For example, in language one would "use the car", while in images such action would be labelled "drive a car".

	Image	**TypeDM**	**R-LDA**	**Window2**	**Window20**	**C.Net**
O_{gs}	0.3	16.1	63.4	16.4	18.3	86.1
O_{rec}	14.9	26.9	66.7	44.7	54.9	115.6
S_{gs}	35.7	181.7	174.9	168.5	174.8	252.5
S_{rec}	46.8	250.5	348	190.2	189.8	241.2
$O_{gs}S_{gs}$	**0.28**	**10.2**	**15.2**	**13.8**	**13.6**	**81.9**
$O_{rec}S_{rec}$	13.6	26.9	66.7	44.7	54.9	115.6

Table 5: Average rank over all images AR_I of the human action recognition using different settings: O_{gs}, O_{rec} use only objects (gold standard and object recognizer); S_{gs}, S_{rec} use only scenes, $O_{gs}S_{gs}$ and $O_{rec}S_{rec}$ integrate both objects and scenes together

If we look at the performance of the language models, TypeDM performs best by a significant margin. This makes sense, as the most powerful term for predicting an action is obviously $P(V|O)$, and we saw earlier that TypeDM produces probabilities $P(V|O)$ which are closest to the image model. For the same reason, the second and third best language model are the Window2 and Window20 models, although their performance is significantly lower when using the predictions for objects and/or scenes. This is somewhat surprising considering that TypeDM, Window2 and Window20 are all very close in distance to the image model. Of course, the distance is just an indication. R-LDA performs poorly because it is much more contextual. Finally, ConceptNet performs the worst.

Another observation is that using the scene identity should theoretically help in human action recognition: Using TypeDM, the use of the gold standard object identity yields an average ranking over all images of 16.1, while using both the scene and object identity yields an average ranking of 10.2, which is significantly better. It means that the use of the scene can disambiguate some actions (e.g. "ride a horse" vs. "feed a horse"). However, when using the recognition system, using the scene does not increase the overall performance. This shows that the

visual recognition system may not be strong enough for recognizing these 15 scenes. Another problem may be the limitation of 15 scenes only: while annotating we frequently found that it was hard for numerous images to put them into one of the 15 scenes. So a bigger scene database may help.

The main problem with most available annotated human action datasets is that they are very restricted and domain-specific. For example, in this dataset with 19 objects and 15 scenes, there are many photos of a person riding a motorbike on rocky mountains as a kind of sport. Consequently, the probability of "riding" given "mountain" learnt from the image dataset is high according to the image data (78%) but is uncommon in general. So the image dataset might be too restricted or biased for general knowledge to work well. In the next section we therefore use a more general dataset.

	Image	**TypeDM**	**R-LDA**	**Window2**	**Window20**	**C.Net**
AR_I	13.6	26.9	66.7	44.7	54.9	115.6
AR_A	16.4	30.8	64.7	45.3	51.9	131.7

Table 6: Average rank over all images vs. actions of the human action recognition using the $O_{rec}S_{rec}$ setting

5.2.2 Objects in Context

For every object in every image in the test set of the SUN database, we guess the identity of an object given the identity of all other objects in the image. In total, there are 78,306 object predictions within 10,652 images.

As shown in Figure 7, the R-LDA model outperforms all other models for both average rank over images and over objects. Interestingly, both R-LDA and TypeDM are better at predicting the correct objects in images than the model learnt from the image training set itself. It shows that for many cases, the relation statistics learnt from language data can help in visual recognition. These language models are even better than the information extracted from general, relatively unbiased image datasets, where annotation is limited. For the limited annotation, this hypothesis is further supported by looking at the average rank over objects, which gives more weight to rarely occurring objects. As seen in Figure 7, all language models except ConceptNet outperform the image model. We conclude that language models

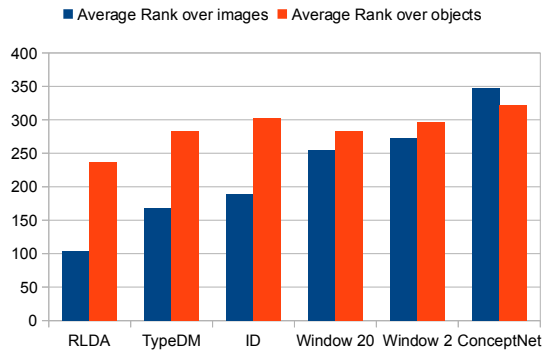


Figure 7: Average rank over all images and objects using different language models and ID (image data)

can aid visual models in large-scale visual recognition problems which use co-occurrence of objects as their context, especially when the annotation is limited, as is often the case.

6 Conclusion

In this paper, we investigated the problem of applying knowledge learnt from language corpora to visual recognition. We compared statistics of various language models mined on general corpora with statistics observed in image datasets. It shows that the generative R-LDA model is good at relating contextual relations (e.g., object - object, object - scene), while the syntactic based distributional model TypeDM is good at representing direct relations such as verb - object in images.

We have evaluated the performance of the language models in two visual scenarios: human action recognition and object prediction. It suggests that the language models need some tailoring when applied to restricted datasets, but for a bigger and more general dataset, the language models even outperform the model learnt from annotated images itself. This shows that language models built from available text corpora can be used for visual recognition instead of expensive annotated image data.

In the future, we want to further investigate the problem of domain adaptation when applying general language models to a new image dataset. This problem can be integrated into the energy-based model during the training phase. We plan to extend work on human action recognition by including the relative position between the human and object in the images.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL. ACL.

Vincent Delaitre, Ivan Laptev, and Josef Sivic. 2010. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2012. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

I. Everts, J. van Gemert, and T. Gevers. 2013. Evaluation of color stips for human action recognition. In *CVPR*.

Abhinav Gupta and Larry S. Davis. 2008. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *Proceedings of the 10th European Conference on Computer Vision*.

Gregor Heinrich. 2004. Parameter estimation for text analysis. Technical report.

H. Kuehne, D. Gehrig, T. Schultz, and R. Stiefelhagen. 2012. On-line action recognition from sparse feature flow. In *VISAPP*.

C.H. Lampert, H. Nickisch, and S. Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *Conference on Computer Vision and Pattern Recognition CVPR*.

S. Lazebnik, C. Schmid, and J. Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Conference on Computer Vision and Pattern Recognition CVPR*, volume 2, pages 2169–2178.

Dieu Thu Le, Raffaella Bernardi, and Jasper Uijlings. 2013. Exploiting language models to recognize unseen actions. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval ICMR*. ACM.

Yann Lecun, Sumit Chopra, Raia Hadsell, Fu J. Huang, G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar Eds. 2006. A tutorial on energy-based learning. In *Predicting Structured Data*.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*.

K. Reddy and M. Shah. 2012. Recognizing 50 human action categories of web videos. In *Machine Vision and Applications*.

Diarmuid Ó. Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 435–444. Association for Computational Linguistics.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*. Springer Berlin Heidelberg.

Munirathnam Srikanth, Joshua Varner, Mitchell Bowden, and Dan Moldovan. 2005. Exploiting ontologies for automatic image annotation. In *Special Interest Group on Information Retrieval SIGIR*. ACM.

C.L. Teo, Yezhou Yang, H. Daume, C. Fermuller, and Y. Aloimonos. 2012. Towards a watson that sees: Language-guided action recognition for robots. In *2012 IEEE International Conference on Robotics and Automation ICRA*.

J R R Uijlings, A W M Smeulders, and R J H Scha. 2010. Real-time Visual Concept Classification. *IEEE Transactions on Multimedia*, 12.

J R R Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. 2013. Selective search for object recognition. *International Journal of Computer Vision*.

Yoshitaka Ushiku, Tatsuya Harada, and Yasuo Kuniyoshi. 2012. Efficient image annotation for automatic sentence generation. In *ACM International Conference on Multimedia ACM MM*.

H. Wang, A. Kläser, C. Schmid, and C. Liu. 2013. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103:60–79.

Jianxiong Xiao, J. Hays, K.A. Ehinger, A. Oliva, and A. Torralba. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *Conference on Computer Vision and Pattern Recognition CVPR*.

Yezhou Yang, Ching Lik Teo, Hal Daumé, III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Conference on Empirical Methods in Natural Language Processing EMNLP*.

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Li Fei-Fei. 2011. Action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision ICCV*.

Xiaodong Yu, Cornelia Fermuller, Ching Lik Teo, Yezhou Yang, and Yiannis Aloimonos. 2011. Active scene recognition with vision and language. In *Proceedings of the 2011 International Conference on Computer Vision*, International Conference on Computer Vision ICCV.

Mining Scientific Terms and their Definitions: A Study of the ACL Anthology

Yiping Jin[1] **Min-Yen Kan**[1,2] **Jun-Ping Ng**[1] **Xiangnan He**[1] *
[1]Department of Computer Science
[2]Interactive and Digital Media Institute
National University of Singapore
13 Computing Drive, Singapore 117417
{yiping, kanmy, junping, xiangnan}@comp.nus.edu.sg

Abstract

This paper presents DefMiner, a supervised sequence labeling system that identifies scientific terms and their accompanying definitions. DefMiner achieves 85% F_1 on a Wikipedia benchmark corpus, significantly improving the previous state-of-the-art by 8%.

We exploit DefMiner to process the ACL Anthology Reference Corpus (ARC) – a large, real-world digital library of scientific articles in computational linguistics. The resulting automatically-acquired glossary represents the terminology defined over several thousand individual research articles.

We highlight several interesting observations: more definitions are introduced for conference and workshop papers over the years and that multiword terms account for slightly less than half of all terms. Obtaining a list of popular defined terms in a corpus of computational linguistics papers, we find that concepts can often be categorized into one of three categories: resources, methodologies and evaluation metrics.

1 Introduction

Technical terminology forms a key backbone in scientific communication. By coining formalized terminology, scholars convey technical information precisely and compactly, augmenting the dissemination of scientific material. Collectively, scholarly compilation efforts result in reference sources such as printed dictionaries, ontologies and thesauri.

While online versions are now common in many fields, these are still largely compiled manually, relying on costly human editorial effort. This leads to resources that are often outdated or stale with respect to the current state-of-the-art. Another indirect result of this leads to a second problem: lexical resources tend to be general, and may contain multiple definitions for a single term. For example, the term "CRF" connotes "Conditional Random Fields" in most modern computational linguistics literature; however, there are many definitions for this acronym in Wikipedia. Because only one correct sense applies, readers may need to expend effort to identify the appropriate meaning of a term in context.

We address both issues in this work by automatically extracting terms and definitions directly from primary sources: scientific publications. Since most new technical terms are introduced in scientific publications, our extraction process addresses the bottleneck of staleness. Second, since science is organized into disciplines and sub-disciplines, we can exploit this inherent structure to gather contextual information about a term and its definition.

Aside from performance improvements, the key contributions of our work are in 1) recasting the problem as a sequence labeling task and exploring suitable learning architectures, 2) our proposal and validation of the the use of shallow parsing and dependency features to target definition extraction, and 3) analyzing the ACL Anthology Reference Corpus from statistical, chronological and lexical viewpoints.

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 780–790,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

The task of definition mining has attracted a fair amount of research interest. The output of such systems can be used to produce glossaries or answer definition questions. The primary model for this task in past work has been one of binary classification: does a sentence contain a definition or not? Existing methods can be cast into three main categories, namely rule-based (Muresan and Klavans, 2002; Westerhout and Monachesi, 2007), supervised machine learning (Fahmi and Bouma, 2006; Westerhout, 2009), and semi-supervised approaches (Navigli and Velardi, 2010; Reiplinger et al., 2012).

Rule-based approaches are intuitive and efficient, and were adopted in early research. Here, system performance is largely governed by the quality of the rules. Muresan and Klavans (2002) developed a rule-based system to extract definitions from online medical articles. The system first selects candidates using hand-crafted cue-phrases (e.g. *is defined as*, *is called*; analogous to "IS-A" phrases), further filtering the candidates with grammar analysis. Westerhout and Monachesi (2007) augmented the set of rules with part-of-speech (POS) tag patterns, achieving an F_2 of 0.43.

While such manually-crafted expert rules have high precision, they typically suffer from low recall. Definitions can be expressed in a variety of ways, making it difficult to develop an exhaustive set of rules to locate all definition sentences. To address low recall, later research adopted data-driven machine learning approaches. Fahmi and Bouma (2006) made used of supervised machine learning to extract definitions from a corpus of Dutch Wikipedia pages in the medical domain. They showed that a baseline approach which simply classifies every first sentence as a definition works surprisingly well, achieving an accuracy of 75.9% (undoubtedly due to the regular structure and style for Wikipedia). Their final system, based on the important feature of sentence position, was augmented with surface-level features (bag-of-words, bigrams, etc.) and syntactic features (type of determiner, position of the subject in the sentence). Their study with three different learners – naïve Bayes, maximum entropy (MaxEnt) and the support vector machine (SVM) – showed that MaxEnt gave the best results (92.2% accurate).

Westerhout (2009) worked on a hybrid approach, augmenting a machine learner to a set of handwritten rules. A random forest classifier is used to exploit linguistic and structural features. Informed by Fahmi and Bouma (2006)'s study, she included article and noun types in her feature set. Lexico-structural cues like the layout of the text are also exploited. She evaluated the performance of different cue phrases including the presence of "IS-A" phrases, other verbs, punctuations and pronouns. The highest F_2 score of 0.63 is reported for the "IS-A" pattern.

Since 2009 the focus of the research shifted to methods not limited to feature engineering. Borg et al. (2009) implemented a fully automated system to extract and rank definitions based on genetic algorithms and genetic programming. They defined two sub-problems including 1) acquiring the relative importance of linguistic forms, and 2) learning of new linguistic forms. Starting with a small set of ten simple hand-coded features, such as having sequence "FW IS" (FW is a tag for foreign word) or containing keyword identified by the system, the system is able to learn simple rules such as "NN is a NN". However, their system is optimized for similar "IS-A" patterns, as was used in (Westerhout, 2009). Their system, achieving an average f-measure of 0.25, also performs poorer than machine learning systems which exploit more specific features.

To cope with the generality of patterns, Navigli and Velardi (2010) proposed the three-step use of directed acyclic graphs, called Word-Class Lattices (WCLs), to classify a Wikipedia dataset of definitions. They first replace the uncommon words in the sentences with a wildcard (*), generating a set of "star patterns". Star patterns are then clustered according to their similarity. For each cluster, a WCL is generated by aligning the sentences in the cluster to form a generalized sentence. Although they reported a higher precision and recall compared with previous work, the result for WCL (F_1 of 75.23%) is not significantly better than the baseline system which exploits only star patterns (F_1 of 75.05%) without generating the directed graphs.

Reiplinger et al. (2012) took a semi-supervised approach. They employed bootstrapping to extract glossary sentences from scientific articles in

the ACL Anthology Reference Corpus (ACL ARC) (Bird et al., 2008). Their results show that bootstrapping is useful for definition extraction. Starting from a few initial term-definition pairs and hand-written patterns as seeds, their system iteratively acquires new term-definition pairs and new patterns.

We note that these previous systems rely heavily on lexico-syntactic patterns. They neither sufficiently exploit the intrinsic characteristics of the term and definition, nor invest effort to localize them within the sentence[1]. Given the significant structure in definitions, we take a more fine-grained approach, isolating the term and its definition from sentences. According to Pearson (1996), a definition can be formally expressed as:

$$X = Y + \text{distinguishing characteristics},$$

where "X" is the *definiendum* (defined term; hereafter, *term*), and "Y + distinguishing characteristics" can be understood as the *definiens* (the term's definition; hereafter, *definition*). The connector "=" can be replaced by a verb such as "define", "call", a punctuation mark or other phrase. Our task is thus to find pairs of terms and their associated definitions in input scholarly articles. The sentence-level task of deciding whether a sentence s is a definition sentence or not, is thus a simplification of our task.

3 Methodology

Our DefMiner system is based on the sequence labeling paradigm – it directly assigns an annotation a_i from $A \in \{$(T)*erm*,(D)*efinition*,(O)*ther*$\}$ for each input word w_i. We post-process our labeler's results to achieve parity with the simplified definition sentence task: When we detect both a term's and definition's presence in a sentence, we deem the sentence a definition sentence. To be clear, this is a requirement; when we detect only either a term or a definition, we filter these out as false positives and do not include them as system output – by definition in DefMiner, terms must appear within the same sentence as their definitions.

To train our classifier, we need a corpus of definition sentences where all terms and definitions are annotated. While Navigli and Velardi (2010) compiled the WCL definition corpus from the English Wikipedia pages, we note that Wikipedia has stylistic conventions that make detection of definitions much easier than in the general case (i.e., "The first paragraph defines the topic with a neutral point of view"[2]). This makes it unsuitable for training a general extraction system from scholarly text.

As such, we choose to construct our own dataset from articles collected from the ACL ARC, following (Reiplinger et al., 2012). We compiled a corpus – the W00 corpus, named for its prefix in the ACL Anthology – derived from a total of 234 workshop papers published in 2000. Due to limited resources and time, only one individual (the first author) performed the corpus annotation. We built three disjoint prototype classifiers to further filter the sentences. The prototype classifiers are based on surface patterns, keyphrases and linguistic phenomena (# of NP, # of VP, etc.). We took all the 2,512 sentences marked as definition sentences by at least one of our individual prototypes and proceeded to annotate all of them. In total, 865 of the total 2,512 sentences were real definition sentences.

The annotation instance is a single token (including single word or punctuation mark). Each token w_i was marked with a_i from $A \in \{$T,D,O$\}$ depending on whether it is part of a term, a definition or neither (other). Therefore, a sentence that is not a definition sentence would have all its tokens marked as O. The corpus and its annotations are available for comparative study[3].

We use Conditional Random Fields (CRFs) (Lafferty et al., 2001) to extract the term and definition from input. We incorporate evidence (features) drawn from several levels of granularity: from the word-, sentence- and document-levels, not limiting ourselves to the window of previous n words. CRFs allow us to encode such features which may not be conditionally independent. We use the open source implementation, CRF++ (Kudo, 2005) in our work[4].

One straightforward approach is to train independent classifiers for terms and definitions, which we

[1] While Navigli and Velardi (2010) tagged *terms* and *definitions* explicitly in their corpus, their evaluation restricts itself to the task of definition sentence identification.

[2] http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section

[3] http://wing.comp.nus.edu.sg/downloads/term_definition_mining/

[4] http://code.google.com/p/crfpp/

test in Section 4.1. While simple, this is suboptimal as it ignores the correlation between the presence of the two components. Term identification (in the guise of key-word/-phrase extraction) is well studied and common features such as $tf.idf$ and English orthography achieve satisfactory results (Nguyen and Kan, 2007). In contrast, definitions exhibit more flexible structure and hence are more difficult to distinguish from normal English text.

As such, we further investigate a serial architecture where we perform the classifications in sequence. *I.e.*, first utilizing the results from *term* classification, and then incorporating them into *definition* classification. This two-stage architecture is explored later in Section 4.2

Our expanded feature set is an amalgamation of related works and utilizes a mix of simple lexical, orthography, dictionary lookup and corpus features (here, idf). Note that each feature class may derive more than one feature (e.g., for the POS tag feature, we extract features from not only the current word but the surrounding contextual words as well). We now enumerate the feature classes (FCs) that we exploit, marking whether they apply to the (W)ord, (S)entence or (D)ocument levels:

FC1) Lexical (W): The word, POS tag, stemmed word, and if the word contains a signal suffix[5].
FC2) Orthography (W): Whether the word is 1) capitalized or 2) mixed case; whether the word contains 3) hyphens or 4) digits.
FC3) Keyphrase List (W): Whether the word is in the keyphrase list of the origin document. We use the open source KEA keyphrase extraction system (Witten et al., 1999) to extract 20 keyphrases for each document.
FC4) Corpus (W): Discretized Inverse Document Frequency (idf), calculated as $log(\frac{N}{c})$, where N is the total number of documents and c is the number of occurrences of the word in all the documents. IDF values are discretize into eight uniform partitions.
FC5) Position (D,S): The 1) Section ID, 2) name and 3) the sentence's relative position in the document.
FC6) Has acronym (S): Whether the sentence contains an acronym. We use Stanford dependency parser (Cer et al., 2010) to parse the sentences. We deem the sentence to contain an acronym if the dependency type "abbrev" is present in the output of the parser.
FC7) Surface pattern (S): Whether the sentence contain one of the hand-crafted pattern, as listed in Table 1. The list is compiled from previous works and augmented based on our observations on the corpus.

[5]The suffixes we extract are "-ion","-ity","-tor","-ics","-ment","-ive" and "-ic".

<term > defined (as|by) <definition>
define(s)? <term> as <definition>
definition of <term> <definition>
<term> a measure of <definition>
<term> is DT <definition> (that|which|where)
<term> comprise(s)? <definition>
<term> consist(s)? of <definition>
<term> denote(s)? <definition>
<term> designate(s)? <definition>
<definition> (is|are|also) called <term>
<definition> (is|are|also) known as <term>

Table 1: Hand-crafted surface patterns used in DefMiner.

Long Distance Features. During development, we noticed that the syntactic variation of the definition might benefit from features that identify long-distance dependencies. As such, we further studied the impact of including additional features developed from the shallow (chunk) and dependency parses of the input. Compared to the above features, these features are much more computationally-intensive.

FC8) Shallow tag (W): Shallow parsing tag for each word (e.g., np, vp). We used OpenNLP toolkit to shallow parse the sentences. (Baldridge, 2005)
FC9) Shallow pattern (S): If the shallow parsing sequence contains one of the seven parse patterns listed in Table 2. We also give some example sentences which can be detected by the patterns.
FC10) Governor (W): The word that the current word depends on in a binary dependency relation. (e.g., for the phrase $computational\ linguistics$, the governor of the word $computational$ is $linguistics$).
FC11) Dependency path distance (W): Distance from the current word to the root of the sentence in the dependency tree.
FC12) Typed dependency path (W): The dependency path from the current word to the root of the sentence (recording the dependency types instead of the words in the path).

Pattern	Example
NP : NP	JavaRAP : An open-source implementation of the RAP
NP is * NP	IR is the activity of obtaining information resources
NP is * NP that/of/which	NLP is a subject which is well studied
NP or NP	Conditional Random field or CRF tackles ...
known as NP	The corpus of English Wikipedia pages, known as EnWiki
NP (* NP)	Hidden Markov Model (HMM) is used to solve ...
NP defined by/as * NP	The accuracy is defined by the production of ...

Table 2: Hand-crafted shallow parsing patterns used in DefMiner.

4 Evaluation

We now assess the overall effectiveness of DefMiner, at both the word and sentence level. Additionally, we want to ascertain the performance changes as we add features to an informed lexical baseline. We not only benchmark DefMiner's performance over our own W00 collection, but also compare DefMiner against previous published work on the definition sentence identification task on the WCL (English Wikipedia) corpus.

4.1 Single-Pass Extraction from W00

We run ten-fold cross validation on our annotated W00 corpus. We first evaluate our results at word level, calculating the precision, recall, and F_1 scores for each incrementally enhanced feature set. We present results on the corpus in the top portion of Table 3 (Rows 1–9).

We calculate both micro and macro- (category) averaged F_1 scores for term and definition extraction. F_{micro} assigns equal weight to every token, while F_{macro} gives equal weight to every category. As definition tokens greatly outnumber term tokens in our corpus (roughly 6:1), we feel that the macro-average is a better indicator of the balance between term and definition identification.

Our baseline system makes use of basic word and POS tag sequences as features (FC1), which are common to baselines in other sequence labeling works. We can see that most features result in performance improvements to the baseline, especially for recall. Interestingly, although the shallow parsing and dependency features we use are rather simple, they effectively improve the performance of the system. In System 7, we only use the seven shallow parsing patterns shown in Table 2, but the F_{macro} measure improves 3%. Our best single-stage system (System 9 in Table 3) boosts recall for term and definition classification by 7% and 5%, respectively, without sacrificing precision. The F_{macro} measure is improved from 0.44 to 0.48.

Unexpectedly, the inclusion of the position features cause performance to drop. One possible reason is that the authors of scientific papers have more flexibility to choose the positions to present definitions. This makes the position feature much less indicative (compared to running on a corpus of Wikipedia articles). Due to this observation, we exclude the position features when carrying out following experiments.

4.2 Serial Term and Definition Classification

We now investigate the two-stage, serial architecture where the system first performs term classification before definition classification (i.e., term$\succ$definition). We provision the second-stage definition classifier with three additional features from the first-stage term classification output: whether the current word (1) is a term, and (2) appears before or (3) after a term.

Row 10 shows this resulting system, which we coin as DefMiner. Interestingly, there is a 10% increase in the precision of definition classification. With the two-stage classifier, F_{macro} score further increases from 0.48 to 0.51. The results verify our intuition that term classification does help in definition classification. Pipelining in the opposite direction (definition$\succ$term; Row 11) does not show any improvement. We posit that since the advantage is only in a single direction, joint inference may be less likely to yield benefits.

To determine the upper bound performance that could result from proper term identification, we provided correct, oracular term labels from our ground truth annotations in our corpus to the second-stage definition classifier. This scenario effectively upper-bounds the performance that perfect term knowledge has on definition classification. The results of this system in Row 12 indicates a strong positive influence on definition extraction, improving definition extraction from 49% to 80%, a leap of 31%. This

System / Feature Class (*cf* Section 3)	Term			Definition			Overall	
	P	R	F_1	P	R	F_1	F_{micro}	F_{macro}
1: Baseline (FC1)	0.49	0.34	0.40	0.41	0.49	0.45	0.45	0.44
2: (1) + Orthography (FC2)	0.46	0.35	0.40	0.42	0.51	0.46	0.46	0.44
3: (2) + Dictionary (FC3)	0.48	0.36	0.41	0.41	0.49	0.44	0.44	0.43
4: (3) + Corpus (FC4)	0.50	0.35	0.41	0.40	0.52	0.45	0.45	0.44
5: (4) + Position (FC5)	0.47	0.37	0.42	0.36	0.48	0.41	0.41	0.41
6: (4) + Shallow parsing tag (FC8)	**0.51**	0.38	0.43	0.41	0.50	0.45	0.45	0.44
7: (6) + Shallow parse pattern (FC9)	0.50	0.40	**0.45**	0.42	0.52	0.47	0.47	0.47
8: (7) + Surface pattern (FC7)	0.49	0.39	0.44	0.43	0.53	0.48	0.48	0.47
9: (8) + Dependency + acronym (FC6,10,11,12)	0.50	**0.41**	**0.45**	0.45	0.54	0.49	0.49	0.48
10 [DefMiner]: (9) + 2-stage	0.50	**0.41**	**0.45**	**0.55**	**0.58**	**0.56**	**0.55**	**0.51**
11: (9) + Reverse 2-stage	0.50	0.40	0.44	0.45	0.54	0.49	0.49	0.48
12: (9) + Term Oracle	*N/A*	*N/A*	*N/A*	0.79	0.82	0.80	*N/A*	*N/A*

Table 3: 10-fold cross validation word-level performance over different system configurations on our W00 corpus.

motivates future work as how to improve the performance of the term classifier so as to reap the benefits possible with our two-stage classifier.

4.3 Comparative results over the WCL Dataset

For most of the related research reviewed, we could neither obtain their source code nor the corpora used in their work, making comparative evaluation difficult. To the best of our knowledge, Reiplinger et al. (2012) is the only attempt to extract definitions from the ACL ARC corpus, which is a superset of our W00 corpus. It would be desirable to have a direct comparison with their work, but their evaluation method is mainly based on human judges and their reported coverage of 90% is only for a sample, short list of domain terms they defined in advance.

To directly compare with the previous, more complex state-of-the-art system from (Navigli and Velardi, 2010), we evaluate DefMiner on the definition sentence detection task. For the sentence-level evaluation, we calculate the $P/R/F_1$ score based on whether the sentence is a definition sentence. We applied DefMiner on their whole WCL annotated corpus, reporting results in Table 4. We randomized the definition and none-definition sentences in their corpus and applied 10-folds cross validation. In each each iteration we used 90% of the sentences for training and 10% for testing.

Compared to their results reported in (Navigli and Velardi, 2010), DefMiner improves overall F_1 by 8%. While certainly less precise (precision of 92% versus 99%), recall is improved over their considerably more complex WCL-3 algorithm by almost 20%. Even using just the simple heuristic of only classifying sentences that have identified terms as well as definitions as definition sentences, DefMiner serves to competitively identify definition sentences.

System	Token Level		Sentence Level
	Term	Definition	
	P / R / F_1	P / R / F_1	P / R / F_1
DefMiner	.82/.78/.80	.82/.79/.81	.92/.79/.85
N&V '10	– / – / –	– / – / –	.99/.61/.77

Table 4: Comparative performance over the WCL.

4.4 Manual Inspection of DefMiner Output

To gauge the noise from our system outside of our cross-validation experiments, we conducted a manual inspection of results over other workshop papers from other years (2001 and 2002), as a sanity check. DefMiner identifies 703 and 1,217 sentences in W01 and W02 as definition sentences separately.

Overall, 77.8% of the extracted sentences are real definition sentences, while the remaining are false positives.

4.4.1 Analysis of Common Errors

The $P/R/F_1$ score by itself only gives a hint of the system's overall performance. We are also interested to study the common errors made by our system, which could help us engineer better features for improving DefMiner. As our two-stage classifier still lags behind the system with oracular term labels by 24% in F_1 for definition detection (Section 4.2), we believe there is still much room for improvement. We show three example misclassified sentences that represent the major types of errors we observed, where DefMiner's output annotations follow tokens marked as part of terms or definitions.

1) *A PSS/TERM thus contains abstract linguistic values for closed features (tense/DEF ,/DEF mood/DEF ,/DEF voice/DEF ,/DEF number/DEF ,/DEF gender/DEF ,/DEF etc/DEF ./DEF) .*

This first instance shows that DefMiner tends to mark the first several tokens as "TERM" while the real term appears somewhere else in the sentence. The actual term being defined is "closed features" instead of "PSS". Many terms in the training set appear at the beginning of the sentence and are preceded by a determinant. "PSS" is also likely to receive a high IDF and orthographic shape (capitalized) score and therefore are misclassified as terms. It may be useful to thus model the (usual) distance between the term and its definition in a feature in future work.

2) *Similarly , 'I'/TERM refers to an/DEF interior/DEF character/DEF and/DEF 'L'/DEF indicates/DEF the/DEF last/DEF character/DEF of/DEF a/DEF word/DEF .*

DefMiner is occasionally confused when encountering recursive definition or multiple definitions in a single sentence. Sentence 2 contains two parallel definitions. The classifier fails to classify "L" as a separate term, incorporating it as part of the definition. One possible improvement is to break the original sentence into clauses that are independent from each other, perhaps by using even simple surface cues such as coordinating conjunctions marked by commas or "and".

3) *Again one could argue that the ability to convey such uncertainty and reliability information to a non-specialist/TERM is a/DEF key/DEF advantage/DEF of/DEF textual/DEF summaries/DEF over/DEF graphs/DEF .*

Another difficult problem faced by the classifier is the lack of contextual information. In sentence 3), if we just look at part of the sentence "a non-specialist is a key advantage of textual summaries over graphs", without trying to understand the meaning of the sentence, we may well conclude that it is a definition sentence because of the cue phrase "is a". But clearly, the whole sentence is not a definition sentence. More sophisticated features based on the sentence parse tree have to be exploited to detect such false positive examples.

5 Insights from the Definitions Extracted from the ACL ARC

In this second half of the paper, we apply DefMiner to gain insights on the distributional and lexical properties of terms and definitions that appear in the large corpus of computational linguistics publications represented by the ACL ARC (Bird et al., 2008). The ARC consists of 10,921 scholarly publications from ACL venues, of which our earlier W00 corpus is a subset (*n.b.*, as such, there is a small amount of overlap). We trained a model using the whole of the W00 corpus and used the obtained classifier to identify a list of terms and definitions for each publication in the ACL ARC.

Inspecting such output gives us an understanding of the properties of definition components, eventually helping the community to define better features to capture them, as well as intrinsically deepening our knowledge of the natural language of definitions and the structure of scientific discourse.

5.1 Demographics

From a term's perspective we can introspect properties of the enclosing paper, the host sentence, the term itself and its definition.

At the *document* level, we can analyze document metadata: its venue (journal, conference or workshop published) and year of publication.

At the *sentence* level, we analyze the position of the sentences that are definition sentences.

Focusing on *terms*, we want to find out in more detail the technical terminology that is defined. Are they different from general keyphrases? What type of entities are defined? What words do these terms consist of? What structures are common?

We are interested in analogous questions when focusing on the accompanying *definitions*. How many words or clauses do definition sentences consist of? Do we lose a lot of recall by restricting definitions to a single sentence? Are embedded definitions (definitions embedded with other definitions) common?

We highlight some specific findings from our basic analyses here:

Where do definitions occur? As terms are usually defined on first use, we expect the distribution of definition sentences to skew towards the beginning portions of a scientific document as input. We count the occurrences of definition sentences in each of ten equally-sized (by number of sentences) non-overlapping partitions. The results are shown in Figure 1, aligning with our intuition: The first three quantiles contribute almost 40% of all detected definition sentences, while the last three quantiles contain only 17.8%.

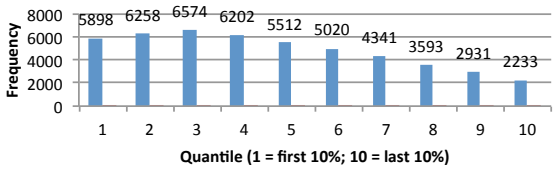
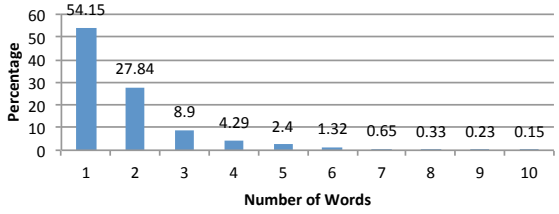
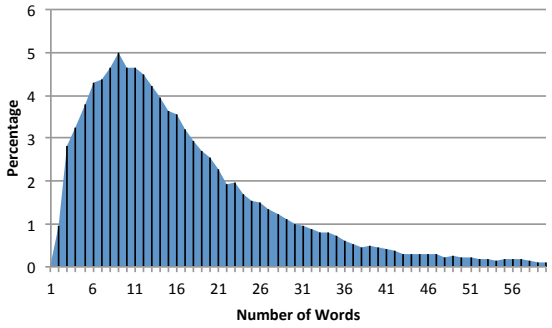


Figure 1: Occurrences of definitions within different segments of an article.

How long are the detected terms and definitions? Figure 2 shows the detected aggregate distributions. Over 54% of the detected terms are single tokens, where majority of the remaining 45% of terms being multi-word terms of six words or less. Among the single token terms, a further analysis reveals that 17.4% are detected as single-character variables, 34.9% are acronyms (consisting of more than one capitalized letters), while the remaining 47.7% are normal words. Definitions, in contrast, are longer and more varied, with a peak length of nine words. Slightly over half of the definitions have a length of 5–16 words; 75% have lengths between 3 and 23 words.



(t) Term length distribution.



(b) Definition length distribution.

Figure 2: Length distributions of (top) terms, (bottom) definitions.

In Table 5, we present the 10 most frequent POS tag bigrams for terms and definitions. We can see that among terms, a sequence of consecutive two nouns is most common, making up four out of the top five bigrams. We notice that determiners and prepositions are absent from the term list but are common in definitions.

5.2 Inspection of Definition over Time

The ACL ARC covers journal articles, conference and workshop proceedings over a few decades. As with other fields in recent years, the amount of computational linguistics literature has steadily increased over the number of years.

We study if definitions appear as frequently in different types of scientific articles (e.g. journals, conference papers or workshop papers). We also want to investigate if there is a significant shift in the distribution of definitions across years. In Figure 3, we present the density of definitions (defined as the

Term	Definition
NNP NNP	DT NN
NN NN	NNP NNP
NNP NN	NN IN
NN NNP	IN DT
JJ NN	NN NN
NN JJ	JJ NN
NNP :	NN :
: NNP	DT JJ
NN NNS	NNS IN
NN ”	NN NNS

Table 5: Most frequent POS bigrams for terms and definitions.

percentage of sentences that are identified as definition sentences), for these three different categories of publications[6].

In Figure 3, the three data series overlap each other, so we cannot conclude definitions appear more often in one type of papers than another. However, as a side effect, we see that while the definition density for journal papers remain relatively constant, for conference and workshop papers the number of definitions extracted per sentence has increased noticeably over time. The average number of definitions presented in conference papers, for instance, increased more than 100% in the 40 years represented in the ACL ARC.

The increasing number of definitions alone does not show that new knowledge is introduced at a faster rate, as definitions may be repeated. To control for this effect, we also need to know which definitions are new or defined in previous year(s). We studied this effect in more detail for the relatively smaller set of journal papers (Figure 4). For journal papers, the number of definitions of previously introduced terms in each year against the number of new definitions. We say a definition is new when the detected term was not identified in any article (not limited to journals) in previous years.

We see that the number of new terms being defined also increases with the years. But the increase is much slower than that for the total definitions. The area between the two lines denotes the definitions where the same term has been multiply defined from the same or previous years as the current year under investigation.

[6]The ACL ARC is organized by the venue of the publication, which is associated to a category. For the assigned category for each venue please refer to Appendix A.

5.3 Trends

We can use terms and definitions to also introspect how the computational linguistics literature has changed over time. Table 6 shows a subset of the most frequently defined terms in the ACL ARC, where we exclude single-character terms (“variables”).

WordNet (292)	Part Of Speech (45)
Precision (172)	Probablistic CFG (43)
Recall (167)	FrameNet (38)
Noun phrase (97)	Conditional Random Field (29)
Word sense disambiguation (60)	Inverse document frequency (28)
Support Vector Machine (60)	PropBank (27)
Hidden Markov Model (54)	Context Free Grammar (25)
Latent Semantic Analysis (57)	Accuracy (20)

Table 6: Subset of most frequently defined terms. Raw counts in parentheses. Variations of the same term (*e.g.* plurals, acronyms) are collapsed into one instance.

To be expected, these popular terms are mostly specific to computational linguistics. From our observation, we can fit these terms into one of three categories, including 1) resources (WordNet, FrameNet, PropBank), 2) methodologies (SVM, HMM, LSA), and 3) evaluation metrics (Precision, Recall, Accuracy). We feel that the final category of evaluation metrics is more general and would be shared among other scientific disciplines.

An interesting analysis that follows from this categorization is that we can study major trends and changes in the research directions of the community. This can help to draw the attention of researchers to emerging trends. We illustrate this approach in Figure 5 that focuses on three sequence labeling methodologies that have been used to address similar problems – namely, hidden Markov model (HMM), maximum entropy Markov model (MEMM), and conditional random fields (CRF) – during the period from 1989 to 2006 (where we have sufficient data points). From the early 90s, we see that HMM was a clear favorite. However since 2000, MEMM gained in popularity and use. Lafferty et al. (2001) introduced CRFs in 2001 and the new methodology was widely adopted soon after that.

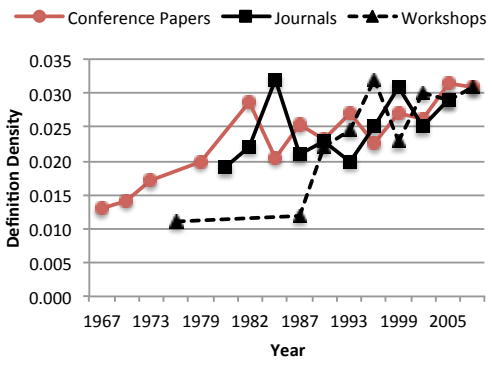


Figure 3: Occurrences of definitions across publication categories.

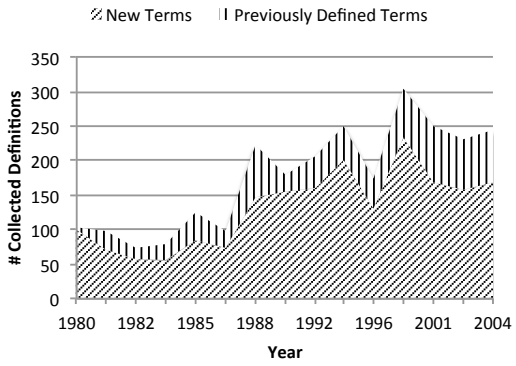


Figure 4: Relative proportions of new and recurring definitions in journal papers.

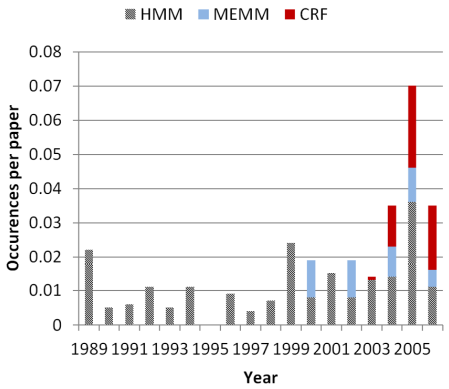


Figure 5: The occurrence of definitions for various sequence labeling methodologies over the years.

6 Conclusions and Future Work

We study the task of identifying definition sentences, as a two-part entity containing a term and its accompanying definition. Unlike previous work, we propose the harder task of delimiting the component term and definitions, which admits sequence labeling methodologies as a compatible solution.

Leveraging the current best practice of using conditional random fields, we contribute two additional ideas that lead to DefMiner, a state-of-the-art scholarly definition mining system. First, we show that shallow parsing and dependency parse features that may provide additional non-local information, are useful in improving task performance. Second, viewing the problem as two correlated subproblems of term and definition extraction, we measure the tightness and dependency of the correlation. We find that a two-stage sequential learning architecture (term first, definition second) leads to best performance. DefMiner outperforms the state of the art, and we feel is fit for macroscopic analysis of scientific corpora, despite significant noise.

We thus deployed DefMiner on the *ACL Anthology Reference Corpus*. We demonstrate how DefMiner can yield insights into both the structure and semantics of terms and definitions, in both static and diachronic modes. We think future work could pursue more in-depth analysis of the distributional and demographic properties of automatically extracted lexica. We can use the lexicon obtained from different years to carry out trend prediction, which we have illustrated here. Downstream systems may predict which term will become popular, or could alert an author if their definition of a term significantly differs from the original source.

We hope to tackle the annotation bottleneck in future work on definition extraction, common in many data-driven learning fields. We plan to explore iterative, semi-supervised methods to best manage human effort to maximize the effectiveness of future annotation.

In addition, with respect to modeling, although we showed that doing definition classification before term classification does not improve over our single-stage classifier, we hope to study whether suitable joint inference models can benefit from the interaction between the two classification processes.

References

Jason Baldridge. 2005. The opennlp project.

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan Gibson, Mark T. Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the 6th International Conference on Language Resources and Evaluation Conference (LREC08)*, pages 1755–1759, Marrakech, Morocco.

Claudia Borg, Mike Rosner, and Gordon Pace. 2009. Evolutionary algorithms for definition extraction. In *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09, pages 26–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *7th International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.

Ismail Fahmi and Gosse Bouma. 2006. Learning to identify definitions using syntactic features. In *Proceedings of the EACL workshop on Learning Structured Information in Natural Language Applications*, Trento, Italy.

Taku Kudo. 2005. Crf++: Yet another crf toolkit. *Software available at http://crfpp. sourceforge. net*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA.

Smaranda Muresan and Judith Klavans. 2002. A method for automatically building and evaluating dictionary resources. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Las Palmas, Canary Islands, Spain.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327, Uppsala, Sweden.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of International Conference on Asian Digital Libraries*, pages 317–326, Hanoi, Vietnam.

Jennifer Pearson. 1996. The expression of definitions in specialised texts: a corpus-based analysis. In *Proceedings of Euralex 96*.

Melanie Reiplinger, Ulrich Schafer, and Magdalena Wolska. 2012. Extracting glossary sentences from scholarly articles: a comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, ACL '12, pages 55–65, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eline Westerhout and Paola Monachesi. 2007. Extraction of dutch definitory contexts for elearning purposes. In *Proceedings of Computational Linguistics (CLIN 2007)*.

Eline Westerhout. 2009. Definition extraction using linguistic and structural features. In *Proceedings of the 1st Workshop on Definition Extraction*, WDE '09, pages 61–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea : Practical automatic keyphrase extraction. *Computer*, pp:254–255.

Joint Learning and Inference for Grammatical Error Correction

Alla Rozovskaya and Dan Roth
Cognitive Computation Group
University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue
Urbana, IL 61801
{rozovska,danr}@illinois.edu

Abstract

State-of-the-art systems for grammatical error correction are based on a collection of independently-trained models for specific errors. Such models ignore linguistic interactions at the sentence level and thus do poorly on mistakes that involve grammatical dependencies among several words. In this paper, we identify linguistic structures with interacting grammatical properties and propose to address such dependencies via joint inference and joint learning.

We show that it is possible to identify interactions well enough to facilitate a joint approach and, consequently, that joint methods correct incoherent predictions that independently-trained classifiers tend to produce. Furthermore, because the joint learning model considers interacting phenomena during training, it is able to identify mistakes that require making multiple changes simultaneously and that standard approaches miss. Overall, our model significantly outperforms the Illinois system that placed first in the CoNLL-2013 shared task on grammatical error correction.

1 Introduction

There has recently been a lot of work addressing errors made by English as a Second Language (ESL) learners. In the past two years, three competitions devoted to grammatical error correction for non-native writers took place: HOO-2011 (Dale and Kilgarriff, 2011), HOO-2012 (Dale et al., 2012), and the CoNLL-2013 shared task (Ng et al., 2013).

Nowadays **phone/phones* **has/have* many functionalities, **included/including* **∅/a* camera and **∅/a Wi-Fi receiver.

Figure 1: Examples of representative ESL errors.

Most of the work in the area of ESL error correction has addressed the task by building statistical models that specialize in correcting a specific type of a mistake. Figure 1 illustrates several types of errors common among non-native speakers of English: article, subject-verb agreement, noun number, and verb form. A significant proportion of research has focused on correcting mistakes in article and preposition usage (Izumi et al., 2003; Han et al., 2006; Felice and Pulman, 2008; Gamon et al., 2008; Tetreault and Chodorow, 2008; Gamon, 2010; Rozovskaya and Roth, 2010b). Several studies also consider verb-related and noun-related errors (Lee and Seneff, 2008; Gamon et al., 2008; Dahlmeier and Ng, 2012). The predictions made by individual models are then applied independently (Rozovskaya et al., 2011) or pipelined (Dahlmeier and Ng, 2012).

The standard approach of training individual classifiers considers each word independently and thus assumes that there are no interactions between errors and between grammatical phenomena. But an ESL writer may make multiple mistakes in a single sentence and these result in misleading local cues given to individual classifiers. In the example shown in Figure 1, the agreement error on the verb "have" interacts with the noun number error: a correction system that takes into account the context may infer, because of the word "phone", that the verb number is correct. For this reason, a system that consid-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 791–802,
Seattle, Washington, USA, 18-21 October 2013.

ers noun and agreement errors separately will fail to identify and correct the interacting errors shown in Fig. 1. Furthermore, it may also produce inconsistent predictions.

Even though it is quite clear that grammatical errors interact, for various conceptual and technical reasons, this issue has not been addressed in a significant way in the literature. We believe that the reasons for that are three-fold: (1) Data: until very recently we did not have data that jointly annotates sufficiently many errors of interacting phenomena (see Sec. 2). (2) Conceptual: Correcting errors in interacting linguistic phenomena requires that one identifies those phenomena and, more importantly, can recognize reliably the interacting components (e.g., given a verb, identify the subject to enable enforcing agreement). The perception has been that this cannot be done reliably (Sec. 4). (3) Technical: The NLP community has started to better understand joint learning and inference and apply it to various phenomena (Roth and Yih, 2004; Punyakanok et al., 2008; Martins et al., 2011; Clarke and Lapata, 2007; Sutton and McCallum, 2007) (Sec. 5).

In this paper we present, for the first time, a successful approach to jointly resolving grammatical errors. Specifically:

• We identify two pairs of interacting phenomena, *subject-verb* and *article-NPhead* agreements; we show how to reliably identify these pairs in noisy ESL data, thereby facilitating the joint correction of these phenomena.

• We propose two joint approaches: (1) a *joint inference* approach implemented on top of individually learned models using an integer linear programming formulation (ILP, (Roth and Yih, 2004)), and (2) a model that *jointly learns* each pair of these phenomena. We show that each of these methods has its advantages, and that both solve the two challenges outlined above: the joint models exclude inconsistent predictions that violate linguistic constraints. The joint learning model exhibits superior performance, as it is also able to overcome the problem of the noisy context encountered by the individual models and to identify errors in contexts, where multiple changes need to be applied at the same time.

We show that our joint models produce state-of-the-art performance and, in particular, significantly outperform the University of Illinois system that placed first in the CoNLL-2013 shared task, increasing the F1 score by 2 and 4 points in different evaluation settings.

2 Task Description and Motivation

To illustrate the utility of jointly addressing interacting grammatical phenomena, we consider the corpus of the CoNLL-2013 shared task on grammatical error correction (Ng et al., 2013), which we found to be particularly well-suited for addressing interactions between grammatical phenomena. The task focuses on the following five common mistakes made by ESL writers: *article*, *preposition*, *noun number*, *subject-verb agreement*, and *verb form*, and we address two interactions: *article-NPhead* and *subject-verb*.

The training data for the task is from the NUCLE corpus (Dahlmeier et al., 2013), an error-tagged collection of essays written by non-native learners of English. The test data is an additional set of essays by learners from the same linguistic background. The training and the test data contain 1.2M and 29K words, respectively. Although the corpus contains errors of other types, the task focuses on five types of errors. Table 1 shows the number of mistakes[1] of each type and the error rates, i.e. the percentage of erroneous words by error type.

Error	**Number of errors and error rate**	
	Training	**Test**
Article	6658 (2.4%)	690 (10.0%)
Prep.	2404 (2.0%)	311 (10.7%)
Noun	3779 (1.6%)	396 (6.0%)
Verb Agr.	1527(2.0%)	124 (5.2%)
Verb Form	1453 (0.8%)	122 (2.5%)

Table 1: **Number of annotated errors in the CoNLL-2013 shared task.** Percentage denotes the error rates, i.e. the number of erroneous instances with respect to the total number of relevant instances in the data. For example, 10.7% of prepositions in the test data are used incorrectly. The numbers in the *revised* data set are slightly higher.

We note that the CoNLL-2013 data set is the first annotated collection that makes a study like ours feasible. The presence of a common test set that

[1]System performance in the shared task is evaluated on data with and without additional revisions added based on the input from participants. The number of mistakes in the revised test data is slightly higher.

contains a good number of interacting errors – article, noun, and verb agreement mistakes – makes the data set well-suited for studying which approach works best for addressing interacting phenomena. The HOO-2011 shared task collection (Dale and Kilgarriff, 2011) contains a very small number of noun and agreement errors (41 and 11 in test, respectively), while the HOO-2012 competition (Dale et al., 2012) only addresses article and preposition mistakes. Indeed, in parallel to the work presented here, Wu and Ng (2013) attempted the ILP-based approach of Roth and Yih (2004) in this domain. They were not able to show any improvement, for two reasons. First, the HOO-2011 data set which they used does not contain a good number of errors in interacting structures. Second, and most importantly, they applied constraints in an indiscriminate manner. In contrast, we show how to identify the interacting structures' components in a reliable way, and this plays a key role in the joint modeling improvements.

Lack of data hindered other earlier efforts for error correction beyond individual language phenomena. Brockett et al. (2006) applied machine-translation techniques to correct noun number errors on mass nouns and article usage but their application was restricted to a small set of constructions. Park and Levy (2011) proposed a language-modeling approach to whole sentence error correction but their model is not competitive with individually trained models. Finally, Dahlmeier and Ng (2012) proposed a decoder model, focusing on four types of errors in the data set of the HOO-2011 competition (Dale and Kilgarriff, 2011). The decoder optimized the sequence in which individual classifiers were to be applied to the sentence. However, because the decoder still corrected mistakes in a pipeline fashion, one at a time, it is unlikely that it could deal with cases that require simultaneous changes.

3 The University of Illinois System

Below, we briefly describe the University of Illinois system (henceforth *Illinois*; in the overview paper of the shared task the system is referred to as UI) that achieved the best result in the CoNLL-2013 shared task and which we use as our baseline model. For a complete description, we refer the reader to Rozovskaya et al. (2013).

The Illinois system implements five machine-learning independently-trained classifiers that follow the popular approach to ESL error correction borrowed from the context-sensitive spelling correction task (Golding and Roth, 1999; Carlson et al., 2001). A *confusion set* is defined that specifies a list of confusable words. Each occurrence of a confusable word in text is represented as a vector of features derived from a context window around the target. The problem is cast as a multi-class classification task and a classifier is trained on native or learner data. At prediction time, the model selects the most likely candidate from the confusion set.

The confusion set for prepositions includes the top 12 most frequent English prepositions. The article confusion set is as follows: {a,the,∅}[2]. The confusion sets for *noun*, *agreement*, and *form* modules depend on the target word and include its morphological variants (Table 2).

"Hence, the environmental ***factor/factors** also ***contributes/contribute** to various difficulties, ***included/including** problems in nuclear technology."	
Error type	**Confusion set**
Noun	{factor, factors}
Verb Agr.	{contribute, contributes}
Verb Form	{included, including, includes, include}

Table 2: **Confusion sets for noun number, agreement, and form classifiers.**

The *article* classifier is a discriminative model that draws on the state-of-the-art approach described in Rozovskaya et al. (2012). The model makes use of the Averaged Perceptron algorithm (Freund and Schapire, 1996) and is trained on the training data of the shared task with rich features.

The other models are trained on native English data, the Google Web 1T 5-gram corpus (henceforth, Google, (Brants and Franz, 2006)) with the Naïve Bayes (NB) algorithm. All models use word n-gram features derived from the 4-word window around the target word. In the *preposition* model, priors for preposition preferences are learned from the shared task training data (Rozovskaya and Roth, 2011).

[2] ∅ denotes noun-phrase-initial contexts where an article is likely to have been omitted. The variants "a" and "an" are conflated and are restored later.

Example	Predictions made by the Illinois system
"They believe that *such situation* must be avoided."	such situation → such a situations
"Nevertheless , electric *cars is* still regarded as a great trial innovation."	cars is → car are
"Every *students have* appointments with the head of the department."	No change

Table 3: **Examples of predictions of the Illinois system that combines independently-trained models.**

The words that are selected as input to classifiers are called *candidates*. Article and preposition candidates are identified with a closed list of words; noun-phrase-initial contexts for the article classifier are determined using a shallow parser[3] (Punyakanok and Roth, 2001). Candidates for the noun, agreement, and form classifiers are identified with a part-of-speech tagger[4], e.g. *noun* candidates are words that are tagged as NN or NNS. Table 4 shows the total number of candidates for each classifier.

	Classifier				
	Art.	**P**	**N**	**Agr.**	**F**
Train	254K	103K	240K	75K	175K
Test	6K	2.5K	2.6K	2.4K	4.8K

Table 4: **Number of candidate words by classifier type in training and test data.**

4 Interacting Mistakes

The approach of addressing each type of mistake individually is problematic when multiple phenomena interact. Consider the examples in Table 3 and the predictions made by the Illinois system. In the first and second sentences, there are two possible ways to correct the structures "such situation" and "cars is". In the former, either the article or the noun number should be changed; in the latter, either the noun number or the verb agreement marker[5]. In these examples, each of the independently-trained classifiers identifies the problem because each system makes a decision using the second error as part of its contextual cues, and thus the individual systems produce inconsistent predictions.

The second type of interaction concerns cases that require correcting more than one word at a time: the last example in Table 3 requires making changes both to the verb and the subject. Since each of the independent classifiers (for nouns and for verb agreement) takes into account the other word as part of its features, they both infer that the verb number is correct and that the grammatical subject "student" should be plural.

We refer to the words whose grammatical properties interact as *structures*. The independently-trained classifiers tend to fail to provide valid corrections in contexts where it is important to consider both words of the structure.

4.1 Structures for Joint Modeling

We address two linguistic structures that are relevant for the grammatical phenomena considered: *article-NPhead* and *subject-verb*. In the *article-NPhead* structures, the interaction is between the head of the noun phrase (NP) and the article that refers to the NP (first example in Table 3). In particular, the model should take into account that the article "a" cannot be combined with a noun in plural form. For subject-verb agreement, the subject and the verb should agree in number.

We now need to identify all pairs of candidates that form the relevant structures. *Article-NPhead* structures are pairs of words, such that the first word is a candidate of type article, while the second word is a noun candidate. Given an article candidate, the head of its NP is determined using the POS information (this information is obtained from the article feature vector because the NP head is a feature used by the article system)[6]. *Subject-verb* structures are pairs of noun-agreement candidates. Given a verb, its subject is identified with a dependency parser (Marneffe et al., 2006).

To evaluate the accuracy of subject and NP head

[3] http://cogcomp.cs.illinois.edu/page/software_view/Chunker

[4] http://cogcomp.cs.illinois.edu/page/software_view/POS

[5] Both of these solutions will result in grammatical output and the specific choice between the two depends on the wider essay context.

[6] Some heads are not identified or belong to a different part of speech.

predictions, a random sample of 500 structures of each type from the training data was examined by a human annotator with formal training in Linguistics. The human annotations were then compared against the automatic predictions. The results of the evaluation for subject-verb and article-NPhead structures are shown in Tables 5 and 6, respectively. Although the overall accuracy is above 90% for both structures, the accuracy varies by the distance between the structure components and drops significantly as the distance increases. For article-NPhead structures, *distance* indicates the position of the NP head with respect to the article, e.g. distance of 1 means that the head immediately follows the article. For subject-verb structures, *distance* is shown with respect to the verb: a distance of -1 means that the subject immediately precedes the verb. Although in most cases the subject is located to the left of the verb, in some constructions, such as existential clauses and inversions, it occurs after the verb.

Based on the accuracy results for identifying the structure components, we select those structures where the components are reliably identified. For article-NPhead, valid structures are those where the distance is at most three words. For subject-verb, we consider as valid those structures where the identified subject is located within two words to the left or three words to the right of the verb.

The valid structures are selected as input to the joint model (Sec. 5). The *joint learning* model considers only those valid structures whose components are *adjacent*. In adjacent structures the NP head immediately follows the article, and the verb immediately follows the subject. *Joint inference* is not restricted to adjacent structures.

The last column of Table 5 shows that valid subject-verb structures account for 67.5% of all verbs whose subjects are common nouns (51.7% are cases where the words are adjacent). Verbs whose subjects are common nouns account for 57.8% of all verbs that have subjects (verbs with different types of subjects, most of which are personal pronouns, are not considered here, since these subjects are not part of the noun classifier).

Valid article-NPhead structures account for 98.0% of all articles whose NP heads are common nouns (47.5% of those are adjacent structures), as shown in the last column of Table 6. 71.0% of articles in the training data belong to an NP whose head is a common noun; NPs whose heads belong to different parts of speech are not considered.

Note also that because a noun may belong both to an article-NPhead and a subject-verb structure, the structures contain an overlap.

Distance	**Accuracy**	**% of all subj. predictions**	**Cumul.**
-1	97.6%	51.7%	51.7%
1,2,3	100.0%	8.9%	60.6%
-2	88.2%	6.9%	67.5%
Other	80.8%	32.5%	100.0%

Table 5: **Accuracy of subject identification on a random sample of subject-verb structures from the training data.** The overall accuracy is 91.52%. For each distance, the following are shown: accuracy based on comparison with human evaluation; the percentage of all predictions that have this distance; the cumulative percentage.

Distance	**Accuracy**	**% of all head predictions**	**Cumul.**
1	94.8%	47.5%	47.5%
2	94.4%	44.0%	91.5%
3	92.3%	6.5%	98.0%
Other	89.1%	2.0%	100%

Table 6: **Accuracy of NP head identification on a random sample of article-NPhead structures from training data.** The overall accuracy is 94.45%. For each distance, the following are shown: accuracy based on comparison with human evaluation; the percentage of all predictions that have this distance; the cumulative percentage.

5 The Joint Model

In this section, we present the *joint inference* and the *joint learning* approaches. In the joint inference approach, we use the independently-learned models from the Illinois system, and the interacting target words identified earlier are considered only at inference stage. In the joint learning method, we jointly learn a model for the interacting phenomena.

The label space in the joint models corresponds to sequences of labels from the confusion sets of the individual classifiers: $\{a-sing, a-pl, the-sing, the-pl, \varnothing-sing, \varnothing-pl\}$ and $\{sing-sing, sing{-}pl, pl{-}sing, pl{-}pl\}$ for article-NPhead and subject-verb structures, respectively[7]. Invalid

[7]"sing" and "pl" refer to the grammatical number of noun

structures, such as *pl-sing* are excluded via hard constraints (when we run joint inference) or via implicit soft constraints (when we use joint learning).

5.1 Joint Inference

In the individual model approach, decisions are made for each word independently, ignoring the interactions among linguistic phenomena. The purpose of joint inference is to include linguistic (i.e. structural) knowledge, such as "plural nouns do not take an indefinite article", and "agreement consistency between the verb and the subject that controls it". This knowledge should be useful for resolving inconsistencies produced by individual classifiers.

The inference approach we develop in this paper follows the one proposed by Roth and Yih (2004) of training individual models and combining them at decision time via joint inference. The advantage of this method is that it allows us to build upon any existing independently-learned models that provide a distribution over their outcome, and produce a coherent global output that respects our declarative constraints. We formulate our component inference problems as integer linear program (ILP) instances as in Roth and Yih (2004).

The inference takes as input the individual classifiers' confidence scores for each prediction, along with a list of constraints. The output is the optimal solution that maximizes the linear sum of the confidence scores, subject to the constraints that encode the interactions. The joint model thus selects a hypothesis that both obtains the best score according to the individual models and satisfies the constraints that reflect the interactions among the grammatical phenomena at the level of linguistic structures, as defined in Sec. 4.

Inference The joint inference is enforced at the level of structures, and each structure corresponds to one ILP instance. All structures consist of two or three words: when an article-NPhead structure and a subject-verb structure include the same noun, the structure input to the ILP consists of an article-noun-verb triple. We formulate the inference problem as follows: Given a structure s that consists of n words, let w_i correspond to the i^{th} word in the structure. Let h denote a hypothesis from the hypothesis space H for s, and $score(w_i, h, l^i)$ denote the score assigned by the appropriate error-specific model to w_i under h for label l from the confusion set of word w_i. We denote by $e_{w,l}$ the Boolean variable that indicates whether the prediction on word w is assigned the value l ($e_{w,l} = 1$) or not ($e_{w,l} = 0$).

We assume that each independent classifier returns a score that corresponds to the likelihood of word w_i under h being labeled l^i. The softmax function (Bishop, 1995) is used to convert raw activation scores to conditional probabilities for the discriminative article model. The NB scores are also normalized and correspond to probabilities. Then the inference task is solved by maximizing the overall score of a candidate assignment of labels l to words w (this set of feasible assignments is denoted H here) subject to the constraints C for the structure s:

$$\hat{h} = \arg\max_{h \in H} score(h) =$$

$$= \arg\max_{h \in H} \sum_{i=1}^{n} score(w_i, h, l^i) e_{w_i, l^i}$$

subject to $C(s)$

Constraints In the $\{0, 1\}$ linear programming formulation described above, we can encode linguistic constraints that reflect the interactions among the linguistic phenomena. The inference enforces the following structural and linguistic constraints:

1. The indefinite article "a" cannot refer to an NP headed by a plural noun.
2. Subject and verb must agree in number.

In addition, we encode "legitimacy" constraints, that make sure that each w is assigned a single label. All constraints are encoded as hard constraints.

5.2 Joint Learning

We now describe how we *learn* the subject-verb and article-NPhead structures jointly. The joint model is implemented as a NB classifier and is trained in the same way as the independent models on the Google corpus with word n-gram features. Unlike the independent models, where the target corresponds to one

and verb agreement candidates. The candidates themselves are the surface forms of specific words that realize these grammatical properties. Note that a subject in subject-verb structures is always third person, since all subjects in subject-verb structures are common nouns; other subjects, including pronouns, are excluded. Thus the agreement distinction is singular vs. plural.

System	Adjacent structures		All distances	
	F1 (Orig.)	F1 (Revised)	F1 (Orig.)	F1 (Revised)
Illinois	31.20	42.14	31.20	42.14
NaïveVerb	31.19	42.20	31.13	42.16
NaïveNoun	31.03	41.87	30.91	41.70
This paper joint systems	**Joint Inference** (adjacent)		**Joint Inference** (all distances)	
	F1 (Orig.)	F1 (Revised)	F1 (Orig.)	F1 (Revised)
Subject-verb	31.90	42.94	31.97	42.86
Article-NPhead	31.63	42.48	31.79	42.59
Subject-verb + article-NPhead	**32.35**	**43.16**	**32.51**	**43.19**

Table 7: **Joint Inference Results.** All results are on the CoNLL-2013 test data using the original and revised gold annotations. *Adjacent* denotes a setting, where the joint inference is applied to structures with consecutive components (article-NPhead or subject-verb). *All distances* denotes a setting, where the constraints are applied to all valid structures, as described in Sec. 4.1. *Illinois* denotes the result obtained by the top CoNLL-2013 shared task system. In all cases, the candidates that are not part of the structures are handled by the respective components of the Illinois system. *NaïveVerb* and *NaïveNoun* denote heuristics, where a verb or subject are changed to ensure agreement. All improvements over the Illinois system are statistically significant (McNemar's test, $p < 0.01$).

word, here the target corresponds to two words that are part of the structure and the label space of the model is modified accordingly. Since we use features that can be computed from the small windows in the Google corpus, the joint learning model handles only adjacent structures (Sec. 4.1). Because the target consists of two words and the Google corpus contains counts for n-grams of length at most five, the features are collected in the three word window around the target.[8]

Unlike with the joint inference, here we do not explicitly encode linguistic constraints. One reason for this is that the NP head and subject predictions are not 100% accurate, so input structures will have noise. However, the joint model learns these constraints through the evidence seen in training.

6 Experiments

In this section, we describe our experimental setup and evaluate the performance of the joint approach. In the joint approach, the joint components presented in Sec. 5 handle the interacting structures described in Sec. 4. The individual classifiers of the Illinois system make predictions for the remaining words. The research question addressed by the experiments is the following: Given independently-trained systems for different types of errors, can we improve the performance by considering the phenomena that interact jointly? To address this, we report the results in the following settings:

1. *Joint Inference*: we compare the Illinois system that is a collection of individually-trained models that are applied independently with a model that uses joint inference encoded as declarative constraints in the ILP formulation and show that using joint inference results in a strong performance gain.
2. *Joint Learning*: we compare the Illinois system with a model that incorporates jointly-trained components for the two linguistic structures that we described in Sec. 4. We show that joint training produces an even stronger gain in performance compared to the Illinois model.
2. *Joint Learning and Inference*: we apply joint inference to the output of the joint learning system to account for dependencies not covered by the joint learning model.

We report F1 performance scored using the official scorer from the shared task (Dahlmeier and Ng, 2012). The task reports two types of evaluation: on the original gold data and on gold data with additional corrections. We refer to the results as *Original* and *Revised*.

6.1 Joint Inference Results

Table 7 shows the results of applying joint inference to the Illinois system. Both the article-NPhead and the subject-verb constraints improve the performance. The results for the joint inference are shown in two settings, adjacent and all structures, so that later we can compare joint inference with the joint learning model that handles only adjacent structures.

[8] Also note that when the article is ∅, the surface form of the structure corresponds to the NP head alone; this does not present a problem because in the NB model the context counts are normalized with the prior counts.

	Illinois system		Illinois-NBArticle	
	F1 (Orig.)	**F1 (Revised)**	**F1 (Orig.)**	**F1 (Revised)**
Illinois	31.20	42.14	31.71	41.38
This paper joint systems	**Joint Learning** (adjacent)		**Joint Learning** (adjacent)	
	F1 (Orig.)	**F1 (Revised)**	**F1 (Orig.)**	**F1 (Revised)**
Subject-verb	32.64*	43.37*	33.09*	**42.78***
Article-NPhead	33.89*	42.57*	33.16*	41.51
Subject-verb + article-NPhead	**35.12***	**43.73***	**34.41***	42.76*

Table 8: **Joint Learning Results.** All results are on the CoNLL-2013 test data using the original and revised gold annotations. *Illinois-NBArticle* denotes the Illinois system, where the discriminative article model is replaced with a NB classifier. *Adjacent* denotes a setting, where the structure components are consecutive (article-NPhead or subject-verb), as described in Sec. 4.1. *Illinois* denotes the result obtained by the top CoNLL-2013 shared task system. In all cases, the candidates that are not part of the structures are handled by the respective components of the Illinois system. Statistically significant improvements (McNemar's test, $p < 0.01$) over the Illinois system are marked with an asterisk (*).

It is also interesting to note that the key improvement comes from considering structures whose components are adjacent. This is not surprising given that the accuracy for subject and NP head identification drops as the distance increases.

For subject-verb constraints, we also implement a naïve approach that looks for contradictions and changes either the verb or the subject if they do not satisfy the number agreement. These two heuristics are denoted as *NaïveVerb* and *NaïveNoun*. The heuristics differ from the joint inference in that they enforce agreement by always changing either the noun (*NaïveNoun*) or the verb (*NaïveVerb*), while the joint inference does this using the scores produced by the independent models. In other words, the key is the objective function, while the components of the objective function are the same in the heuristics and the joint inference. The results in Table 7 show that simply enforcing agreement does not work well and that the ILP formulation is indeed effective and improves over the independently-trained models in all cases.

Recall that valid structures include only those whose components can be identified in a reliable way (Sec. 4.1). To evaluate the impact of that filtering, we perform two experiments with subject-verb structures (long-distance dependencies are more common in those constructions than in the article-NPhead structures): first, we apply joint inference to all subject-verb structures. We obtain F1 scores of 31.61 and 42.28, on original and revised gold data, respectively, which is significantly worse than the results on subject-verb structures in Table 7 (31.97 and 42.86, respectively) and only slightly better than the baseline performance of the Illinois system. Furthermore, when we apply joint inference to those structures which were excluded by filtering in Sec. 4.1, we find that the performance degrades compared to the Illinois system (30.85 and 41.58). These results demonstrate that the joint inference improvements are due to structures whose components can be identified with high accuracy and that it is essential to identify these structures; bad structures, on the other hand, hurt performance.

6.2 Joint Learning Results

Now we show experimental results of the joint learning (Table 8). Note that the joint learning component considers only those structures where the words are adjacent. Because the Illinois system presented in Sec. 3 makes use of a discriminative article model, while the joint model uses NB, we also show results, where the article model is replaced by a NB classifier trained on the Google corpus. In all cases, joint learning demonstrates a strong performance gain.

6.3 Joint Learning and Inference Results

Finally, we apply joint inference to the output of the joint learning system in Sec. 6.2. Table 9 shows the results of the Illinois model, the model that applies joint inference and joint learning separately, and both. Even though the joint learning performs better than the joint inference, the joint learning covers only adjacent structures. Furthermore, joint learning does not address overlapping structures of triples that consist of article, subject, and verb (6% of all structures). Joint inference allows us to ensure consistent predictions in cases not addressed by the

Example	Illinois system	JL and JI
"Moreover, the increased **technologies help** people to overcome different natural disasters.	No change	technology helps
"At that time,... **there are surveillances** in everyone's heart and criminals are more difficult to hide."	there are* surveillance*	there is surveillance
"In **such situation**, individuals will lose their basic privacy."	such a* situations*	such a situation
"In supermarket **monitor is** needed because we have to track thieves."	No change	monitors are

Table 10: **Examples of mistakes that are corrected by the joint model but not by the Illinois model.** *Illinois* denotes the result obtained by the top CoNLL-2013 shared task system from the University of Illinois. *JL* and *JI* stand for joint learning and joint inference, respectively. Inconsistent predictions are starred.

	F1 (Orig.)	F1 (Revised)
Illinois	31.20	42.14
Joint Inference	32.51	43.19
Joint Learning	35.12	43.73
Joint Learn. + Inf.	**35.21**	**43.74**

Table 9: **Joint Learning and Inference.** All results are on the CoNLL-2013 test data using the original and revised gold annotations. Results of the joint models that include the joint inference component are shown for structures of all distances. *Illinois* denotes the result obtained by the top CoNLL-2013 shared task system. All joint systems demonstrate a statistically significant improvement over the Illinois system; joint learning improvements are also statistically significant compared to the joint inference results (McNemar's test, $p < 0.01$).

joint learning model. Indeed, we can get a small improvement by adding joint inference on top of the joint learning on original annotations. Since the revised corrections are based on the participants' input and are most likely biased towards system predictions for corrections missed by the original annotators (Ng et al., 2013), it is more difficult to show improvement on revised data.

7 Discussion and Error Analysis

In the previous section, we evaluated the proposed joint inference and joint learning models that handle interacting grammatical phenomena. We showed that the joint models produce significant improvements over the highest-scoring CoNLL-2013 shared task system that consists of independently-trained classifiers: the joint approaches increase the F1 score by 4 F1 points on the original gold data and almost 2 points on the revised data (Table 9).

These results are interesting from the point of view of developing a practical error correction system. However, recall that the errors in the interacting structures are only a subset of mistakes in the CoNLL-2013 data set but the evaluation in Sec. 6 is performed with respect to all of these errors. From a scientific point of view, it is interesting to evaluate the impact of the joint models more precisely by considering the improvements on the relevant structures only. Table 11 shows how much the joint learning approach improves on the subset of relevant mistakes.

Structure	Performance (F1)	
	Illinois	Joint Learning
Subject-verb	39.64	**52.25**
Article-NPhead	30.65	**35.90**

Table 11: **Evaluation of the joint learning performance on the subset of the data containing interacting errors.** All results are on the CoNLL-2013 test data using the original annotations. *Illinois* denotes the result obtained by the top CoNLL-2013 shared task system. All improvements are statistically significant over the Illinois system (McNemar's test, $p < 0.01$).

Error Analysis To better understand where the joint models have an advantage over the independently-trained classifiers, we analyze the output produced by each of the approaches. In Table 10 we show examples of mistakes that the model that uses joint learning and inference is able to identify correctly, along with the original predictions made by the Illinois system.

Joint Inference vs. Joint Learning We wish to stress that the joint approaches do not simply perform better but also make coherent decisions by disallowing illegitimate outputs. The joint inference approach does this by enforcing linguistic constraints on the output. The joint learning model, while not explicitly encoding these constraints, learns them from the distribution of the training data.

Joint inference is a less expensive model, since it uses the scores produced by the individual classifiers and thus does not require additional training. Joint learning, on the other hand, is superior to joint inference, since it is better at modeling interactions where multiple errors occur simultaneously – it eliminates the noisy context present when learning the independent classifiers. Consider the first example from Table 10, where both the noun and the agreement classifiers receive noisy input: the verb "help" and the noun "technologies" act as part of input features for the noun and agreement classifiers, respectively. The noisy features prevent both modules from identifying the two errors.

Finally, an important distinction of the joint learning method is that it considers all possible output sequences *in training*, and thus it is able to better identify errors that require multiple changes, such as the last example in Table 10, where the Illinois system proposes no changes.

7.1 Error Correction: Challenges

We finalize our discussion with a few comments on the challenges of the error correction task.

Task Difficulty As shown in Table 1 in Sec. 2, only a small percentage of words have mistakes, while over 90% (about 98% in training) are used correctly. The low error rates are the key reason the error correction task is so difficult: it is quite challenging for a system to improve over a writer that already performs at the level of over 90%. Indeed, very few NLP tasks already have systems that perform at that level, even when the data is not as noisy as the ESL data.

Evaluation Metrics In the CoNLL-2013 competition, as well as the competitions alluded to earlier, systems were compared on F1 performance, and, consequently, this is the metric we optimize in this paper. Practical error correction systems, however, should be tuned to minimize recall to guarantee that the overall quality of the text does not go down. Indeed, the error sparsity makes it very challenging to identify mistakes accurately, and no system in the shared task achieves a precision over 50%. However, once the precision drops below 50%, the system introduces more mistakes than it identifies.

Clearly, optimizing the F1 measure does not ensure that the quality of the text improves as a result of running the system. Thus, it can be argued that the F1 measure is not the right measure for error correction. A different evaluation metric based on the *accuracy* of the data before and after running the system was proposed in Rozovskaya and Roth (2010c). When optimizing for this metric, the noun module, for instance, at recall point 20%, achieves a precision of 63.93%. This translates into accuracy of 94.46%, while the baseline on noun errors in the test data (i.e. the accuracy of the data before running the system) is 94.0% (Table 1). This means that the system improves the quality of the data.

Annotation Lastly, we believe that it is important to provide alternative corrections, as the agreement on what constitutes a mistake even among native English speakers can be quite low (Madnani et al., 2011).

8 Conclusion

This work presented the first successful study that jointly corrects grammatical mistakes. We addressed two pairs of interacting phenomena and showed that it is possible to reliably identify their components, thereby facilitating the joint approach.

We described two joint methods: a *joint inference* approach implemented via ILP and a *joint learning* model. The joint inference enforces constraints using the scores produced by the independently-trained models. The joint learning model learns the interacting phenomena as structures. The joint methods produce a significant improvement over a state-of-the-art system that combines independently-trained models and, importantly, produce linguistically legitimate output.

Acknowledgments

The authors thank Peter Chew, Jennifer Cole, Mark Sammons, and the anonymous reviewers for their helpful feedback. The authors thank Josh Gioja for the code that performs phonetic disambiguation of the indefinite article. This material is based on research sponsored by DARPA under agreement number FA8750-13-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

C. Bishop. 1995. *Neural Networks for Pattern Recognition, chapter 6.4: Modelling conditional distributions*. Oxford University Press.

T. Brants and A. Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia, PA.

C. Brockett, D. B. William, and M. Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia, July. Association for Computational Linguistics.

A. J. Carlson, J. Rosen, and D. Roth. 2001. Scaling up context sensitive text correction. In *IAAI*.

J. Clarke and M. Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*.

D. Dahlmeier and H.T Ng. 2012. A beam-search decoder for grammatical error correction. In *EMNLP-CoNLL*, Jeju Island, Korea, July. Association for Computational Linguistics.

D. Dahlmeier, H.T. Ng, and S.M. Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proc. of the NAACL HLT 2013 Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, Georgia, June. Association for Computational Linguistics.

R. Dale and A. Kilgarriff. 2011. Helping Our Own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

R. Dale, I. Anisimoff, and G. Narroway. 2012. A report on the preposition and determiner error correction shared task. In *Proc. of the NAACL HLT 2012 Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, Montreal, Canada, June. Association for Computational Linguistics.

R. De Felice and S. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK, August.

Y. Freund and R. E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W. Dolan, D. Belenko, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*.

M. Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *NAACL*, pages 163–171, Los Angeles, California, June.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*.

N. Han, M. Chodorow, and C. Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Journal of Natural Language Engineering*, 12(2):115–129.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 145–148, Sapporo, Japan, July.

J. Lee and S. Seneff. 2008. Correcting misuse of verb forms. In *ACL*, pages 174–182, Columbus, Ohio, June. Association for Computational Linguistics.

N. Madnani, M. Chodorow, J. Tetreault, and A. Rozovskaya. 2011. They can help: Using crowdsourcing to improve the evaluation of grammatical error detection systems. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 508–513, Portland, Oregon, USA, June. Association for Computational Linguistics.

M. Marneffe, B. MacCartney, and Ch. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

A. Martins, Noah N. Smith, M. Figueiredo, and P. Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 238–249, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

H. T. Ng, S. M. Wu, Y. Wu, Ch. Hadiwinoto, and J. Tetreault. 2013. The conll-2013 shared task on grammatical error correction. In *Proc. of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

A. Park and R. Levy. 2011. Automated whole sentence grammar correction using a noisy channel model. In *ACL*, Portland, Oregon, USA, June. Association for Computational Linguistics.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *CoNLL*.

A. Rozovskaya and D. Roth. 2010a. Annotating ESL errors: Challenges and rewards. In *Proceedings of the*

NAACL Workshop on Innovative Use of NLP for Building Educational Applications.

A. Rozovskaya and D. Roth. 2010b. Generating confusion sets for context-sensitive error correction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A. Rozovskaya and D. Roth. 2010c. Training paradigms for correcting errors in grammar and usage. In *NAACL*.

A. Rozovskaya and D. Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *ACL*.

A. Rozovskaya, M. Sammons, J. Gioja, and D. Roth. 2011. University of Illinois system in HOO text correction shared task.

A. Rozovskaya, M. Sammons, and D. Roth. 2012. The UI system in the HOO 2012 shared task on error correction.

A. Rozovskaya, K.-W. Chang, M. Sammons, and D. Roth. 2013. The University of Illinois system in the CoNLL-2013 shared task. In *CoNLL Shared Task*.

C. Sutton and A. McCallum. 2007. Piecewise pseudolikelihood for efficient training of conditional random fields. In Zoubin Ghahramani, editor, *ICML*.

J. Tetreault and M. Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 865–872, Manchester, UK, August.

Y. Wu and H.T. Ng. 2013. Grammatical error correction using integer linear programming. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Sofia, Bulgaria, August. Association for Computational Linguistics.

With blinkers on: robust prediction of eye movements across readers

Franz Matties and Anders Søgaard
University of Copenhagen
Njalsgade 142
DK-2300 Copenhagen S
Email: soegaard@hum.ku.dk

Abstract

Nilsson and Nivre (2009) introduced a tree-based model of persons' eye movements in reading. The individual variation between readers reportedly made application across readers impossible. While a tree-based model seems plausible for eye movements, we show that competitive results can be obtained with a linear CRF model. Increasing the inductive bias also makes learning across readers possible. In fact we observe next-to-no performance drop when evaluating models trained on gaze records of multiple readers on new readers.

1 Introduction

When we read a text, our gaze does not move smoothly and continuously along its lines. Rather, our eyes fixate at a word, then skip a few words, to jump to a new fixation point. Such rapid eye movements are called *saccades*. Sometimes we even jump backwards. Backward saccades are called *regressions*. Gaze can be recorded using eye tracking devices (Starr and Rayner, 2001). Since eye movements in reading give us important information about what readers find complicated in a text, and what readers find completely predictable, predicting eye movements on new texts has many practical applications in text-to-text generation and human computer interaction, for example.

The problem of predicting eye movements in reading is, for a reader r_i and a given sequence of word tokens $w_1 \dots w_n$, to predict a set of fixation points $F \subseteq \{w_1, \dots, w_n\}$, i.e., the fixation points of r_i's gaze. For each token w_j, the reader r_i may skip w_j or fixate at w_j. Models are evaluated on recordings of human reading obtained using eye tracking devices. The supervised prediction problem that we consider in this paper, also uses eye tracking data for learning models of eye movement.

Nilsson and Nivre (2009) first introduced this supervised learning task and used the Dundee corpus to train and evaluate a tree-based model, essentially treating the problem of predicting eye movements in reading as transition-based dependency parsing.

We follow Hara et al. (2012) in modeling only forward saccades and *not* regressions and refixations. While Nilsson and Nivre (2009) try to model a subset of regressions and refixations, they do *not* evaluate this part of their model focusing only on fixation accuracy and distribution accuracy, i.e., they evaluate how well they predict *a set of* fixation points rather than a sequence of points in order. This enables us to model eye movements in reading as a sequential problem of determining the length of forward saccades, increasing the inductive bias of our learning algorithm in a motivated way. Note that because we work with visual input, we do not tokenize our input in our experiments, i.e., punctuation does not count as input tokens.

Example Figure 1 presents an example sentence and gaze records from the Dundee corpus. The Dundee corpus contains gaze records of 10 readers in total. Note that there is little consensus on what words are skipped. 5/10 readers skip the first word. Generally, closed class items (prepositions, copulae, quantifiers) seem to be skipped more open, but we do see a lot of individual variation. While others for this reason have refrained from evaluation across readers (Nilsson and Nivre, 2009; Hara et al., 2012),

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 803–807,
Seattle, Washington, USA, 18-21 October 2013.

	Sentence									
	Are	tourists	enticed	by	these	attractions	threathening	their	very	existence?
r_1	Fixate	Fixate	Fixate	Skip	Fixate	Fixate	Fixate	Skip	Fixate	Fixate
r_2	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate
r_3	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Skip	Fixate
r_4	Skip	Fixate	Fixate	Skip	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate
r_5	Skip	Fixate	Fixate	Skip	Fixate	Fixate	Fixate	Skip	Fixate	Fixate
r_6	Skip	Fixate	Fixate	Skip	Fixate	Fixate	Fixate	Fixate	Skip	Fixate
r_7	Skip	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate
r_8	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate
r_9	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate
r_{10}	Skip	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Fixate	Skip	Fixate
# skips	5	0	0	4	0	0	0	2	3	0

Figure 1: The gaze records of the three first readers for the first sentence in the Dundee corpus.

we show that our model predicts gaze better *across readers* than a previously proposed model (Nilsson and Nivre, 2009) does training and evaluating on the *same* readers. A final observation is that fixations are very frequent at the word level – in fact, even skilled readers make 94 fixations per 100 words (Starr and Rayner, 2001) – which motivates using F_1-score of skips as metric. We follow Nilsson and Nivre (2009) in reporting word-level accuracy, but find it particularly interesting that the simple model proposed here outperforms previous models by a large margin in F_1-score over skips.

Related work Below we use a sequential model rather than a tree-based model to bias our model toward predicting forward saccades. Nilsson and Nivre (2009), in contrast, present a more expressive tree-based model for modeling eye movements, with some constraints on the search space. The transition-based model uses consecutive classification rather than structured prediction. The features used in their model are very simple. In particular, they use use word lengths and frequencies, like us, as well as distances between tokens (important in a transition-based model), and, finally, the history of previous decisions.

Hara et al. (2012) use a linear CRF model for the same problem, like us, but they consider a slightly different problem, namely that of predicting eye movement when reading text on a specific screen. They therefore use screen position as a feature. In addition, they use word forms, POS, various measures of surprise of word length, as well as perplexity of bi- and trigrams. The features relating to screen position were the most predictive ones.

2 Our approach

We use linear CRFs to model eye movements in reading. We follow Hara et al. (2012) in using small window sizes (at most five words) for extracting features. Rather than using word forms, POS, etc., we use only word length and the log probability of words – both known to correlate well with likelihood of fixation, as well as fixation times (McDonald and Shillcock, 2012; Kliegl et al., 2004; Reingold et al., 2012). The model thus reflects a hypothesis that eye movements are largely unaffected by semantic content, that eye movements depend on the physical properties and frequency of words, and that there is a sequential dependence between fixation times. Tabel 1 gives the complete set of features. We also evaluated using word forms and POS on held-out data, but this did not lead to improvements. There is evidence for the impact of morphology on eye movements (Liversedge and Blythe, 2007; Bertram, 2011), but we did not incorporate this into our model. Finally, we did not incorporate predictability of tokens, although this is also known to correlate with fixation times (Kliegl et al., 2004). Hara et al. (2012) use perplexity features to capture this.

We use a publicly available implementation of linear CRFs[1] with default parameters (L_2-regularized, $C = 1$).

[1] https://code.google.com/p/crfpp/

3 Predicting a reader's eye movements

In this experiment we consider exactly the same setup as Nilsson and Nivre (2009) considered. In the Dundee corpus, we have gaze data for 10 persons. The corpus consists of 2,379 sentences, 56,212 tokens and 9,776 types. The gaza data was recorded using a Dr. Bouis Oculometer Eyetracker, sampling the position of the right eye every millisecond. We use texts 1–16 (1911 sentences) for training, 17–18 (237 sentences) for development and 19–20 (231 sentences) for testing.

Results are presented in Table 2 and are slightly better than Nilsson and Nivre (2009), mainly because of better predictions of skips. Our error reduction over their model in terms of F_1 over skips is 9.4%. The baseline model used in Nilsson and Nivre (2009), the E-Z Reader (Reichle et al., 1998), obtained a fixation accuracy of 57.7%.

4 Predicting across readers

Hara et al. (2012) consider the problem of learning from the concatenation of the gaze data from the 10 persons in the Dundee corpus, but they also evaluate on data from these persons. In our second experiment, we consider the more difficult problem of learning from one person's gaze data, but evaluating on gaze data from another test person. This is a more realistic scenario if we want to use our model to predict eye movements in reading on anyone but our test persons. This has been argued to be impossible in previous work (Nilsson and Nivre, 2009; Hara et al., 2012).

Our results are presented in Table 3. Interestingly, results are very robust across reader pairs. In fact, only in 4/10 cases do we get the best results training on gaze data from the reader we evaluate on. Note also that the readers seem to form two groups – (a, b, h, i, j) and (c, d, e, f, g) – that provide good training material for each other. Training on concatenated data from all members in each group may be beneficial.

5 Learning from multiple readers

In our final experiment, we learn from the gaze records of nine readers and evaluate on the tenth. This is a realistic evaluation of our ability to predict fixations for new, previously unobserved readers. Interestingly we can predict the fixations of new readers better than Nilsson and Nivre (2009) predict fixations when the training and test data are produced by the same reader. The results are presented in Table 4. In fact our skip F_1 score is actually better than in our first experiments. As already mentioned, this result can probably be improved by using a subset of readers or by weighting training examples, e.g., by importance weighting (Shimodaira, 2000). For now, this is left for future work.

6 Discussion

Our contributions in this paper are: (i) a model for predicting a reader's eye movements that is competitive to state-of-the-art, but simpler, with a smaller search space than Nilsson and Nivre (2009) and a smaller feature model than Hara et al. (2012), (ii) showing that the simpler model is robust enough to model eye movements across readers, and finally, (iii) showing that even better models can be obtained training on records from multiple readers.

It is interesting that a model without lexical information is more robust across readers. This suggests that deep processing has little impact on eye movements. See Starr and Rayner (2001) for discussion. The features used in this study are well-motivated and account as well for the phenomena as previously proposed models. It would be interesting to incorporate morphological features and perplexity-based features, but we leave this for future work.

7 Conclusion

This study is, to the best of our knowledge, the first to consider the problem of learning to predict eye movements in reading across readers. We present a very simple model of eye movements in reading that performs a little better than Nilsson and Nivre (2009) in terms of fixation accuracy, evaluated on one reader at a time, but predicts skips significantly better. The true merit of the approach, however, is its ability to predict eye movements across readers. In fact, it predicts the eye movements of new readers better than Nilsson and Nivre (2009) do when the training and test data are produced by the same reader.

References

Raymond Bertram. 2011. Eye movements and morphological processing in reading. *The Mental Lexicon*, 6:83–109.

Tadayoshi Hara, Daichi Mochihashi, Yoshinobu Kano, and Akiko Aizawa. 2012. Predicting word fixation in text with a CRF model for capturing general reading strategies among readers. In *Workshop on Eye-tracking and NLP, COLING*.

Reinhold Kliegl, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology*, 16:262–284.

Simon Liversedge and Hazel Blythe. 2007. Lexical and sublexical influences on eye movements during reading. *Language and Linguistic Compass*, 1:17–31.

Scott McDonald and Richard Shillcock. 2012. Eye movements reveal the on-line computation of lexical probabilities during reading. *Psychological Science*, 14:648–652.

Matthias Nilsson and Joakim Nivre. 2009. Learning where to look: Modeling eye movements in reading. In *CoNLL*.

Erik Reichle, Alexander Pollatsek, Donald Fisher, and Keith Rayner. 1998. Toward a model of eye movement control in reading. *Psychological Review*, 105:125–157.

Eyal Reingold, Erik Reichle, Mackenzie Glaholt, and Heather Sheridan. 2012. Direct lexical control of eye movements in reading. *Cognitive Psychology*, 65:177–206.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

Matthew Starr and Keith Rayner. 2001. Eye movements during reading: some current controversies. *Trends in Cognitive Science*, 5:156–163.

Feature		Description
WordLength	$\{L_{-2}, L_{-1}, L_0, L_1, L_2\}$	The number of letters for a token
WordProbability	$\{P_{-1}, P_0, P_1\}$	The log probability of a word (rounded) as given in the Dundee data

Table 1: Feature template

	Fixation Accuracy		Fixations (F1)		Skips (F1)	
Reader	N&N	Model	N&N	Model	N&N	Model
a	70.0	70.2	71.8	70.0	67.4	70.3
b	66.5	66.2	74.1	71.2	75.0	58.8
c	70.9	70.4	77.3	74.7	59.4	64.4
d	78.9	76.5	84.7	81.3	65.9	68.5
e	71.8	70.5	73.5	69.9	69.9	71.0
f	67.9	66.4	76.8	72.8	47.7	55.8
g	56.6	65.1	61.7	61.8	49.9	67.8
h	66.9	67.7	72.7	70.3	58.2	64.6
i	69.1	71.5	74.1	73.9	60.7	68.8
j	76.3	74.6	82.0	77.3	65.2	71.1
average	69.5	**69.9**	**75.2**	72.3	62.6	**66.1**

Table 2: Comparison between NN09 and our model.

train/test	a	b	c	d	e	f	g	h	i	j
a	-	67.2	67.6	71.5	69.7	63.4	64.9	66.9	70.7	72.6
b	67.7	-	70.1	**76.9**	68.0	65.7	62.9	67.1	69.1	72.8
c	69.3	67.3	-	**76.5**	69.7	65.1	64.3	67.4	71.0	74.2
d	69.0	67.2	70.0	-	69.1	65.1	63.9	67.3	70.1	73.9
e	70.1	66.6	67.5	71.2	-	63.8	64.7	66.9	70.9	72.6
f	66.5	65.9	69.1	**76.7**	66.5	-	62.4	66.8	68.6	71.4
g	69.7	67.1	67.2	69.5	69.6	61.6	-	67.8	70.3	70.3
h	**70.5**	67.5	69.3	74.7	70.5	64.2	64.5	-	70.8	74.2
i	**70.9**	**68.1**	69.6	74.4	**70.7**	64.0	64.6	68.0	-	74.2
j	**70.7**	**68.0**	69.5	74.7	70.4	64.1	64.7	**68.2**	**71.5**	-

Table 3: Results learning across readers. Bold-faced numbers better than when training on same reader

	Fixation Accuracy		Fixations (F1)		Skips (F1)	
Reader	N&N	Model	N&N	Model	N&N	Model
a	70.0	70.3	71.8	72.1	67.4	68.2
b	66.5	67.9	74.1	70.6	75.0	64.6
c	70.9	69.8	77.3	73.1	59.4	65.6
d	78.9	75.5	84.7	79.5	65.9	69.5
e	71.8	70.6	73.5	72.0	69.9	69.0
f	67.9	64.5	76.8	68.6	47.7	59.2
g	56.6	64.7	61.7	65.0	49.9	64.5
h	66.9	68.1	72.7	70.9	58.2	64.8
i	69.1	71.3	74.6	74.1	60.7	67.9
j	76.3	74.2	82.0	77.2	65.2	70.4
average	69.5	**69.7**	**75.2**	72.3	62.6	**66.4**

Table 4: Comparison of NN09 and our cross-reader model trained on nine readers

Using Paraphrases and Lexical Semantics to Improve the Accuracy and the Robustness of Supervised Models in Situated Dialogue Systems

Claire Gardent
CNRS/LORIA, Nancy
claire.gardent@loria.fr

Lina M. Rojas Barahona
Université de Lorraine/LORIA, Nancy
lina.rojas@loria.fr

Abstract

This paper explores to what extent lemmatisation, lexical resources, distributional semantics and paraphrases can increase the accuracy of supervised models for dialogue management. The results suggest that each of these factors can help improve performance but that the impact will vary depending on their combination and on the evaluation mode.

1 Introduction

One strand of work in dialog research targets the rapid prototyping of virtual humans capable of conducting a conversation with humans in the context of a virtual world. In particular, question answering (QA) characters can respond to a restricted set of topics after training on a set of dialogs whose utterances are annotated with dialogue acts (Leuski and Traum, 2008).

As argued in (Sagae et al., 2009), the size of the training corpus is a major factor in allowing QA characters that are both robust and accurate. In addition, the training corpus should arguably be of good quality in that (i) it should contain the various ways of expressing the same content (paraphrases) and (ii) the data should not be skewed. In sum, the ideal training data should be large (more data is better data) ; balanced (similar amount of data for each class targeted by the classifier) and varied (it should encompass the largest possible number of paraphrases and synonyms for the utterances of each class).

In this paper, we explore different ways of improving and complementing the training data of a supervised QA character. We expand the size and the quality (less skewed data) of the training corpus using paraphrase generation techniques. We compare the performance obtained on lemmatised vs. non lemmatised data. And we investigate how various resources (synonym dictionaries, WordNet, distributional neighbours) can be used to handle unseen words at run time.

2 Related work

Previous work on improving robustness of supervised dialog systems includes detecting and handling out of domain utterances for generating feedback (Lane et al., 2004) ; using domain-restricted lexical semantics (Hardy et al., 2004) ; and work on manual data expansion (DeVault et al., 2011). Our work follows up on this research but provides a systematic investigation of how data expansion, lemmatisation and synonym handling impacts the performance of a supervised QA engine.

3 Experimental Setup

We run our experiments on a dialog engine developed for a serious game called *Mission Plastechnologie*. In this game, the player must interact with different virtual humans through a sequence of 12 subdialogs, each of them occurring in a different part of the virtual world.

Training Data. The training corpus consists of around 1250 Human-Human dialogues which were manually annotated with dialog moves. As the following dialog excerpt illustrates, the dialogs are conducted in French and each dialog turn is manually annotated using a set of 28 dialog acts. For

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 808–813,
Seattle, Washington, USA, 18-21 October 2013.

a more detailed presentation of the training corpus and of the annotation scheme, the reader is referred to (Rojas-Barahona et al., 2012a)

dialog : 01_dialogDirecteur-Tue Jun 14 11 :04 :23 2011

>M.Jasper : Bonjour, je suis M.Jasper le directeur. || ***greet***

(Hello, I am the director, Mr. Jasper.)

>M.Jasper : Qu'est-ce que je peux faire pour vous ? || ***ask(task(X))***

(What can I do for you ?)

>Lucas : je dois sauver mon oncle || ***first_step***

(I must rescue my uncle)

>M.Jasper : Pour faire votre manette, il vous faut des plans. Allez voir dans le bureau d'études, ils devraient y être. || ***inform(do(first_step))***

(To build the joystick you will need the plans. You will find them in the Designing Office.)

>M.Jasper : Bonne Chance ! || ***quit***

(Good Luck !)

Dialog Systems For our experiments, we use a hybrid dialog system similar to that described in (Rojas Barahona et al., 2012b; Rojas Barahona and Gardent, 2012). This system combines a classifier for interpreting the players utterances with an information state dialog manager which selects an appropriate system response based on the dialog move assigned by the classifier to the user turn. The classifier is a logistic regression classifier [1] which was trained for each subdialog in the game. The features used for training are the set of content words which are associated with a given dialog move and which remain after TF*IDF [2] filtering. Note that in this experiment, we do not use contextual features such as the dialog acts labeling the previous turns. There are two reasons for this. First, we want to focus on the impact of synonym handling, paraphrasing and lemmatisation on dialog management. Removing contextual features allows us to focus on how content features (content words) can be improved by these mechanisms. Second, when evaluating on the H-C corpus (see below), contextual features are often incorrect (because the system might incorrectly interpret and thus label a user turn). Excluding contextual features from training allows for a fair comparison between the H-H and the H-C evaluation.

Test Data and Evaluation Metrics We use accuracy (the number of correct classifications divided by the number of instances in the testset) to measure performance and we carry out two types of evaluation. On the one hand, we use 10-fold cross-validation on the EmoSpeech corpus (H-H data). On the other hand, we report accuracy on a corpus of 550 Human-Computer (H-C) dialogues obtained by having 22 subjects play the game against the QA character trained on the H-H corpus. As we shall see below, performance decreases in this second evaluation suggesting that subjects produce different turns when playing with a computer than with a human thereby inducing a weak out-of-domain effect and negatively impacting classification. Evaluation on the H-H corpus therefore gives a measure of how well the techniques explored help improving the dialog engine when used in a real life setting.

Correspondingly, we use two different tests for measuring statistical significance. In the H-H evaluation, significance is computed using the Wilcoxon signed rank test because data are dependent and are not assumed to be normally distributed. When building the testset we took care of not including paraphrases of utterances in the training partition (for each paraphrase generated automatically we keep track of the original utterance), however utterances in both datasets might be generated by the same subject, since a subject completed 12 distinct dialogues during the game. Conversely, in the H-C evaluation, training (H-H data) and test (H-C data) sets were collected under different conditions with different subjects therefore significance was computed using the McNemar sign-test (Dietterich, 1998).

4 Paraphrases, Synonyms and Lemmatisation

We explore three main ways of modifying the content features used for classification : lemmatising the training and the test data ; augmenting the training data with automatically acquired paraphrases ; and substituting unknown words with synonyms at run time.

Lemmatisation We use the French version of Treetagger [3] to lemmatise both the training and the test data. Lemmas without any filtering were used

1. We used MALLET (McCallum, 2002) for the LR classifier with L1 Regularisation.

2. TF*IDF = Term Frequency*Inverse Document Frequency

3. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

to train classifiers. We then compare performance with and without lemmatisation. As we shall see, the lemma and the POS tag provided by TreeTagger are also used to lookup synonym dictionaries and EuroWordNet when using synonym handling at run time.

Paraphrases : (DeVault et al., 2011) showed that enriching the training corpus with manually added paraphrases increases accuracy. Here we exploit automatically acquired paraphrases and use these not only to increase the size of the training corpus but also to better balance it [4]. We proceed as follows.

First, we generated paraphrases using a pivot machine translation approach where each user utterance in the training corpus (around 3610 utterances) was translated into some target language and back into French. Using six different languages (English, Spanish, Italian, German, Chinese and Arabian), we generated around 38000 paraphrases. We used Google Translate API for translating.

Category	Train Instances	Balanced Instances
greet	24	86
help	20	82
yes	92	123
no	55	117
ack	73	135
other	27	89
quit	38	100
find_plans	115	146
job	26	88
staff	15	77
studies	20	82
security_policies	24	86
μ	44.08	100.92
σ	$\pm$32.68	$\pm$23.32

TABLE 1: Skewed and Balanced Data on a sample subdialog. The category with lowest number of paraphrases is `greet`, with 62 paraphrases, hence $l_p = 62$. All categories were increased by 62 except `find_plans` and `yes` that were increased by half : 31.

Second, we eliminate from these paraphrases, words that are likely to be incorrect lexical translations by removing words with low normalized term frequency (< 0.001) across translations i.e., lexical translations given by few translations and/or translation systems. We then preprocessed the paraphrases in the same way the utterances of the initial training corpus were preprocessed i.e., utterances were unaccented, converted to lower-case and stop words were removed, the remaining words were filtered with TF*IDF. After preprocessing, duplicates were removed.

4. The Emospeech data is highly skewed with some classes being populated with many utterances and others with few.

```
Algorithm extendingDataWithParaphrases(trainingset ts)
1. Let c be the set of categories in ts.
2. μ be the mean of train instances per category
3. σ be the standard deviation of train instances per category
4. Let Np_c be the number of paraphrases per category
5. Let l_p ← min Np_cj
6. Repeat
7.   set i ← 0
8.   Ninst_ci be the number of instances per category c_i
9.   d_i ← Ninst_ci − μ
10.  if d_i < σ then
11.    Ninst_ci ← l_p
12.  else
13.    Ninst_ci ← l_p/2
14.  end if
15.  set i←i+1
16.  if i>‖c‖ then
17.   terminate
18. end
```

FIGURE 1: Algorithm for augmenting the training data with paraphrases.

Third, we added the paraphrases to the training data seeking to improve the balance between dialog moves per dialog, as shown in Figure 1. To this end, we look for the category c with the lowest number of paraphrases l_p (line 5). We then compute the deviation d_i for each dialog move c_i from the mean μ in the original training set (line 9). If the deviation d_i is lower than the standard deviation then we add l_p number of paraphrases instances (line 11). Conversely, if d_i is higher than the standard deviation, we reduce the number of instances to be added by half $\frac{l_p}{2}$ (line 13). Table 1 shows the original and the extended training data for the third sub-dialog in the Emospeech game. In this dialogue the player is supposed to ask information about the joystick plans ($find\_plans$, which is the mandatory goal). The categories cover mandatory and optional goals and general dialogue acts, such as greetings, asking for help, confirm and disconfirm, acknowledgment and out of topic questions (i.e. other).

Substituting Synonyms for Unknown Words A word is unknown, if it is a well-formed French

word [5] and if it does not appear in the training corpus. Conversely, a word is known if it is not unknown.

When an unknown word w is detected in a player utterance at runtime, we search for a word w' which occurs in the training data and is either a synonym of w or a distributional neighbour. After disambiguation, we substitute the unknown word for the synonym.

To identify synonyms, we make use of two lexical resources namely, the French version of EuroWordNet (EWN) (Vossen, 1998), which includes 92833 synonyms, hyperonyms and hyponyms pairs, and a synonym lexicon for French (DIC) [6] which contains 38505 lemmas and 254149 synonym pairs. While words are categorised into Noun, Verbs and Adjectives in EWN, DIC contains no POS tag information.

To identify distributional neighbours, we constructed semantic word spaces for each subdialog in the EmoSpeech corpus [7] using random indexing (RI) [8] on the training corpus expanded with paraphrases. Using the cosine measure as similarity metrics, we then retrieve for any unknown word w, the word w' which is most similar to w and which appear in the training corpus.

For lexical disambiguation, two methods are compared. We use the POS tag provided by TreeTagger. In this case, disambiguation is syntactic only. Or we pick the synonym with highest probability based on a trigram language model trained on the H-H corpus [9].

5 Results and Discussion

Table 2 summarises the results obtained in four main configurations : (i) with and without paraphrases ; (ii) with and without synonym handling ; (iii) with and without lemmatisation ; and (iv) when combining lemmatisation with synonym handling. We also compare the results obtained when evaluating using 10-fold cross validation on the training data (H-H dialogs) vs. evaluating the performance of the system on H-C interactions.

Overall Impact The largest performance gain is obtained by a combination of the three techniques explored in this paper namely, data expansion, synonym handling and lemmatisation (+8.9 points for the cross-validation experiment and +2.3 for the H-C evaluation).

Impact of Lexical Substitution at Run Time Because of space restrictions, we do not report here the results obtained using lexical resources without lemmatisation. However, we found that lexical resources are only useful when combined with lemmatisation. This is unsurprising since synonym dictionaries and EuroWordNet only contain lemmas. Indeed when distributional neighbours are used, lemmatisation has little impact (e.g., 65.11% using distributional neighbours without lemmatisation on the H-H corpus without paraphrases vs. 66.41% when using lemmatisation).

Another important issue when searching for a word synonym concerns lexical disambiguation : the synonym used to replace an unknown word should capture the meaning of that word in its given context. We tried using a language model trained on the training corpus to choose between synonym candidates (i.e., selecting the synonym yielding the highest sentence probability when substituting that synonym for the unknown word) but did not obtain a significant improvement. In contrast, it is noticeable that synonym handling has a higher impact when using EuroWordNet as a lexical resource. Since EuroWordNet contain categorial information while the synonym dictionaries we used do not, this suggests that the categorial disambiguation provided by TreeTagger helps identifying an appropriate synonym in EuroWordNet.

Finally, it is clear that the lexical resources used for this experiment are limited in coverage and quality. We observed in particular that some words which are very frequent in the training data (and thus which could be used to replace unknown words) do not occur in the synonym dictionaries. For instance when using paraphrases and dictionaries (fourth row and

5. A word is determined to be a well-formed French word if it occurs in the LEFFF dictionary, a large-scale morphological and syntactic lexicon for French (Sagot, 2010)

6. DICOSYN (http ://elsap1.unicaen.fr/dicosyn.html).

7. We also used distributional semantics from the Gigaword corpus but the results were poor probably because of the very different text genre and domains between the the Gigaword and the MP game.

8. Topics are Dialog acts while documents are utterances ; we used the S-Space Package `http://code.google.com/p/airhead-research/wiki/RandomIndexing`

9. We used SRILM (`http://www.speech.sri.com/projects/srilm`)

H		Lemmatisation			
H-H	Orig.	Lemmas	+EWN	+DIC	+RI
Orig.	$65.70\% \pm 5.62$	$\mathbf{66.04\% \pm 6.49}$	$68.17\% \pm 6.98$	$67.92\% \pm 4.51$	$66.83\% \pm 5.92$
Parap.	$70.89\% \pm 6.45$	$\mathbf{74.31\% \pm 4.78}$*	$\mathbf{74.60\% \pm 5.99}$*	$\mathbf{73.07\% \pm 7.71}$*	$\mathbf{72.63\% \pm 5.82}$*
H-C	Orig.	Lemmas	+EWN	+DIC	+RI
Orig.	$59.71\% \pm 16.42$	$59.88\% \pm 7.19$	$61.14\% \pm 16.65$	$61.41\% \pm 16.59$	$60.75\% \pm 17.39$
Parap.	$59.82\% \pm 15.53$	$59.48\% \pm 14.02$	$\mathbf{61.70\% \pm 14.09}$*	$\mathbf{62.01\% \pm 14.37}$*	$61.16\% \pm 14.41$*

TABLE 2: Accuracy on the H-H and on the H-C corpus. The star denotes statistical significance with the Wilcoxon test ($p < 0.005$) used for the HH corpus and the McNemar test ($p < 0.005$) for the HC corpus.

fourth column in Table 2) 50% of the unknown words were solved, 17% were illformed and 33% remained unsolved. To compensate this deficiency, we tried combining the three lexical resources in various ways (taking the union or combining them in a pipeline using the first resource that would yield a synonym). However the results did not improve and even in some cases worsened due probably to the insufficient lexical disambiguation. Interestingly, the results show that paraphrases always improves synonym handling presumably because it increases the size of the known vocabulary thereby increasing the possibility of finding a known synonym.

In sum, synonym handling helps most when (i) words are lemmatised and (ii) unknown words can be at least partially (i.e., using POS tag information) disambiguated. Moreover since data expansion increases the set of known words available as potential synonyms for unknown words, combining synonym handling with data expansion further improves accuracy.

Impact of Lemmatisation When evaluating using cross validation on the training corpus, lemmatisation increases accuracy by up to 3.42 points indicating that unseen word forms negatively impact accuracy. Noticeably however, lemmatisation has no significant impact when evaluating on the H-C corpus. This in turn suggests that the lower accuracy obtained on the H-C corpus results not from unseen word forms but from unseen lemmas.

Impact of Paraphrases On the H-H corpus, data expansion has no significant impact when used alone. However it yields an increase of up to 8.27 points and in fact, has a statistically significant impact, for all configurations involving lemmatisation. Thus, data expansion is best used in combination with lemmatisation and their combination permits creating better, more balanced and more general training data. On the H-C corpus however, the impact is negative or insignificant suggesting that the decrease in performance on the H-C corpus is due to content words that are new with respect to the training data i.e., content words for which neither a synonym nor a lemma can be found in the expanded training data.

Conclusion

While classifiers are routinely trained on dialog data to model the dialog management process, the impact of such basic factors as lemmatisation, automatic data expansion and synonym handling has remained largely unexplored. The empirical evaluation described here suggests that each of these factors can help improve performance but that the impact will vary depending on their combination and on the evaluation mode. Combining all three techniques yields the best results. We conjecture that there are two main reasons for this. First, synonym handling is best used in combination with POS tagging and lemmatisation because these supports partial lexical semantic disambiguation. Second, data expansion permits expanding the set of known words thereby increasing the possibility of finding a known synonym to replace an unknown word with.

Acknowledgments

This work was partially supported by the EU funded Eurostar EmoSpeech project. We thank Google for giving us access to the University Research Program of Google Translate.

References

David DeVault, Anton Leuski, and Kenji Sagae. 2011. Toward learning and evaluation of dialogue policies with text examples. In *12th SIGdial Workshop on Discourse and Dialogue*, Portland, OR, June.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10 :1895–1923.

Hilda Hardy, Tomek Strzalkowski, Min Wu, Cristian Ursu, Nick Webb, Alan W. Biermann, R. Bryce Inouye, and Ashley McKenzie. 2004. Data-driven strategies for an automated dialogue system. In *ACL*, pages 71–78.

Ian Richard Lane, Tatsuya Kawahara, and Shinichi Ueno. 2004. Example-based training of dialogue planning incorporating user and situation models. In *INTERSPEECH*.

Anton Leuski and David Traum. 2008. A statistical approach for text processing in virtual humans. In *Proceedings of the 26th Army Science Conference*.

Andrew Kachites McCallum. 2002. Mallet : A machine learning for language toolkit. http ://mallet.cs.umass.edu.

Lina Maria Rojas Barahona and Claire Gardent. 2012. What should I do now ? Supporting conversations in a serious game. In *SeineDial 2012 - 16th Workshop on the Semantics and Pragmatics of Dialogue*, Paris, France. Jonathan Ginzburg (chair), Anne Abeillé, Margot Colinet, Gregoire Winterstein.

Lina M. Rojas-Barahona, Alejandra Lorenzo, and Claire Gardent. 2012a. Building and exploiting a corpus of dialog interactions between french speaking virtual and human agents. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*.

Lina M. Rojas Barahona, Alejandra Lorenzo, and Claire Gardent. 2012b. An end-to-end evaluation of two situated dialog systems. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 10–19, Seoul, South Korea, July. Association for Computational Linguistics.

K. Sagae, G. Christian, D. DeVault, , and D.R. Traum. 2009. Towards natural language understanding of partial speech recognition results in dialogue systems. In *Proceedings of Human Language Technologies : The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Companion Volume : Short Papers*, pages 53–56.

Benoît Sagot. 2010. The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French. In *7th international conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.

Piek Vossen, editor. 1998. *EuroWordNet : a multilingual database with lexical semantic networks*. Kluwer Academic Publishers, Norwell, MA, USA.

Cascading Collective Classification for Bridging Anaphora Recognition Using a Rich Linguistic Feature Set

Yufang Hou[1], **Katja Markert**[2], **Michael Strube**[1]
[1] Heidelberg Institute for Theoretical Studies gGmbH, Heidelberg, Germany
(yufang.hou|michael.strube)@h-its.org
[2]School of Computing, University of Leeds, UK
scskm@leeds.ac.uk

Abstract

Recognizing bridging anaphora is difficult due to the wide variation within the phenomenon, the resulting lack of easily identifiable surface markers and their relative rarity. We develop linguistically motivated discourse structure, lexico-semantic and genericity detection features and integrate these into a cascaded minority preference algorithm that models bridging recognition as a subtask of learning fine-grained information status (IS). We substantially improve bridging recognition without impairing performance on other IS classes.

1 Introduction

In *bridging* or *associative anaphora* (Clark, 1975; Prince, 1981; Gundel et al., 1993), the antecedent and anaphor are not coreferent but are linked via a variety of contiguity relations.[1] In Example 1, the phrases *a resident*, *the stairs* and *the lobby* are bridging anaphors with the antecedent *One building*.[2]

(1) *One building* was upgraded to red status while people were taking things out, and **a resident** called up **the stairs** to his girlfriend, telling her to keep sending things down to **the lobby**.

Bridging is an important problem as it affects linguistic theory and applications alike. For example, without bridging resolution, entity coherence between the first and second coordinated clause in Example 1 cannot be established. This is a problem both for coherence theories such as Centering (Grosz et al., 1995) (where bridging is therefore incorporated as an indirect realization of previous entities) as well as applications relying on entity coherence modelling, such as readability assessment or sentence ordering (Barzilay and Lapata, 2008).

Full bridging resolution needs (i) recognition that a bridging anaphor is present and (ii) identification of the antecedent and contiguity relation. In recent work, these two tasks have been tackled separately, with bridging recognition handled as part of information status (IS) classification (Markert et al., 2012; Cahill and Riester, 2012; Rahman and Ng, 2012). Each mention in a text gets assigned one IS class that describes its accessibility to the reader at a given point in a text, bridging being one possible class. We stay within this framework.

Bridging recognition is a difficult task, so that we had to report very low results on this IS class in previous work (Markert et al., 2012). This is due to the phenomenon's variety, leading to a lack of clear surface features for recognition. Instead, we formulate in this paper novel discourse structure and lexico-semantic features as well as features that distinguish bridging from generics (see Section 3). In addition, making up between 5% and 20% of definite descriptions (Gardent and Manuélian, 2005; Caselli and Prodanof, 2006) and around 6% of all NPs (Markert et al., 2012), bridging is still less frequent than many other IS classes and recognition of minority classes is well known to be more difficult. We therefore use a cascaded classification algorithm to address this problem (Omuya et al., 2013).

[1]We exclude *comparative anaphora* where anaphor and antecedent are in a similarity/exclusion relation, indicated by anaphor modifiers such as *other* or *similar* (Modjeska et al., 2003).

[2]Examples are from OntoNotes (Weischedel et al., 2011). Bridging anaphora are set in boldface; antecedents in italics.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 814–820,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Most bridging research concentrates on antecedent selection only (Poesio and Vieira, 1998; Poesio et al., 2004a; Markert et al., 2003; Lassalle and Denis, 2011; Hou et al., 2013), assuming that bridging recognition has already been performed. Previous work on recognition is either limited to definite NPs based on heuristics evaluated on small datasets (Hahn et al., 1996; Vieira and Poesio, 2000), or models it as a subtask of learning fine-grained IS (Rahman and Ng, 2012; Markert et al., 2012; Cahill and Riester, 2012). Results within this latter framework for bridging have been mixed: We reported in Markert et al. (2012) low results for bridging in written news text whereas Rahman and Ng (2012) report high results for the four subcategories of bridging annotated in the Switchboard dialogue corpus by Nissim et al. (2004). We believe this discrepancy to be due to differences in corpus size and genre as well as in bridging definition. Bridging in Switchboard includes non-anaphoric, syntactically linked part-of and set-member relationships (such as *the building's lobby*), as well as comparative anaphora, the latter being marked by surface indicators such as *other, another etc*. Both types are much easier to identify than anaphoric bridging cases.[3] In addition, many non-anaphoric lexical cohesion cases have been annotated as bridging in Switchbard as well.

We also separate bridging recognition and antecedent selection. One could argue that a joint model is more attractive as potential antecedents such as *building* "trigger" subsequent bridging cases such as *stairs* (Example 1). However, bridging can be indicated by referential patterns without world knowledge about the anaphor/antecedent NPs, as the nonsense example 2 shows: *the wug* is clearly a bridging anaphor although we do not know the antecedent.[4]

(2) The blicket couldn't be connected to the dax. **The wug** failed.

Similarly, Clark (1975) distinguishes between bridging via necessary, probable and inducible parts/roles and argues that only in the first and maybe the second case the antecedent triggers the bridging anaphor in the sense that we already spontaneously think of the anaphor when we read the antecedent. Also, bridging recognition on its own can be valuable for applications: for example, prosody is influenced by IS status without needing antecedent knowledge (Baumann and Riester, 2013).

[3] See also the high results for our specific category for comparative anaphora (Markert et al., 2012).

[4] We thank an anonymous reviewer for pointing this out.

3 Characterizing Bridging Anaphora for Automatic Recognition

3.1 Properties of bridging anaphora

Bridging anaphors are rarely marked by surface features. Indeed, even the common practice (Vieira and Poesio, 2000; Lassalle and Denis, 2011; Cahill and Riester, 2012) to limit bridging to definite NPs does not seem to be correct: We report in previous work (Hou et al., 2013) that less than 40% of the bridging anaphora in our corpus are definites. Instead, bridging is diverse with regard to syntactic form and function: bridging anaphora can be definite NPs (Examples 4 and 6), indefinite NPs (Example 5) or bare NPs (Examples 3, 8 and 9). The only frequent syntactic property shared is that bridging NPs tend to have a simple internal structure with regards to modification. Bridging is also easily confused with generics: *friends* is used as bridging anaphor in Example 9 but generically in Example 10.

(3) . . . *meat* . . . The Communists froze **prices** instead.

(4) . . . *the fund's building* . . . **The budget** was only $400,000.

(5) . . . *employees* . . . **A food caterer** stashed stones in the false bottom of a milk pail.

(6) . . . *his truck* . . . The farmer at **the next truck** shouts, "Wheat!"

(7) . . . *the firms* . . . Crime was the reason that **26%** reported difficulty recruiting personnel and that **19%** said they were considering moving.

(8) . . . *the company* . . . His father was **chairman** and **chief executive** until his death in an accident five years ago.

(9) . . . *Josephine Baker* . . . **Friends** pitched in.

(10) Friends are part of the glue that holds life and faith together.

Bridging anaphora can have almost limitless variation. However, we observe that bridging anaphors are often licensed because of discourse structure

Markert et al. (2012) local feature set	
$f1$ FullPrevMention (b)	$f2$ FullPreMentionTime (n)
$f3$ PartialPreMention (b)	$f4$ ContentWordPreMention (b)
$f5$ Determiner (n)	$f6$ NPtype (n)
$f7$ NPlength (int)	$f8$ GrammaticalRole (n)
$f9$ NPNumber (n)	$f10$ PreModByCompMarker (b)
$f11$ SemanticClass (n)	
Markert et al. (2012) relational feature set	
$f12$ HasChild (r)	$f13$ Precedes (r)

Table 1: Markert et al.'s (2012) feature set, b indicates binary, n nominal, r relational features.

new local features for bridging	
discourse structure	$f1$ IsCoherenceGap (b)
	$f2$ IsSentFirstMention (b)
	$f3$ IsDocFirstMention (b)
semantics	$f4$ IsWordNetRelationalNoun (b)
	$f5$ IsInquirerRoleNoun (b)
	$f6$ IsBuildingPart (b)
	$f7$ IsSetElement (b)
	$f8$ PreModSpatialTemporal (b)
	$f9$ IsYear (b)
	$f10$ PreModifiedByCountry (b)
generic NP features	$f11$ AppearInIfClause (b)
	$f12$ VerbPosTag (l)
	$f13$ IsFrequentGenericNP (b)
	$f14$ WorldKnowledgeNP (l)
	$f15$ PreModByGeneralQuantifier (b)
other features	$f16$ Unigrams (l)
	$f17$ BridgingHeadNP (l)
	$f18$ HasChildNP (b)
new features for other mediated categories	
aggregate	$f19$ HasChildCoordination (r)
function	$f20$ DependOnChangeVerb (b)
worldKnowledge	$f21$ IsFrequentProperName (b)

Table 2: New feature set, l indicates lexical features.

and/or lexical or world knowledge. With regard to discourse structure, Grosz et al. (1995) observe that bridging is often needed to establish entity coherence between two adjacent sentences (Examples 1, 2, 4, 5, 6, 7 and 9). With regard to lexical and world knowledge, relational noun phrases (Examples 3, 4, 8 and 9), building parts (Example 1), set membership elements (Example 7), or, more rarely, temporal/spatial modification (Example 6) may favor a bridging reading. Motivated by these observations, we develop discourse structure and lexico-semantic features indicating bridging anaphora as well as features designed to separate genericity from bridging.

3.2 Features

In Markert et al. (2012) we classify eight fine-grained IS categories for NPs in written text: *old*, *new* and 6 *mediated* categories (*syntactic*, *worldKnowledge*, *bridging*, *comparative*, *aggregate* and *function*). This feature set (Table 1, $f1$-$f13$) works well to identify *old*, *new* and several *mediated* categories. However, it fails to recognize most bridging anaphora which we try to remedy in this work by including more diverse features.

Discourse structure features (Table 2, $f1$-$f3$). Bridging occurs frequently in sentences where otherwise there would no entity coherence to previous sentences/clauses (see Grosz et al. (1995) and Poesio et al. (2004b) for discussions about bridging, entity coherence and centering transitions in the Centering framework). This is especially true for *topic* NPs (Halliday and Hasan, 1976) in such sentences.

We follow these insights by identifying coherence gap sentences (see Examples 1, 4, 5, 6, 7, 9 and also 2): a sentence has a coherence gap ($f1$) if it has none of the following three coherence elements: (1) entity coreference to previous sentences, as approximated via string match or presence of pronouns, (2) comparative anaphora approximated by mentions modified via a small set of comparative markers (see also Table 1, $f10$ *PreModByCompMarker*), or (3) proper names. We approximate the topic of a sentence via the first mention ($f2$).

$f3$ models that bridging anaphors do not appear at the beginning of a text.

Semantic features (Table 2, $f4$-$f10$). In contrast to generic patterns, our semantic features capture lexical properties of nouns that make them more likely to be the head of a bridging NP. We create $f4$-$f8$ to capture four kinds of bridging anaphora.

Löbner (1985) distinguishes between relational nouns that take on at least one obligatory semantic role (such as *friend*) and sortal nouns. It is likely that relational nouns are more frequently used as bridging than sortal nouns (see Examples 3, 4, 8 and 9). We extract a list containing around 4,000 relational nouns from WordNet and a list containing around 500 nouns that specify professional roles from the General Inquirer lexicon (Stone et al., 1966), then determine whether the NP head appears in these lists

or not ($f4$ and $f5$). The obligatory semantic role for a relational noun can of course also be filled NP internally instead of anaphorically and we use the features $f10$ (for instances such as *the Egyptian president*) and $f18$ (for complex NPs that are likely to fill needed roles NP internally) to address this.

Because part-of relations are typical bridging relations (see Example 1 and Clark (1975)), we use $f6$ to determine whether the NP is a part of the building or not, using again a list extracted from Inquirer.

$f7$ is used to identify set membership bridging cases (see Example 7), by checking whether the NP head is a number or indefinite pronoun (such as *none*, *one*, *some*) or modified by *each, one*. However, not all numbers are bridging cases (such as *1976*) and we use $f9$ to exclude such cases.

Lassalle and Denis (2011) note that some bridging anaphors are indicated by spatial or temporal modifications (see Example 6). We use $f8$ to detect this by compiling 20 such adjectives from Inquirer.

Features to detect generic nouns (Table 2, $f11$-$f15$). Generic NPs (Example 10) are easily confused with bridging anaphora. Inspired by Reiter and Frank (2010) who build on linguistic research, we develop features ($f11$-$f15$) to exclude generics.

First, hypothetical entities are likely to refer to generic entities (Mitchell et al., 2002), We approximate this by determining whether the NP appears in an if-clause ($f11$). Also the clause tense and mood may play a role to decide genericity (Reiter and Frank, 2010). This is often reflected by the main verb of a clause, so we extract its POS tag ($f12$).

Some NPs are commonly used generically, such as *children*, *men*, or *the dollar*. The ACE-2 corpus (distinct from our corpus) contains generic annotation . We collect all NPs from ACE-2 that are always used generically ($f13$). We also try to learn NPs that are uniquely identifiable without further description or anaphoric links such as *the sun* or *the pope*. We do this by extracting common nouns which are annotated as *worldKnowledge* from the training part of our corpus[5] and use these as lexical features ($f14$).

Finally, motivated by the ACE-2 annotation guidelines, we identify six quantifiers that may indicate genericity, such as *all*, *no*, *neither* ($f15$).

[5]This list varies for each run of our algorithm in 10-fold cross validation.

Other features for bridging (Table 2, $f16$-$f18$). Following Rahman and Ng (2012), we use unigrams ($f16$). We also extract heads of bridging anaphors from the training data as lexical features ($f17$) to learn typical nouns used for bridging that we did not cover in lexicon extraction ($f4$ to $f6$).

Feature $f18$ models that bridging anaphora most often have a simple internal structure and usually do not contain any other NPs.

Features for other IS categories (Table 2, $f19$-$f21$). We propose three features to improve other IS categories. In the relational feature $f19$, we separate coordination parent-child from other parent-child relations to help with the class *aggregate*. $f20$ determines whether a number is the object of an increase/decrease verb (using a list extracted from Inquirer) and therefore likely to be the IS class *function*. Frequent proper names are more likely to be hearer old and hence of the class *worldKnowledge*. $f21$ extracts proper names that occur in at least 100 documents in the Tipster corpus to approximate this.

4 Experiments and Results

Experimental setup. We perform experiments on the corpus provided in Markert et al. (2012)[6]. It consists of 50 texts taken from the WSJ portion of the OntoNotes corpus (Weischedel et al., 2011) with almost 11,000 NPs annotated for information status including 663 bridging NPs and their antecedents. All experiments are performed via 10-fold cross-validation on documents. We use gold standard mentions and the OntoNotes named entity and syntactic annotation layers for feature extraction.

Reimplemented baseline system (*rbls*). *rbls* uses the same features as Markert et al. (2012) (Table 1) but replaces the local decision tree classifier with LibSVM as we will need to include lexical features.

rbls + Table 2 feature set (*rbls*+*newfeat*). Based on *rbls*, all the new features from Table 2 are added.

Cascading minority preference system (*cmps*). Minority classes such as bridging suffer during standard multi-class classification. Inspired by Omuya

[6]http://www.h-its.org/nlp/download/isnotes.php

	collective									***cascade + collective***					
	markert 12			*rbls*			*rbls+newfeat*			*cmps*			*cmps−newfeat*		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
old	84.1	85.2	84.6	84.6	85.5	85.1	84.4	86.0	85.2	82.2	87.2	84.7	78.9	89.5	83.8
med/worldKnowledge	60.6	70.0	65.0	65.9	69.6	67.7	67.4	77.3	**72.0**	67.2	77.2	**71.9**	67.5	66.7	67.1
med/syntactic	75.7	80.1	77.9	77.8	81.2	79.4	82.2	81.9	82.0	81.6	82.5	82.0	73.9	81.7	77.6
med/aggregate	43.1	55.8	48.7	47.9	58.0	52.5	64.5	79.5	**71.2**	63.5	77.9	**70.0**	46.9	60.0	52.7
med/function	35.4	53.5	48.7	33.8	56.4	42.3	67.7	72.1	**69.8**	67.7	72.1	**69.8**	41.5	50.0	45.4
med/comparative	81.4	82.0	81.7	81.8	82.5	82.1	81.8	82.1	82.0	86.6	78.2	82.2	86.2	78.7	82.3
med/bridging	12.2	41.7	18.9	10.7	36.6	16.6	19.3	39.0	25.8	44.9	39.8	**42.2**	31.8	23.9	27.3
new	87.7	73.3	79.8	87.5	74.8	80.7	86.5	76.1	81.0	83.0	78.1	80.5	82.4	76.1	79.1
acc		76.8			77.6			78.9			78.6			75.0	

Table 3: Experimental results

et al. (2013), we develop a cascading minority preference system for fine-grained IS classification. For the five minority classes (*function*, *aggregate*, *comparative*, *bridging* and *worldKnowledge*) that each make up less than the expected $\frac{1}{8}$ of the data set, we develop five binary classifiers with LibSVM[7] using all features from Tables 1 and 2 and apply them in order from rarest to more frequent category. Whenever a minority classifier predicts true, this class is assigned. When all minority classifiers say false, we back off to the multiclass *rbls+newfeat* system.

cmps − Table 2 feature set (*cmps−newfeat*). To test the effect of using the minority preference system without additional features, we employ a *cmps* system with baseline features from Table 1 only.

Results and Discussion (Table 3). Our novel features in *rbls+newfeat* show improvements for *worldKnowledge*, *aggregate* and *function* as well as *bridging* categories compared to both baseline systems, although the performance for *bridging* is still low. In addition, the overall accuracy is significantly better than the two baseline systems (at the level of 1% using McNemar's test). Using the cascaded minority preference system *cmps* in addition improves bridging results substantially while the performance on other categories does not worsen. The algorithm needs both our novel feature classes as well as cascaded modelling to achieve this improvement as the comparison to *cmps−newfeat* shows: the latter lowers overall accuracy as it tends to overgenerate rare classes (including bridging) with low precision if the features are not strong enough. Our novel features (addressing linguistic properties of bridging) and the cascaded algorithm (addressing data sparseness) appear to be complementary.

To look at the impact of features in our best system, we performed an ablation study. Lexical features as well as semantic ones have the most impact. Discourse structure and genericity information features have less of an impact. We believe the latter to be due to noise involved in extracting these features (such as approximating coreference for the coherence gap feature) as well as genericity recognition still being in its infancy (Reiter and Frank, 2010).

5 Conclusions

This paper aims to recognize bridging anaphora in written text. We develop *discourse structure*, *lexico-semantic* and *genericity* features based on linguistic intuition and corpus research. By using a cascading minority preference system, we show that our approach outperforms the bridging recognition in Markert et al. (2012) by a large margin without impairing the performance on other IS classes.

Acknowledgements. Yufang Hou is funded by a PhD scholarship from the Research Training Group *Coherence in Language Processing* at Heidelberg University. Katja Markert receives a Fellowship for Experienced Researchers by the Alexander-von-Humboldt Foundation. We thank HITS gGmbH for hosting Katja Markert and funding the annotation.

[7]Parameter against data imbalance is set according to the ratio between positive and negative instances in the training set.

References

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Stefan Baumann and Arndt Riester. 2013. Coreference, lexical givenness and prosody in German. *Lingua*. Accepted.

Aoife Cahill and Arndt Riester. 2012. Automatically acquiring fine-grained information status distinctions in German. In *Proceedings of the SIGdial 2012 Conference: The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Seoul, Korea, 5–6 July 2012, pages 232–236.

Tommaso Caselli and Irina Prodanof. 2006. Annotating bridging anaphors in Italian: In search of reliability. In *Proceedings of the 5th International Conference on Language Resources and Evaluation,* Genoa, Italy, 22–28 May 2006.

Herbert H. Clark. 1975. Bridging. In *Proceedings of the Conference on Theoretical Issues in Natural Language Processing,* Cambridge, Mass., June 1975, pages 169–174.

Claire Gardent and Hélène Manuélian. 2005. Création d'un corpus annoté pour le traitement des descriptions définies. *Traitement Automatique des Langues*, 46(1):115–140.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Jeanette K. Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

Udo Hahn, Michael Strube, and Katja Markert. 1996. Bridging textual ellipses. In *Proceedings of the 16th International Conference on Computational Linguistics,* Copenhagen, Denmark, 5–9 August 1996, volume 1, pages 496–501.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. London, U.K.: Longman.

Yufang Hou, Katja Markert, and Michael Strube. 2013. Global inference for bridging anaphora resolution. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 907–917.

Emmanuel Lassalle and Pascal Denis. 2011. Leveraging different meronym discovery methods for bridging resolution in French. In *Proceedings of the 8th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2011),* Faro, Algarve, Portugal, 6–7 October 2011, pages 35–46.

Sebastian Löbner. 1985. Definites. *Journal of Semantics*, 4:279–326.

Katja Markert, Malvina Nissim, and Natalia N. Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proceedings of the EACL Workshop on the Computational Treatment of Anaphora.* Budapest, Hungary, 14 April 2003, pages 39–46.

Katja Markert, Yufang Hou, and Michael Strube. 2012. Collective classification for fine-grained information status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics,* Jeju Island, Korea, 8–14 July 2012, pages 795–804.

Alexis Mitchell, Stephanie Strassel, Mark Przybocki, JK Davis, George Doddington, Ralph Grishman, Adam Meyers, Ada Brunstain, Lisa Ferro, and Beth Sundheim. 2002. ACE-2 Version 1.0. LDC2003T11, Philadelphia, Penn.: Linguistic Data Consortium.

Natalia M. Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing,* Sapporo, Japan, 11–12 July 2003, pages 176–183.

Malvina Nissim, Shipara Dingare, Jean Carletta, and Mark Steedman. 2004. An annotation scheme for information status in dialogue. In *Proceedings of the 4th International Conference on Language Resources and Evaluation,* Lisbon, Portugal, 26–28 May 2004, pages 1023–1026.

Malvina Nissim. 2006. Learning information status of discourse entities. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing,* Sydney, Australia, 22–23 July 2006, pages 94–012.

Adinoyi Omuya, Vinodkumar Prabhakaran, and Owen Rambow. 2013. Improving the quality of minority class identification in dialog act tagging. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Atlanta, Georgia, 9–14 June 2013, pages 802–807.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004a. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pages 143–150.

Massimo Poesio, Rosemary Stevenson, Barbara Di Eugenio, and Janet Hitzeman. 2004b. Centering: A parametric theory and its instantiations. *Computational Linguistics*, 30(3). 309-363.

Ellen F. Prince. 1981. Towards a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*, pages 223–255. Academic Press, New York, N.Y.

Altaf Rahman and Vincent Ng. 2012. Learning the fine-grained information status of discourse entities. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics,* Avignon, France, 23–27 April 2012, pages 798–807.

Nils Reiter and Anette Frank. 2010. Identifying generic noun phrases. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 40–49.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie, and Cambridge Computer Associates. 1966. *General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, Mass.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. OntoNotes release 4.0. LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium.

A synchronous context free grammar for time normalization

Steven Bethard
University of Alabama at Birmingham
Birmingham, Alabama, USA
bethard@cis.uab.edu

Abstract

We present an approach to time normalization (e.g. *the day before yesterday*⇒2013-04-12) based on a synchronous context free grammar. Synchronous rules map the source language to formally defined operators for manipulating times (FINDENCLOSED, STARTATENDOF, etc.). Time expressions are then parsed using an extended CYK+ algorithm, and converted to a normalized form by applying the operators recursively. For evaluation, a small set of synchronous rules for English time expressions were developed. Our model outperforms HeidelTime, the best time normalization system in TempEval 2013, on four different time normalization corpora.

1 Introduction

Time normalization is the task of converting a natural language expression of time into a formal representation of a time on a timeline. For example, the expression *the day before yesterday* would be normalized to the formal representation 2013-04-12 (assuming that today is 2013-04-14) in the ISO-TimeML representation language (Pustejovsky et al., 2010). Time normalization is a crucial part of almost any information extraction task that needs to place entities or events along a timeline. And research into methods for time normalization has been growing since the ACE[1] and TempEval (Verhagen et al., 2010; UzZaman et al., 2013) challenges began to include time normalization as a shared task.

[1] http://www.itl.nist.gov/iad/mig/tests/ace/

Most prior work on time normalization has taken a rule-based, string-to-string translation approach. That is, each word in a time expression is looked up in a normalization lexicon, and then rules map this sequence of lexical entries directly to the normalized form. HeidelTime (Strötgen and Gertz, 2012), which had the highest performance in TempEval 2010 and 2013, and TIMEN (Llorens et al., 2012), which reported slightly higher performance in its own experiments, both follow this approach. A drawback of this approach though is that there is no nesting of rules: for example, in HeidelTime the rules for *yesterday* and *the day before yesterday* are completely separate, despite the compositional nature of the latter.

A notable exception to the string-to-string approach is the work of (Angeli et al., 2012). They define a target grammar of typed pre-terminals, such as YESTERDAY (a SEQUENCE) or DAY (a DURATION), and compositional operations, such as SHIFTLEFT (a (RANGE, DURATION) → RANGE). They apply an expectation-maximization approach to learn how words align to elements of the target grammar, and achieve performance close to that of the rule-based systems. However, their grammar does not allow for non-binary or partially lexicalized rules (e.g. SEQUENCE → DURATION *before* SEQUENCE would be impossible), and some of their primitive elements could naturally be expressed using other primitives (e.g. YESTERDAY as SHIFTLEFT(TODAY, 1 DAY)).

We present a synchronous grammar for time normalization that addresses these shortcomings. We first define a grammar of formal operations over temporal elements. We then develop synchronous rules that map time expression words to temporal opera-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 821–826,
Seattle, Washington, USA, 18-21 October 2013.

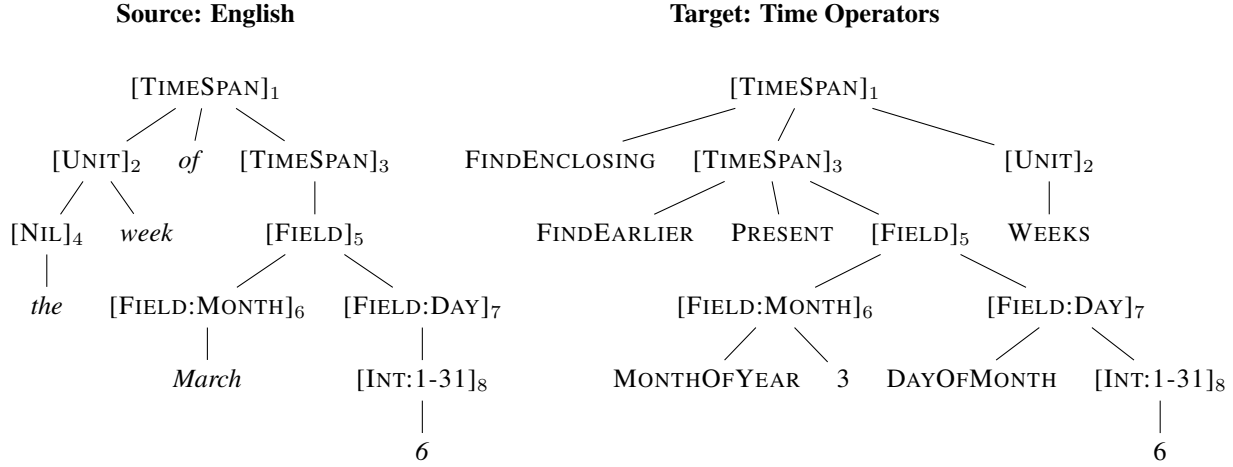


Figure 1: The synchronous parse from the source language *the week of March 6* to the target formal time representation FINDENCLOSING(FINDEARLIER(PRESENT, MONTHOFYEAR→3, DAYOFMONTH→6), WEEKS). Subscripts on non-terminals indicate the alignment between the source and target parses.

tors, and perform normalization by parsing with an extended CYK+ parsing algorithm. We evaluate this approach to time normalization on the TimeBank, AQUAINT, Timen and TempEval 2013 corpora.

2 Synchronous grammars

Our time grammar is based on the synchronous context free grammar formalism. Synchronous grammars allow two trees, one in the source language and one in the target language, to be constructed simultaneously. A synchronous context free grammar has rules of the form $X \rightarrow (\mathbf{S}, \mathbf{T}, A)$, where X is a non-terminal, $\mathbf{S}$ is the sequence of terminals and non-terminals that X expands to in the source language, $\mathbf{T}$ is the sequence of terminals and non-terminals that X expands to in the target language, and A is the alignment between the non-terminals of $\mathbf{S}$ and $\mathbf{T}$ (which must be the same).

For time normalization, the source side is the natural language text, and the target side is a formal grammar of temporal operators. Figure 1 shows a synchronous parse of *the week of March 6* [2]. The left side is the source side (an English expression), the right side is the target side (a temporal operator expression), and the alignment is shown via subscripts.

[2]Figure 1 corresponds to an interpretation along the lines of *the week of the last March 6*. The full grammar developed in this article would also produce an interpretation corresponding to *the week of the next March 6*, since the phrase is ambiguous.

3 Target time grammar

The right side of Figure 1 shows an example of our target formal representation: FINDENCLOSING(FINDEARLIER(PRESENT, MONTHOFYEAR→3, DAYOFMONTH→6), WEEKS). Each terminal in the parse is either a numeric value or an operator like FINDENCLOSING, WEEKS or MONTHOFYEAR. Each non-terminal combines terminals or non-terminals to create a [TIMESPAN], [PERIOD], [FIELD], [UNIT] or [INT]. The list of rules allowed by our target grammar (the right-hand side of our synchronous grammar) is given in Table 1.

Each of the target operators defines a procedure for creating a temporal object from others. For example, FINDENCLOSING takes a [TIMESPAN] and a [UNIT] and expands the start and end of the time span to fill a period of one unit. This could be used, for example, to define *today* as FINDENCLOSING(PRESENT, DAYS), where the PRESENT, which is instantaneous, is expanded out to the enclosing day. Note that we define things like *today* and *yesterday* in terms of primitive operations, rather than making them primitives themselves as in (Angeli et al., 2012).

The left side of Figure 1 shows the synchronous parse of the source language. Note that each of the non-terminals is aligned (shown as a subscript) with a non-terminal in the target parse[3], while terminals are not aligned and may freely appear or disappear

[3]We actually allow a slightly asynchronous grammar, where a non-terminal may be used 0 or more times on the target side.

[INT]	→	integer
[UNIT]	→	unit
[FIELD]	→	field [INT]
[FIELD]	→	[FIELD]*
[PERIOD]	→	SIMPLE [INT] [UNIT]
[PERIOD]	→	FRACTIONAL [INT] [INT] [UNIT]
[PERIOD]	→	UNSPECIFIED [UNIT]
[PERIOD]	→	WITHMODIFIER [PERIOD] modifier
[TIMESPAN]	→	PAST
[TIMESPAN]	→	PRESENT
[TIMESPAN]	→	FUTURE
[TIMESPAN]	→	FINDEARLIER [TIMESPAN] [FIELD]
[TIMESPAN]	→	FINDLATER [TIMESPAN] [FIELD]
[TIMESPAN]	→	FINDENCLOSING [TIMESPAN] [UNIT]
[TIMESPAN]	→	FINDENCLOSED [TIMESPAN] [FIELD]
[TIMESPAN]	→	STARTATENDOF [TIMESPAN] [PERIOD]
[TIMESPAN]	→	ENDATSTARTOF [TIMESPAN] [PERIOD]
[TIMESPAN]	→	MOVEEARLIER [TIMESPAN] [PERIOD]
[TIMESPAN]	→	MOVELATER [TIMESPAN] [PERIOD]
[TIMESPAN]	→	WITHMODIFIER [TIMESPAN] modifier

Table 1: Rules allowed by the target time grammar. A "unit" is any `java.time.temporal.TemporalUnit`, e.g. SECONDS, WEEKS or DECADES. A "field" is any `java.time.temporal.TemporalField`, e.g. HOUROFAMPM, DAYOFMONTH or CENTURY. A "modifier" is any of the TIMEX3 "mod" values defined in TimeML.

from the source to the target. Each non-terminal thus corresponds to a synchronous grammar rule that describes how a source expression should be translated into the target time grammar. For example the root nodes correspond to an application of the following full synchronous rule:

[TIMESPAN] →
source: [UNIT] *of* [TIMESPAN]
target: FINDENCLOSING [TIMESPAN] [UNIT]

4 Parsing algorithm

Parsing with a synchronous context free grammar is much the same as parsing with just the source side of the grammar. Only a small amount of bookkeeping is necessary to allow the generation of the target parse once the source parse is complete. We can therefore apply standard parsing algorithms to this task.

However, we have some additional grammar requirements. As shown in Figure 1, we allow rules that expand into more than two terminals or non-terminals, the mixing of terminals and non-terminals in a production, a special [NIL] non-terminal for the ignoring of words, and a special [INT] non-terminal that can match ranges of integers and does not require all possible integers to be manually listed in the grammar. This means that we can't directly use CYK parsing or even CYK+ parsing (Chappelier and Rajman, 1998), which allows rules that expand into more than two terminals or non-terminals, but does not meet our other requirements.

Algorithm 1 shows our extended version of CYK+ parsing. As with standard CYK+ parsing, two charts are filled, one for rules that have been completed (C) and one for rules that have been only partially advanced (P). All parses covering 1 terminal are completed first, then these are used to complete parses covering 2 terminals, etc. until all parses covering all terminals are complete.

Our extensions to the standard CYK+ parsing are as follows. To handle integers, we modify the initialization to generate new rules on the fly for any numeric terminals that fit the range of an [INT:X-Y] non-terminal in the grammar (starts at line 5). To allow mixing of terminals and non-terminals, we extend the initialization step to also produce partial parses (line 17), and extend the parse advancement step to allow advancing rules with terminals (starting at line 23). Finally, to handle [NIL] rules, which consume tokens but are not included in the final parse, we add a step where rules are allowed to advance, unchanged, past a [NIL] rule (starting at line 35).

5 Parsing example

As an example, consider parsing *the week of March 6* with the following source side grammar:

[NIL] → *the*
[UNIT] → *week*
[MONTH] → *March*
[DAY] → [INT:1-31]
[FIELD] → [MONTH][DAY]
[TIMESPAN] → [FIELD]
[TIMESPAN] → [UNIT] *of* [TIMESPAN]

First the algorithm handles the numeric special case, completing an [INT] parse for the token *6* at index 4:

$C_{(1,4)} \cup= [\text{INT:1-31}] \rightarrow 6$

Then it completes parses based on just the terminals:

$C_{(1,0)} \cup= [\text{NIL}] \rightarrow \textit{the}$
$C_{(1,1)} \cup= [\text{UNIT}] \rightarrow \textit{week}$
$C_{(1,3)} \cup= [\text{MONTH}] \rightarrow \textit{March}$

Next, the algorithm starts working on parses that span 1 token. It can start two partial parses, using the [UNIT] at $C_{(1,1)}$, and using the [MONTH] at $C_{(1,3)}$:

$P_{(1,1)} \cup= [\text{TIMESPAN}] \rightarrow [\text{UNIT}] \bullet \textit{of}\ [\text{TIMESPAN}]$
$P_{(1,3)} \cup= [\text{FIELD}] \rightarrow [\text{MONTH}] \bullet [\text{DAY}]$

Algorithm 1 CYK+ parsing, extended for partially lexicalized rules, [Nil] rules and numbers

Require: G a set of rules, w a sequence of tokens

function PARSE(G, w)
  $C \leftarrow$ a new $|w|+1$ by $|w|$ matrix
  $P \leftarrow$ a new $|w|+1$ by $|w|$ matrix
  // Generate rules on the fly for numeric tokens
  for $i \leftarrow 0 \ldots (|w|-1)$ **do**
    if ISNUMBER(w_i) **then**
      for all [INT:x-y] $\in$ non-terminals of G **do**
        if $x \leq$ TONUMBER(w_i) $\leq y$ **then**
          $C_{(1,i)} \cup\!= [\text{INT:}x\text{-}y] \rightarrow w_i$
  // Start any rules that begin with terminals
  for $i \leftarrow 0 \ldots (|w|-1)$ **do**
    for all $\text{X} \rightarrow \alpha\beta \in G$ **do**
      if $\exists j \mid \alpha = w_{i:j} \wedge \neg$ISTERMINAL($\beta_0$) **then**
        if $\beta = \epsilon$ **then**
          $C_{(|w_{i:j}|,i)} \cup\!= \text{X} \rightarrow w_{i:j}\alpha$
        else
          $P_{(|w_{i:j}|,i)} \cup\!= (|w_{i:j}|, \text{X} \rightarrow w_{i:j}\alpha)$
  for $n \leftarrow 1 \ldots |w|; i \leftarrow 0 \ldots (|w|-n)$ **do**
    // Find all parses of size n starting at i
    for $m \leftarrow 1 \ldots n$ **do**
      for all $(p, \text{X} \rightarrow \alpha) \in P_{(m,i)}$ **do**
        // Advance partial parses using terminals
        if $w_{i+m:i+n} = \alpha_{p:p+n-m}$ **then**
          if $\alpha_{p+n-m:|\alpha|} = \epsilon$ **then**
            $C_{(n,i)} \cup\!= \text{X} \rightarrow \alpha$
          else
            $P_{(n,i)} \cup\!= (p+n-m, \text{X} \rightarrow \alpha)$
        // Advance partial parses using completes
        for all $\alpha_p \rightarrow \beta \in C_{(n-m,i+m)}$ **do**
          if $|\alpha| = p+1$ **then**
            $C_{(n,i)} \cup\!= \text{X} \rightarrow \alpha$
          else
            $P_{(n,i)} \cup\!= (p+1, \text{X} \rightarrow \alpha)$
      // Advance complete parses past [Nil] parses
      for all $\text{X} \rightarrow \alpha \in C_{(m,i)}$ **do**
        for all $\text{Y} \rightarrow \beta \in C_{(n-m,i+m)}$ **do**
          if $\text{X} \neq$ Nil $\wedge$ Y $=$ Nil **then**
            $C_{(n,i)} \cup\!= \text{X} \rightarrow \alpha$
          else if X $=$ Nil $\wedge$ $\text{Y} \neq$ Nil **then**
            $C_{(n,i)} \cup\!= \text{Y} \rightarrow \beta$
    // Start any rules that begin with a complete parse
    for all $\text{X} \rightarrow \alpha \in C_{(n,i)}$ **do**
      for all $\text{Y} \rightarrow \text{X}\alpha \in C_{(n,i)}$ **do**
        if $\alpha = \epsilon$ **then**
          $C_{(n,i)} \cup\!= \text{Y} \rightarrow \text{X}\alpha$
        else
          $P_{(n,i)} \cup\!= (1, \text{Y} \rightarrow \text{X}\alpha)$
  return $C_{(|w|,0)}$

(The • is the visual equivalent of the first element in the partial parse tuples of Algorithm 1, which marks parsing progress.) And given the [INT:1-31] at $C_{(1,4)}$ the algorithm can make a complete size 1 parse:

$C_{(1,4)} \cup\!=$ [DAY] $\rightarrow$ [INT:1-31]

The algorithm then moves on to create parses that span 2 tokens. The special handling of [NIL] allows the [UNIT] at $C_{(1,1)}$ to absorb the [NIL] at $C_{(1,0)}$:

$C_{(2,0)} \cup\!=$ [UNIT] $\rightarrow$ *week*

This [UNIT] then allows the start of a partial parse:

$P_{(2,0)} \cup\!=$ [TIMESPAN] $\rightarrow$ [UNIT] • *of* [TIMESPAN]

The partial parse at $P_{(1,1)}$ can be advanced using *of* at position 2, creating another 2 token partial parse:

$P_{(2,1)} \cup\!=$ [TIMESPAN] $\rightarrow$ [UNIT] *of* • [TIMESPAN])

The partial parse at $P_{(1,3)}$ can be advanced using the [DAY] at $C_{(1,4)}$, completing the 2 token parse:

$C_{(2,3)} \cup\!=$ [FIELD] $\rightarrow$ [MONTH][DAY]

This [FIELD] allows completion of a 2 token parse:

$C_{(2,3)} \cup\!=$ [TIMESPAN] $\rightarrow$ [FIELD]

The algorithm then moves on to 3 token parses. Only one is possible: the partial parse at $P_{(2,0)}$ can be advanced using the *of* at position 2, yielding:

$P_{(3,0)} \cup\!=$ [TIMESPAN] $\rightarrow$ [UNIT] *of* • [TIMESPAN]

The algorithm moves on to 4 token parses, finding that the partial parse at $P_{(2,1)}$ can be advanced using the [TIMESPAN] at $C_{(2,3)}$, completing the parse:

$C_{(4,1)} \cup\!=$ [TIMESPAN] $\rightarrow$ [UNIT] *of* [TIMESPAN]

Finally, the algorithm moves on to 5 token parses, where (1) the special handling of [NIL] allows the partial parse at $C_{(4,1)}$ to consume the [NIL] at $C_{(1,0)}$ and (2) the partial parse at $P_{(3,0)}$ can be advanced using the [TIMESPAN] at $C_{(2,3)}$. Both of these yield:

$C_{(5,0)} \cup\!=$ [TIMESPAN] $\rightarrow$ [UNIT] *of* [TIMESPAN]

The complete parses in $C_{(5,0)}$ are then deterministically translated into target side parses using the alignments in the rules of the synchronous grammar.

6 Evaluation

Using our synchronous grammar formalism for time normalization, we manually developed a grammar for English time expressions. Following the lead of TIMEN and HeidelTime, we developed our grammar by inspecting examples from the AQUAINT[4] and

[4] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2002T31

	N	TIMEN	HeidelTime	SCFG
AQUAINT	652	69.5	74.7	**76.5**
TimeBank	1426	67.7	80.9	**84.9**
Timen	214	**67.8**	49.1	56.5
TempEval2013	158	74.1	78.5	**81.6**

Table 2: Performance of TIMEN, HeidelTime and our synchronous context free grammar (SCFG) on each evaluation corpus. (N is the number of time expressions.)

TimeBank (Pustejovsky et al., 2003) corpora. The resulting grammar has 354 rules, 192 of which are only lexical, e.g., [UNIT] → (*seconds*, SECONDS).

Our grammar produces multiple parses when the input is ambiguous. For example, the expression *Monday* could mean either the previous Monday or the following Monday, and the expression *the day* could refer either to a period of one day, or to a specific day in time, e.g. 2013-04-14. For such expressions, our grammar produces both parses. To choose between the two, we employ a very simple set of heuristics: (1) prefer [TIMESPAN] to [PERIOD], (2) prefer an earlier [TIMESPAN] to a later one and (3) prefer a [TIMESPAN] with QUARTERS granularity if the anchor time is also in QUARTERS (this is a common rule in TimeBank annotations).

We evaluate on the AQUAINT corpus, the TimeBank corpus, the Timen corpus (Llorens et al., 2012) and the TempEval 2013 test set (UzZaman et al., 2013)[5]. We compare to two[6] state-of-the-art systems: TIMEN and HeidelTime. Table 2 shows the results. Our synchronous grammar approach outperformed HeidelTime on all corpora, both on the training corpora (AQUAINT and TimeBank) and on the test corpora (Timen and TempEval 2013). Both our model and HeidelTime outperformed TIMEN on all corpora except for the Timen corpus.

To better understand the issues in the Timen corpus, we manually inspected the 33 time expressions that TIMEN normalized correctly and our approach normalized incorrectly. 4 errors were places where our heuristic was wrong (e.g. we chose the earlier, not the later *Sept. 22*). 6 errors were coverage problems of our grammar, e.g. not handling *season*, *every time* or *long ago*. 2 errors were actually human annotation errors (*several years ago* was annotated as PASTREF and *daily* was annotated as XXXX-XX-XX, while the guidelines say these should be PXY and P1D respectively). The remaining 21 errors were from two new normalization forms not present at all in the training data: 19 instances of THH:MM:SS (times were always YYYY-MM-DDTHH:MM:SS in the training data) and 2 instances of BCYYYY (years were always YYYY in the training data).

7 Discussion

Our synchronous grammar approach to time normalization, which handles recursive structures better than existing string-to-string approaches and handles a wider variety of grammars than existing parsing approaches, outperforms the HeidelTime system on four evaluation corpora and outperforms the TIMEN system on three of the four corpora.

Our time normalization code and models are freely available. The source code and English grammar are hosted at `https://github.com/bethard/timenorm`, and official releases are published to Maven Central (group=`info.bethard`, artifact=`timenorm`).

In future work, we plan to replace the heuristic for selecting between ambiguous parses with a more principled approach. It would be a simple extension to support a probabilistic grammar, as in (Angeli et al., 2012). But given an expression like *Monday*, it would still be impossible to decide whether it refers to the future or the past, since the surrounding context, e.g. tense of the governing verb, is needed for such a judgment. A more promising approach would be to train a classifier that selects between the ambiguous parses based on features of the surrounding context.

Acknowledgements

The project described was supported in part by Grant Number R01LM010090 from the National Library Of Medicine. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Library of Medicine or the National Institutes of Health.

[5]We evaluate normalization accuracy over all time expressions, not the F1 of both finding and normalizing expressions, so the numbers here are not directly comparable to those reported by the TempEval 2013 evaluation.

[6]Though its performance was slightly lower than HeidelTime, we also intended to compare to the (Angeli et al., 2012) system. Its authors graciously helped us get the code running, but to date all models we were able to train performed substantially worse than their reported results, so we do not compare to them here.

References

[Angeli et al.2012] Gabor Angeli, Christopher Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 446–455, Montréal, Canada, June. Association for Computational Linguistics.

[Chappelier and Rajman1998] Jean-Cédric Chappelier and Martin Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *First Workshop on Tabulation in Parsing and Deduction (TAPD98)*, pages 133–137. Citeseer.

[Llorens et al.2012] Hector Llorens, Leon Derczynski, Robert Gaizauskas, and Estela Saquete. 2012. TIMEN: An open temporal expression normalisation resource. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May. European Language Resources Association (ELRA).

[Pustejovsky et al.2003] James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK corpus. In *Corpus Linguistics*, pages 647–656, Lancaster, UK.

[Pustejovsky et al.2010] James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An international standard for semantic annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

[Strötgen and Gertz2012] Jannik Strötgen and Michael Gertz. 2012. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*.

[UzZaman et al.2013] Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

[Verhagen et al.2010] Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, page 5762, Uppsala, Sweden, July. Association for Computational Linguistics.

Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems!

Laura Chiticariu
IBM Research - Almaden
San Jose, CA
chiti@us.ibm.com

Yunyao Li
IBM Research - Almaden
San Jose, CA
yunyaoli@us.ibm.com

Frederick R. Reiss
IBM Research - Almaden
San Jose, CA
frreiss@us.ibm.com

Abstract

The rise of "Big Data" analytics over unstructured text has led to renewed interest in information extraction (IE). We surveyed the landscape of IE technologies and identified a major disconnect between industry and academia: while rule-based IE dominates the commercial world, it is widely regarded as dead-end technology by the academia. We believe the disconnect stems from the way in which the two communities measure the benefits and costs of IE, as well as academia's perception that rule-based IE is devoid of research challenges. We make a case for the importance of rule-based IE to industry practitioners. We then lay out a research agenda in advancing the state-of-the-art in rule-based IE systems which we believe has the potential to bridge the gap between academic research and industry practice.

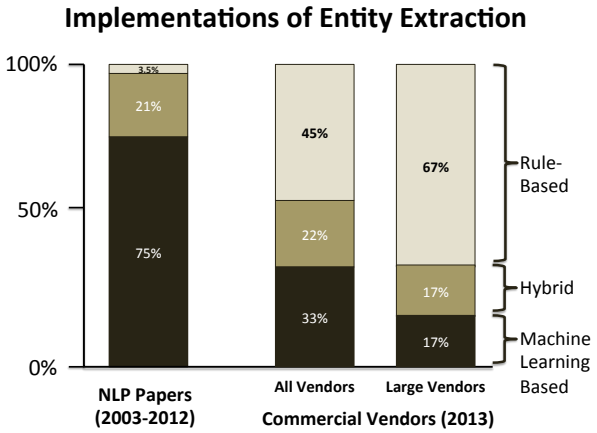


Figure 1: Fraction of NLP conference papers from EMNLP, ACL, and NAACL over 10 years that use machine learning versus rule-based techniques to perform entity extraction over text (left); the same breakdown for commercial entity extraction vendors one year after the end of this 10-year period (right). The rule-based approach, although largely ignored in the research community, dominates the commercial market.

1 Introduction

The recent growth of "Big Data" analytics over large quantities of unstructured text has led to increased interest in information extraction technologies from both academia and industry (Mendel, 2013).

Most recent academic research in this area starts from the assumption that statistical machine learning is the best approach to solving information extraction problems. Figure 1 shows empirical evidence of this trend drawn from a survey of recent published research papers. We examined the EMNLP, ACL, and NAACL conference proceedings from 2003 through 2012 and identified 177 different EMNLP research papers on the topic of entity extraction. We then classified these papers into three categories, based on the techniques used: purely rule-based, purely machine learning-based, or a hybrid of the two. We focus on entity extraction, as it is a classical IE task, and most industrial IE systems offer this feature.

The left side of the graph shows the breakdown of research papers according to this categorization. Only six papers relied solely on rules to perform the extraction tasks described. The remainder relied entirely or substantially on statistical techniques. As shown in Figure 2, these fractions were roughly constant across the 10-year period studied, indicating that attitudes regarding the relative importance of the different techniques have remained constant.

We found that distinguishing "hybrid" systems

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 827–832,
Seattle, Washington, USA, 18-21 October 2013.

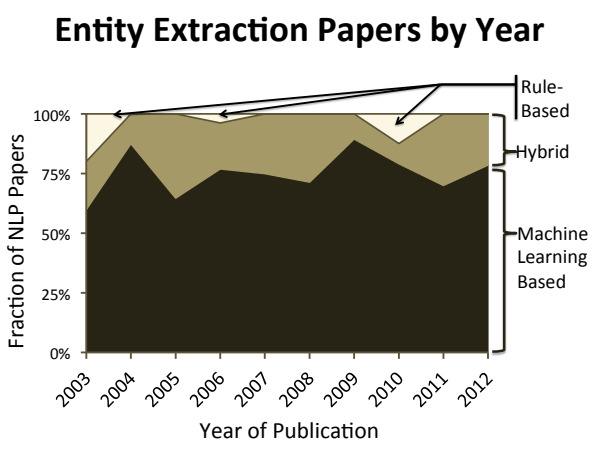


Figure 2: The conference paper data (left-hand bar) from Figure 1, broken down by year of publication. The relative fractions of the three different techniques have not changed significantly over time.

from pure machine learning systems was quite challenging. The papers that use a mixture of rule-based and machine learning techniques were generally written so as to obfuscate the use of rules, emphasizing the machine learning aspect of the work. Authors hid rules behind euphemisms such as "dependency restrictions" (Mausam et al., 2012), "entity type constraints" (Yao et al., 2011), or "seed dictionaries" (Putthividhya and Hu, 2011).

In the commercial world, the situation is largely reversed. The right side of Figure 1 shows the result of a parallel survey of commercial entity extraction products from 54 different vendors listed in (Yuen and Koehler-Kruener, 2012). We studied analyst reports and product literature, then classified each product according to the same three categories. Table 1 shows the 41 products considered in the study [1]. We conducted this industry survey in 2013, one year after the ten-year run of NLP papers we studied. One would expect the industrial landscape to reflect the research efforts of the previous 10 years, as mature technology moved from academia to industry. Instead, results of this second survey showed the opposite effect, with rule-based systems comprising the largest fraction of those surveyed. Only 1/3 of the vendors relied entirely on machine learning. Among public companies and private compa-

[1] Other products do not offer entity extraction, or we did not find sufficient evidence to classify the technology.

Table 1: Vendors and products considered in the study.

ai-one	*NathanApp*
Attensity	*Command Center*
Basis Technology	*Rosette*
Clarabridge	*Analyze*
Daedalus	*Stilus NER*
GATE	*Information Extraction*
General Sentiment	
HP	*Autonomy IDOL Eduction*
IBM	*InfoSphere BigInsights Text Analytics*
IBM	*InfoSphere Streams Text Analytics*
IBM	*SPSS Text Analytics for Surveys*
IntraFind	*iFinder NAMER*
IxReveal	*uHarmonize*
Knime	
Language Computer	*Cicero LITE*
Lexanalytics	*Salience*
alias-i	*LingPipe*
Marklogic	*Analytics & Business Intelligence*
MeshLabs	*eZi CORE*
Microsoft	*FAST Search Server*
MotiveQuest	
Nice Systems	*NiceTrack Open Source Intelligence*
OpenAmplify	*Insights*
OpenText	*Content Analytics*
Pingar	
Provalis Research	*WordStat*
Rapid-I	*Text Processing Extension*
Rocket	*AeroText*
salesforce.com	*Radian 6*
SAP	*HANA Text Analysis*
SAS	*Text Analytics*
Serendio	
Smartlogic	*Semaphore Classification and Text Mining Server*
SRA International	*NetOwl Text Analytics*
StatSoft	*STATISTICA Text Miner*
Temis	*Luxid Content Enrichment Platform*
Teradata	*(integration w/ Attensity)*
TextKernel	*Extract!*
Thompson Reuters	*OpenCalais*
Veda	*Semantics Entity Identifier*
ZyLab	*Text Mining&Analytics*

Table 2: Pros and Cons

	Pros	Cons
Rule-based	• Declarative • Easy to comprehend • Easy to maintain • Easy to incorporate domain knowledge • Easy to trace and fix the cause of errors	• Heuristic • Requires tedious manual labor
ML-based	• Trainable • Adaptable • Reduces manual effort	• Requires labeled data • Requires retraining for domain adaptation • Requires ML expertise to use or maintain • Opaque

nies with more than $100 million in revenue, the situation is even more skewed towards rule-based systems, with large vendors such as IBM, SAP, and Microsoft being completely rule-based.

2 Explaining the Disconnect

What is the source of this disconnect between research and industry? There does not appear to be a lack of interaction between the two communities. Indeed, many of the smaller companies we surveyed were founded by NLP researchers, and many of the larger vendors actively publish in the NLP literature. We believe that the disconnect arises from a difference in how the two communities measure the costs and benefits of information extraction.

Table 2 summarizes the pros and cons of machine learning (ML) and rule-based IE technologies (Atzmueller and Kluegl, 2008; Grimes, 2011; Leung et al., 2011; Feldman and Rosenfeld, 2006; Guo et al., 2006; Krishnan et al., 2005; Yakushiji et al., 2006; Kluegl et al., 2009). On the surface, both academia and commercial vendors acknowledge essentially the same pros and cons for the two approaches. However, the two communities weight the pros and cons significantly differently, leading to the drastic disconnect in Figure 1.

Evaluating the benefits of IE. Academic papers evaluate IE performance in terms of precision and recall over standard labeled data sets. This simple, clean, and objective measure is useful for judging competitions, but the reality of the business world is much more fluid and less well-defined.

In a business context, definitions of even basic entities like "product" and "revenue" vary widely from one company to another. Within any of these ill-defined categories, some entities are more important to get right than others. For example, in electronic legal discovery, correctly identifying names of executives is much more important than finding other types of person names.

In real-world applications, the output of extraction is often the input to a larger process, and it is the quality of the larger process that drives business value. This quality may derive from an aspect of extracted output that is only loosely correlated with overall precision and recall. For example, does extracted sentiment, when broken down and aggregated by product, produce an unbiased estimate of average sentiment polarity for each product?

To be useful in a business context, IE must function well with metrics that are ill-defined and subject to change. ML-based IE models, which require a careful up-front definition of the IE task, are poor fit for these metrics. The commercial world greatly values rule-based IE for its interpretability, which makes IE programs easier to adopt, understand, debug, and maintain in the face of changing requirements (Kluegl et al., 2009; Atzmueller and Kluegl, 2008). Furthermore, rule-based IE programs are valued for allowing one to easily incorporate domain knowledge, which is essential for targeting specific business problems (Grimes, 2011). As an example, an application may pose simple requirements to its entity recognition component to output only full person names, and not include salutation. With a rule-based system, such a requirement translates to removing a few rules. On the other hand, a ML-based approach requires a complete retrain.

Evaluating the costs of IE. In a business setting, the most significant costs of using information extraction are the *labor cost* of developing or adapting extractors for a particular business problem, and the *hardware cost* of compute resources required by the system.

NLP researchers generally have a well-developed sense of the labor cost of writing extraction rules, viewing this task as a "*tedious and time-consuming process*" that "*is not really practical*" (Yakushiji et al., 2006). These criticisms are valid, and, as we

point out in the next section, they motivate a research effort to build better languages and tools.

But there is a strong tendency in the NLP literature to ignore the complex and time-consuming tasks inherent in solving an extraction problem using machine learning. These tasks include: defining the business problem to be solved in strict mathematical terms; understanding the tradeoffs between different types of models in the context of the NLP task definition; performing feature engineering based on a solid working understanding of the chosen model; and gathering extensive labeled data — far more than is needed to measure precision and recall — often through clever automation.

All these steps are time-consuming; even highly-qualified workers with postgraduate degrees routinely fail to execute them effectively. Not surprisingly, in industry, ML-based systems are often deemed risky to adopt and difficult to understand and maintain, largely due to model opaqueness (Fry, 2011; Wagstaff, 2012; Malioutov and Varshney, 2013). The infeasibility of gathering labeled data in many real-world scenarios further increases the risk of committing to a ML-based solution.

A measure of the system's scalability and runtime efficiency, hardware costs are a function of two metrics: throughput and memory footprint. These figures, while extremely important for commercial vendors, are typically not reported in NLP literature. Nevertheless, our experience in practice suggests that ML-based approaches are much slower, and require more memory compared to rule-based approaches, whose throughput can be in the order of MB/second/core for complex extraction tasks like NER (Chiticariu et al., 2010).

The other explanation. Finally, we believe that the most notable reason behind the academic community's steering away from rule-based IE systems is the (false) perception of lack of research problems. The general attitude is one of *"What's the research in rule-based IE? Just go ahead and write the rules."* as indicated by anecdotal evidence and only implicitly stated in the literature, where any usage of rules is significantly underplayed as explained earlier. In the next section, we strive to debunk this perception.

3 Bridging the Gap

As NLP researchers who also work regularly with business customers, we have become increasingly worried about the gap in perception between information extraction research and industry. The recent growth of Big Data analytics has turned IE into big business (Mendel, 2013). If current trends continue, the business world will move ahead with unprincipled, ad-hoc solutions to customers' business problems, while researchers pursue ever more complex and impractical statistical approaches that become increasingly irrelevant. Eventually, the gap between research and practice will become insurmountable, an outcome in neither community's best interest.

The academic NLP community needs to stop treating rule-based IE as a dead-end technology. As discussed in Section 2, the domination of rule-based IE systems in the industry is well-justified. Even in their current form, with ad-hoc solutions built on techniques from the early 1980's, rule-based systems serve the industry needs better than the latest ML techniques. Nonetheless, there is an enormous untapped opportunity for researchers to make the rule-based approach more principled, effective, and efficient. In the remainder of this section, we lay out a research agenda centered around capturing this opportunity. Specifically, taking a systemic approach to rule-based IE, one can identify a set of research problems by separating rule development and deployment. In particular, we believe research should focus on: (a) data models and rule language, (b) systems research in rule evaluation and (c) machine learning research for learning problems in this richer target language.

Define standard IE rule language and data model. If research on rule-based IE is to move forward in a principled way, the community needs a standard way to express rules. We believe that the NLP community can replicate the success of the SQL language in connecting data management research and practice. SQL has been successful largely due to: (1) *expressivity*: the language provides all primitives required for performing basic manipulation of structured data, (2) *extensibility*: the language can be extended with new features without fundamental changes to the language, (3) *declarativity*: the language allows the specification of com-

putation logic without describing its control flow, thus allowing developers to code *what* the program should accomplish, rather than *how* to accomplish it.

An earlier attempt in late 1980's to formalize a rule language resulted in the Common Pattern Specification Language (CPSL) (Appelt and Onyshkevych, 1998). While CPSL did not succeed due to multiple drawbacks, including expressivity limitations, performance limitations, and its lack of support for core operations such as part of speech (Chiticariu et al., 2010), CPSL did gain some traction, e.g., it powers the JAPE language of the GATE open-source NLP system (Cunningham et al., 2011). Meanwhile, a number of declarative IE languages developed in the database community, including AQL (Chiticariu et al., 2010; Li et al., 2011), xLog (Shen et al., 2007), and SQL extensions (Wang et al., 2010; Jain et al., 2009), have shown that formalisms of rule-based IE systems are possible, as exemplified by (Fagin et al., 2013). However, they largely remain unknown in the NLP community.

We believe now is the right time to establish a standard IE rule language, drawing from existing proposals and experience over the past 30 years. Towards this goal, IE researchers need to answer the following questions: What is the right data model to capture text, annotations over text, and their properties? Can we establish a standard declarative extensible rule language for processing data in this model with a clear set of constructs that is sufficiently expressive to solve most IE tasks encountered so far?

Systems research based on standard IE rule language. Standard IE data model and language enables the development of systems implementing the standard. One may again wonder, *"Where is the research in that?"* As in the database community, initial research should focus on systemic issues such as data representation and speeding up rule evaluation via automatic performance optimization. Once baseline systems are established, system-related research would naturally diverge in several directions, such as extending the language with new primitives (and corresponding optimizations), and exploring modern hardware.

ML research based on standard IE rule language. A standard rule language and corresponding execution engine enables researchers to use the standard language as the expressivity of the output model, and define learning problems for this target language, including learning basic primitives such as regular expressions and dictionaries, or complete rule sets. (One need not worry about choosing the language, nor runtime efficiency.) With an expressive rule language, a major challenge is to prevent the system from generating arbitrarily complex rule sets, which would be difficult to understand or maintain. Some interesting research directions include devising proper measures for rule complexity, constraining the search space such that the learnt rules closely resemble those written by humans, active learning techniques to cope with scarcity of labeled data, and visualization tools to assist rule developers in exploring and choosing between different automatically generated rules. Finally, it is conceivable that some problems will not fit in the target language, and therefore will need alternative solutions. However, the community would have shown – objectively – that the problem is not learnable with the available set of constructs, thus motivating follow-on research on extending the standard with new primitives, if possible, or developing novel hybrid IE solutions by leveraging the standard IE rule language together with ML technology.

4 Conclusion

While rule-based IE dominates the commercial world, it is widely considered obsolete by the academia. We made a case for the importance of rule-based approaches to industry practitioners. Drawing inspiration from the success of SQL and the database community, we proposed directions for addressing the disconnect. Specifically, we call for the standardization of an IE rule language and outline an ambitious research agenda for NLP researchers who wish to tackle research problems of wide interest and value in the industry.

Acknowledgments

We would like to thank our colleagues, Howard Ho, Rajasekar Krishnamurthy, and Shivakumar Vaithyanathan, as well as the anonymous reviewers for their thoughtful and constructive comments.

References

Douglas E. Appelt and Boyan Onyshkevych. 1998. The Common Pattern Specification Language. In *Proceedings of a workshop held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER '98, pages 23–30.

Martin Atzmueller and Peter Kluegl. 2008. Rule-Based Information Extraction for Structured Data Acquisition using TextMarker. In *LWA*.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. SystemT: An Algebraic Approach to Declarative Information Extraction. In *ACL*.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. Text Processing with GATE (Version 6), Chapter 8: JAPE: Regular Expressions over Annotations.

Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. 2013. Spanners: a formal framework for information extraction. In *PODS*.

Ronen Feldman and Benjamin Rosenfeld. 2006. Boosting Unsupervised Relation Extraction by Using NER. In *EMNLP*, pages 473–481.

C. Fry. 2011. Closing the Gap between Analytics and Action. *INFORMS Analytics Mag.*, 4(6):405.

Seth Grimes. 2011. Text/Content Analytics 2011: User Perspectives on Solutions. `http://www.medallia.com/resources/item/text-analytics-market-study/`.

Hong Lei Guo, Li Zhang, and Zhong Su. 2006. Empirical study on the performance stability of named entity recognition model across domains. In *EMNLP*, pages 509–516.

Alpa Jain, Panagiotis Ipeirotis, and Luis Gravano. 2009. Building query optimizers for information extraction: the sqout project. *SIGMOD Record*, 37(4):28–34.

Peter Kluegl, Martin Atzmueller, and Frank Puppe. 2009. TextMarker: A Tool for Rule-Based Information Extraction. In *UIMA@GSCL Workshop*, pages 233–240.

Vijay Krishnan, Sujatha Das, and Soumen Chakrabarti. 2005. Enhanced answer type inference from questions using sequential models. In *HLT*, pages 315–322.

Cane Wing-ki Leung, Jing Jiang, Kian Ming A. Chai, Hai Leong Chieu, and Loo-Nin Teow. 2011. Unsupervised Information Extraction with Distributional Prior Knowledge. In *EMNLP*, pages 814–824.

Yunyao Li, Frederick Reiss, and Laura Chiticariu. 2011. Systemt: A declarative information extraction system. In *ACL*.

Dmitry M. Malioutov and Kush R. Varshney. 2013. Exact rule learning via boolean compressed sensing. In *ICML*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *EMNLP-CoNLL*, pages 523–534.

Thomas Mendel. 2013. Business Intelligence and Big Data Trends 2013. `http://www.hfsresearch.com/Business-Intelligence-and-Big-Data-Trends-2013` (accessed March 28th, 2013).

Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped Named Entity Recognition for Product Attribute Extraction. In *EMNLP*, pages 1557–1567.

Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. 2007. Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In *VLDB*, pages 1033–1044.

Kiri Wagstaff. 2012. Machine learning that matters. In *ICML*.

Daisy Zhe Wang, Eirinaios Michelakis, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. 2010. Probabilistic Declarative Information Extraction. In *ICDE*.

Akane Yakushiji, Yusuke Miyao, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2006. Automatic construction of predicate-argument structure patterns for biomedical information extraction. In *EMNLP*, pages 284–292.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured Relation Discovery using Generative Models. In *EMNLP*, pages 1456–1466.

Daniel Yuen and Hanns Koehler-Kruener. 2012. Who's Who in Text Analytics, September.

Improving Learning and Inference in a Large Knowledge-base using Latent Syntactic Cues

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{mg1,ppt,bkisiel,tom.mitchell}@cs.cmu.edu

Abstract

Automatically constructed Knowledge Bases (KBs) are often incomplete and there is a genuine need to improve their coverage. Path Ranking Algorithm (PRA) is a recently proposed method which aims to improve KB coverage by performing inference directly over the KB graph. For the first time, we demonstrate that addition of edges labeled with latent features mined from a large dependency parsed corpus of 500 million Web documents can significantly outperform previous PRA-based approaches on the KB inference task. We present extensive experimental results validating this finding. The resources presented in this paper are publicly available.

1 Introduction

Over the last few years, several large scale Knowledge Bases (KBs) such as Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010), and YAGO (Suchanek et al., 2007) have been developed. Each such KB consists of millions of facts (e.g., (*Tiger Woods*, *playsSport*, *Golf*)) spanning over multiple relations. Unfortunately, these KBs are often incomplete and there is a need to increase their coverage of facts to make them useful in practical applications.

A strategy to increase coverage might be to perform inference directly over the KB represented as a graph. For example, if the KB contained the following facts, (*Tiger Woods*, *participatesIn*, *PGA Tour*)) and (*Golf*, *sportOfTournament*, *PGA Tour*), then by putting these two facts together, we could potentially infer that (*Tiger Woods*, *playsSport*, *Golf*). The recently proposed Path Ranking Algorithm (PRA) (Lao and Cohen, 2010) performs such inference by automatically learning semantic inference rules over the KB (Lao et al., 2011). PRA uses features based off of sequences of edge types, e.g., ⟨*playsSport*, *sportOfTournament*⟩, to predict missing facts in the KB.

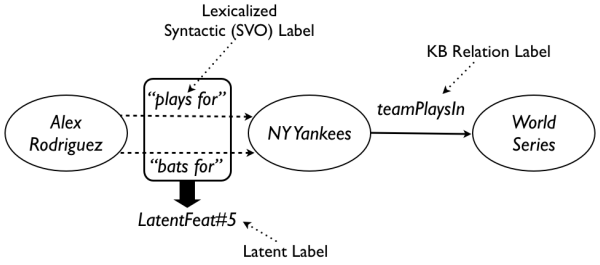


Figure 1: Example demonstrating how lexicalized syntactic edges can improve connectivity in the KB enabling PRA (Lao and Cohen, 2010) to discover relationships between *Alex Rodriguez* and *World Series*. Edges with latent labels can improve inference performance by reducing data sparsity. See Section 1.1 for details.

PRA was extended by (Lao et al., 2012) to perform inference over a KB augmented with dependency parsed sentences. While this opens up the possibility of learning syntactic-semantic inference rules, the set of syntactic edge labels used are just the *unlexicalized* dependency role labels (e.g., *nobj*, *dobj*, etc., without the corresponding words), thereby limiting overall expressitivity of the learned inference rules. To overcome this limitation, in this paper we augment the KB graph by adding edges with more expressive *lexicalized* syntactic labels (where the labels are words instead of dependen-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 833–838,
Seattle, Washington, USA, 18-21 October 2013.

cies). These additional edges, e.g., (*Alex Rodriguez*, *"plays for"*, *NY Yankees*), are mined by extracting 600 million Subject-Verb-Object (SVO) triples from a large corpus of 500m dependency parsed documents, which would have been prohibitively expensive to add directly as in (Lao et al., 2012). In order to overcome the explosion of path features and data sparsity, we derive edge labels by learning latent embeddings of the lexicalized edges. Through extensive experiments on real world datasets, we demonstrate effectiveness of the proposed approach.

1.1 Motivating Example

In Figure 1, the KB graph (only solid edges) is disconnected, thereby making it impossible for PRA to discover any relationship between *Alex Rodriguez* and *World Series*. However, addition of the two edges with SVO-based lexicalized syntactic edges (e.g., (*Alex Rodriguez*, *plays for*, *NY Yankees*)) restores this inference possibility. For example, PRA might use the edge sequence $\langle$*"plays for"*, *teamPlaysIn*$\rangle$ as evidence for predicting the relation instance (*Alex Rodriguez*, *athleteWonChampionship*, *World Series*). Unfortunately, such naïve addition of lexicalized edges may result in significant data sparsity, which can be overcome by mapping lexicalized edge labels to some latent embedding (e.g., (*Alex Rodriguez*, *LatentFeat#5*, *NY Yankees*) and running PRA over this augmented graph. Using latent embeddings, PRA could then use the following edge sequence as a feature in its prediction models: $\langle$*LatentFeat#5*, *teamPlaysIn*$\rangle$. We find this strategy to be very effective as described in Section 4.

2 Related Work

There is a long history of methods using suface-level lexical patterns for extracting relational facts from text corpora (Hearst, 1992; Brin, 1999; Agichtein and Gravano, 2000; Ravichandran and Hovy, 2002; Etzioni et al., 2004). Syntactic information in the form of dependency paths have been explored in (Snow et al., 2006; Suchanek et al., 2006). A method of latent embedding of relation instances for sentence-level relation extraction was shown in (Wang et al., 2011). However, none of this prior work makes explicit use of the background KBs as we explore in this paper.

Path Ranking Algorithm (PRA) (Lao and Cohen, 2010) has been used previously to perform inference over graph-structured KBs (Lao et al., 2011), and to learn formation of online communities (Settles and Dow, 2013). In (Lao et al., 2012), PRA is extended to perform inference over a KB using syntactic information from parsed text. In contrast to these previous PRA-based approaches where all edge labels are either KB labels or at surface-level, in this paper we explore using latent edge labels in addition to surface-level labels in the graph over which PRA is applied. In particular, we focus on the problem of performing inference over a large KB and learn latent edge labels by mining dependency syntax statistics from a large text corpus.

Though we use Principal Components Analysis (PCA) for dimensionality reduction for the experiments in this paper, this is by no means the only choice. Various other dimensionality reduction techniques, and in particular, other verb clustering techniques (Korhonen et al., 2003), may also be used.

OpenIE systems such as Reverb (Etzioni et al., 2011) also extract verb-anchored dependency triples from large text corpus. In contrast to such approaches, we focus on how latent embedding of verbs in such triples can be combined with explicit background knowledge to improve coverage of existing KBs. This has the added capability of inferring facts which are not explicitly mentioned in text.

The recently proposed Universal Schema (Riedel et al., 2013) also demonstrates the benefit of using latent features for increasing coverage of KBs. Key differences between that approach and ours include our use of syntactic information as opposed to surface-level patterns in theirs, and also the ability of the proposed PRA-based method to generate useful inference rules which is beyond the capability of the matrix factorization approach in (Riedel et al., 2013).

3 Method

3.1 Path Ranking Algorithm (PRA)

In this section, we present a brief overview of the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010), building on the notations in (Lao et al., 2012). Let $G = (V, E, T)$ be the graph, where V is the set of vertices, E is the set of edges, and T is the set of edge types. For each edge $(v_1, t, v_2) \in E$, we have

$v_1, v_2 \in V$ and $t \in T$. Let $R \subset T$ be the set of types predicted by PRA. R could in principal equal T, but in this paper we restrict prediction to KB relations, while T also includes types derived from surface text and latent embeddings. Let $\pi = \langle t_1, t_2, \ldots, t_w \rangle$ be a *path type* of length w over graph G, where $t_i \in T$ is the type of the i^{th} edge in the path. Each such path type is also a *feature* in the PRA model. For a given source and target node pair $s, t \in V$, let $P(s \rightarrow t; \pi)$ be the value of the feature π specifying the probability of reaching node t starting from node s and following a path constrained by path type π. We approximate these probabilities using random walks. A value of 0 indicates unreachability from s to t using path type π.

Let $B = \{\pi_1, \ldots, \pi_m\}$ be the set of all features (path types). The score that relation r holds between node s and node t is given by the following function:

$$\text{Score}_{\text{PRA}}(s, t, r) = \sum_{\pi \in B} P(s \rightarrow t; \pi)\, \theta_\pi^r$$

where θ_π^r is the weight of feature π in class $r \in R$.

Feature Selection: The set B of possible path types grows exponentially in the length of the paths that are considered. In order to have a manageable set of features to compute, we first perform a feature selection step. The goal of this step is to select for computation only those path types that commonly connect sources and targets of relation r. We perform this feature selection by doing length-bounded random walks from a given list of source and target nodes, keeping track of how frequently each path type leads from a source node to a target node. The most common m path types are selected for the set B.

Training: We perform standard logistic regression with L2 regularization to learn the weights θ_π^r. We follow the strategy in (Lao and Cohen, 2010) to generate positive and negative training instances.

3.2 $\text{PRA}_{\text{syntactic}}$

In this section, we shall extend the knowledge graph $G = (V, E, T)$ from the previous section with an augmented graph $G' = (V, E', T')$, where $E \subset E'$ and $T \subset T'$, with the set of vertices unchanged.

In order to get the edges in $E' - E$, we first collect a set of Subject-Verb-Object (SVO) triples $D = \{(s, v, o, c)\}$ from a large dependency parsed text corpus, with $c \in \mathbb{R}_+$ denoting the frequency of this triple in the corpus. The additional edge set is then defined as $E_{\text{syntactic}} = E' - E = \{(s, v, o) \mid \exists (s, v, o, c) \in D,\ s, o \in V\}$. We define $S = \{v \mid \exists (s, v, o) \in E_{\text{syntactic}}\}$ and set $T' = T \cup S$. In other words, for each pair of directly connected nodes in the KB graph G, we add an additional edge between those two nodes for each verb which takes the NPs represented by two nodes as subjects and objects (or vice versa) as observed in a text corpus. In Figure 1, (*Alex Rodriguez*, *"plays for"*, *NY Yankees*) is an example of such an edge.

PRA is then applied over this augmented graph G', over the same set of prediction types R as before. We shall refer to this version of PRA as $\text{PRA}_{\text{syntactic}}$. For the experiments in this paper, we collected $|D| = 600$ million SVO triples[1] from the entire ClueWeb corpus (Callan et al., 2009), parsed using the Malt parser (Nivre et al., 2007) by the Hazy project (Kumar et al., 2013).

3.3 $\text{PRA}_{\text{latent}}$

In this section we construct $G'' = (V, E'', T'')$, another syntactic-information-induced extension of the knowledge graph G, but instead of using the surface forms of verbs in S (see previous section) as edge types, we derive those edges types T'' based on latent embeddings of those verbs. We note that $E \subset E''$, and $T \subset T''$.

In order to learn the latent or low dimensional embeddings of the verbs in S, we first define $Q_S = \{(s, o) \mid \exists (s, v, o, c) \in D,\ v \in S\}$, the set of subject-object tuples in D which are connected by at least one verb in S. We now construct a matrix $X_{|S| \times |Q_S|}$ whose entry $X_{v,q} = c$, where $v \in S, q = (s, o) \in Q_S$, and $(s, v, o, c) \in D$. After row normalizing and centering matrix X, we apply PCA on this matrix. Let $A_{|S| \times d}$ with $d << |Q_S|$ be the low dimensional embeddings of the verbs in S as induced by PCA. We use two strategies to derive mappings for verbs from matrix A.

- $\text{PRA}_{\text{latent}_c}$: The verb is mapped to concatenation of the $\frac{k}{2}$ most positive columns in the row in A that corresponds to the verb. Similarly, for the most negative $\frac{k}{2}$ columns.

[1] This data and other resources from the paper are publicly available at http://rtw.ml.cmu.edu/emnlp2013_pra/.

	Precision	Recall	F1
PRA	0.800	0.331	0.468
$\text{PRA}_{\text{syntactic}}$	0.804	0.271	0.405
$\text{PRA}_{\text{latent}_c}$	**0.885**	0.334	0.485
$\text{PRA}_{\text{latent}_d}$	0.868	**0.424**	**0.570**

Table 1: Comparison of performance of different variants of PRA micro averaged across 15 NELL relations. We find that use of latent edge labels, in particular the proposed approach $\text{PRA}_{\text{latent}_d}$, significantly outperforms other approaches. This is our main result. (See Section 4)

- $\text{PRA}_{\text{latent}_d}$: The verb is mapped to disjunction of top-k most positive and negative columns in the row in A that corresponds to the verb.

4 Experiments

We compared the various methods using 15 NELL relations. For each relation, we split NELL's known relation instances into 90% training and 10% testing. For each method, we then selected 750 path features and trained the model, as described in Section 3, using GraphChi (Kyrola et al., 2012) to perform the random walk graph computations. To evaluate the model, we took all source nodes in the testing data and used the model to predict target nodes. We report the precision and recall (on the set of known target nodes) of the set of predictions for each model that are above a certain confidence threshold. Because we used strong regularization, we picked for our threshold a model score of 0.405, corresponding to 60% probability of the relation instance being true; values higher than this left many relations without any predictions. Table 1 contains the results.

As can be seen in the table, $\text{PRA}_{\text{syntactic}}$ on average performs slightly worse than PRA. While the extra syntactic features are very informative for some relations, they also introduce a lot of sparsity, which makes the model perform worse on other relations. When using latent factorization methods to reduce the sparsity of the syntactic features, we see a significant improvement in performance. $\text{PRA}_{\text{latent}_c}$ has a 45% reduction in precision errors vs. PRA while maintaining the same recall, and $\text{PRA}_{\text{latent}_d}$ reduces precision errors by 35% while improving recall by 27%. Section 4.1 contains some qualitative analysis of how sparsity is reduced with the latent methods. As a piece quantitative analysis, there were 908 possible path types found in the feature selection step with PRA on the relation *cityLiesOnRiver* (of which we then selected 750). For $\text{PRA}_{\text{syntactic}}$, there were 73,820, while $\text{PRA}_{\text{latent}_c}$ had 47,554 and $\text{PRA}_{\text{latent}_d}$ had 58,414.

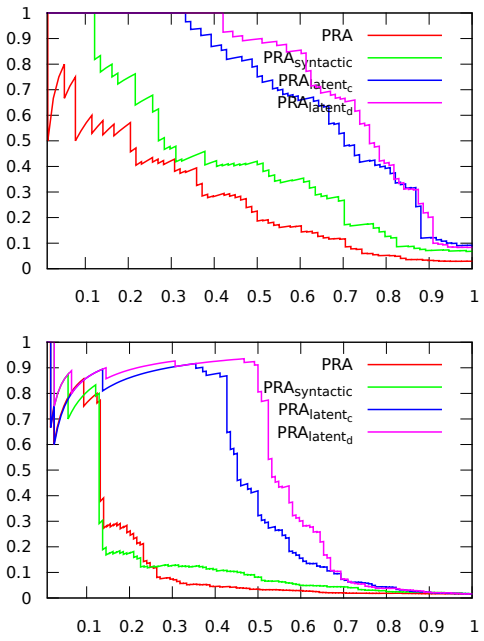


Figure 2: Precision (y axis) - Recall (x axis) plots for the relations *cityLiesOnRiver* (top) and *athletePlaysForTeam* (bottom). $\text{PRA}_{\text{latent}_d}$ (rightmost plot), the proposed approach which exploits latent edge labels, outperforms other alternatives.

Table 2 shows F1 scores for each model on each relation, and Figure 2 shows representative Precision-Recall plots for two NELL relations. In both cases, we find that $\text{PRA}_{\text{latent}_d}$ significantly outperforms other baselines.

4.1 Discussion

While examining the model weights for each of the methods, we saw a few occasions where surface relations and NELL relations combined to form interpretable path types. For example, in *athletePlaysForTeam*, some highly weighted features took the form of $\langle$*athletePlaysSport*, *"(sport) played by (team)"*$\rangle$. A high weight on this feature would bias the prediction towards teams that are known to play the same sport as the athlete.

For PRA, the top features for the best performing relations are path types that contain a single edge

	PRA	$\text{PRA}_{\text{syntactic}}$	$\text{PRA}_{\text{latent}_c}$	$\text{PRA}_{\text{latent}_d}$
animalIsTypeOfAnimal	0.52	0.50	0.47	**0.53**
athletePlaysForTeam	0.22	0.21	0.56	**0.64**
athletePlaysInLeague	**0.81**	0.75	0.73	0.74
cityLiesOnRiver	0.05	0	0.07	**0.31**
cityLocatedInCountry	0.15	0.20	0.45	**0.55**
companyCeo	0.29	0.18	0.25	**0.35**
countryHasCompanyOffice	0	0	0	0
drugHasSideEffect	**0.96**	0.95	0.94	0.94
headquarteredIn	0.31	0.11	0.41	**0.64**
locationLocatedWithinLocation	0.40	0.38	0.38	**0.41**
publicationJournalist	0.10	0.06	0.10	**0.16**
roomCanContainFurniture	0.72	0.70	0.71	**0.73**
stadiumLocatedInCity	0.53	0	0.13	**0.67**
teamPlaysAgainstTeam	**0.47**	0.24	0.26	0.21
writerWroteBook	0.59	0.62	0.73	**0.80**

Table 2: F1 performance of different variants of PRA for all 15 relations tested.

which is a supertype or subtype of the relation being predicted. For instance, for the relation *athletePlaysForTeam* (shown in Figure 2), the highest-weighted features in PRA are *athleteLedSportsTeam* (more specific than *athletePlaysForTeam*) and *personBelongsToOrganization* (more general than *athletePlaysForTeam*). For the same relation, $\text{PRA}_{\text{syntactic}}$ has features like *"scored for"*, *"signed"*, *"have"*, and *"led"*. When using a latent embedding of these verb phrases, *"signed"*, *"have"*, and *"led"* all have the same representation in the latent space, and so it seems clear that $\text{PRA}_{\text{latent}}$ gains a lot by reducing the sparsity inherent in using surface verb forms.

For *cityLiesOnRiver*, where PRA does not perform as well, there is no NELL relation that is an immediate supertype or subtype, and so PRA does not have as much evidence to use. It finds features that, e.g., are analogous to the statement "cities in the same state probably lie on the same river". Adding lexical labels gives the model edges to use like *"lies on"*, *"runs through"*, *"flows through"*, *"starts in"* and *"reaches"*, and these features give a significant boost in performance to $\text{PRA}_{\text{syntactic}}$. Once again, almost all of those verb phrases share the same latent embedding, and so $\text{PRA}_{\text{latent}}$ gains another significant boost in performance by combining them into a single feature.

5 Conclusion

In this paper, we introduced the use of latent lexical edge labels for PRA-based inference over knowledge bases. We obtained such latent edge labels by mining a large dependency parsed corpus of 500 million web documents and performing PCA on the result. Through extensive experiments on real datasets, we demonstrated that the proposed approach significantly outperforms previous state-of-the-art baselines.

Acknowledgments

We thank William Cohen (CMU) for enlightening conversations on topics discussed in this paper. We thank the ClueWeb project (CMU) and the Hazy Research Group (http://hazy.cs.wisc.edu/hazy/) for their generous help with data sets; and to the anonymous reviewers for their constructive comments. This research has been supported in part by DARPA (under contract number FA8750-13-2-0005), and Google. Any opinions, findings, conclusions and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM conference on Digital libraries*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web.

J. Callan, M. Hoy, C. Yoo, and L. Zhao. 2009. Clueweb09 data set. *boston.lti.cs.cmu.edu*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of WWW*.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of IJCAI*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational Linguistics*.

Anna Korhonen, Yuval Krymolowski, and Zvika Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of ACL*.

Arun Kumar, Feng Niu, and Christopher Ré. 2013. Hazy: making it easier to build and maintain big-data analytics. *Communications of the ACM*, 56(3):40–49.

Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. 2012. Graphchi: Large-scale graph computation on just a pc. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 31–46.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP-CoNLL*.

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

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*.

Burr Settles and Steven Dow. 2013. Let's get together: the formation and success of online creative collaborations. In *Proceedings of CHI*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of ACL*.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of KDD*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*.

Chang Wang, James Fan, Aditya Kalyanpur, and David Gondek. 2011. Relation extraction with relation topics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1426–1436. Association for Computational Linguistics.

What is Hidden among Translation Rules

Libin Shen
Persado
50 West 17th Street
New York, NY 10011
libin.shen@persado.com

Bowen Zhou
IBM T. J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
zhou@us.ibm.com

Abstract

Most of the machine translation systems rely on a large set of translation rules. These rules are treated as discrete and independent events. In this short paper, we propose a novel method to model rules as observed generation output of a compact hidden model, which leads to better generalization capability. We present a preliminary generative model to test this idea. Experimental results show about one point improvement on TER-BLEU over a strong baseline in Chinese-to-English translation.

1 Introduction

Most of the modern Statistical Machine Translation (SMT) systems, for example (Koehn et al., 2003; Och and Ney, 2004; Chiang, 2005; Marcu et al., 2006; Shen et al., 2008), employ a large rule set that may contain tens of millions of translation rules or even more. In these systems, each translation rule has about 20 dense features, which represent key statistics collected from the training data, such as word translation probability, phrase translation probability etc. Except for these common features, there is no connection among the translation rules. The translation rules are treated as independent events.

The use of sparse features as in (Arun and Koehn, 2007; Watanabe et al., 2007; Chiang et al., 2009) to some extent mitigated this problem. In their work, there are as many as 10,000 features defined on the appearance of certain frequent words and Part of Speech (POS) tags in rules. They provide significant improvement in automatic evaluation metrics. However, these sparse features fire quite randomly and infrequently on each rule. Thus, there is still plenty of space to better model translation rules.

In this paper, we will explore the relationship among translation rules. We no longer view rules as discrete or unrelated events. Instead, we view rules, which are observed from training data, as random variables generated by a hidden model. This generative process itself is also hidden. All possible generative processes can be represented with factorized structures such as weighted hypergraphs and finite state machines. This approach leads to a compact model that has better generalization capability and allows translation rules not explicitly observed in training date.

This paper reports work-in-progress to exploit hidden relations among rules. Preliminary experiments show about one point improvement on TER-BLEU over a strong baseline in Chinese-to-English translation.

2 Hidden Models

Let $\mathcal{G} = \{(r, f)\}$ be a grammar observed from parallel training data, where f is the frequency of a bilingual translation rule r.

Let $\mathcal{M}$ be a hidden model that generates every translation rule r. For example, $\mathcal{M}$ could be modeled with a weighted hypergraph or finite state machine. For the sake of convenience, in this section we assume $\mathcal{M}$ is a meta-grammar $\mathcal{M} = \{m\}$, where each m represents a meta-rule. For each translation r, there exists a hypergraph H_r that represents all possible derivations $D_r = \{d\}$ that can generate rule r. Here, each derivation d is a hyperpath using meta-rules M_d, where $M_d \subseteq \mathcal{M}$. Thus, we can use hypergraph H_r to characterize r. Translation rules in $\mathcal{G}$

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 839–844,
Seattle, Washington, USA, 18-21 October 2013.

can share nodes and meta-rules in their hypergraphs, so that $\mathcal{M}$ is more compact model than $\mathcal{G}$.

In the rest of this section, we will introduce three methods to quantify H_r as features of rule r. It should be noted that there are more ways to exploit the compact model of $\mathcal{M}$ than these three.

2.1 Type 1 : A Generative Model

Let θ be the parameters of a statistical model $Pr(m;\theta)$ for meta-rules m in meta-grammar $\mathcal{M}$ estimated from the observed translation grammar $\mathcal{G}$. The probability of a translation rule r can be calculated as follows.

$$\begin{aligned} Pr(r;\theta) &\propto Pr(H_r;\theta) \\ &= \sum_{d \in D_r} Pr(d;\theta) \end{aligned} \quad (1)$$

By assuming separability,

$$Pr(d;\theta) = \prod_{m \in M_d} Pr(m;\theta) \quad (2)$$

we can further decompose rule probability $Pr(r;\theta)$ as below.

$$Pr(r;\theta) = \sum_{d \in D_r} \prod_{m \in M_d} Pr(m;\theta) \quad (3)$$

In practice, $Pr(r;\theta)$ in (3) can be calculated through bottom-up dynamic programming on hypergraph H_r. Hypergraphs of different rules can share nodes and meta-rules. This reveals the underlying relationship among translation rules.

As a by-product of this generative model, we use the log-likelihood of a translation rule, $\log Pr(r;\theta)$, as a new dense feature. We call it *Type 1* in experiments.

2.2 Type 2 : Meta-Rules as Sparse Features

As given in (3), likelihood of a translation rule is a function over $Pr(m;\theta)$, in which θ is estimated from the training data with a generative model. Previous work in (Chiang et al., 2009) showed the advantage of using a discriminative model to optimize individual weights for these factors towards a better automatic score.

Following this practice, we treat each meta-rule m as a sparse feature. Feature value $f(m) = 1$ if and only if m is used in hypergraph H_r. Otherwise, its default value is 0. We call these features *Type 2* in experiments. The Type 2 system contains the log-likelihood feature in Type 1.

2.3 Type 3 : Posterior as Feature Values

A natural question on the binary sparse features defined above is why all the active features have the same value of 1. We use these meta-rules to represent a translation rule in feature space. Intuitively, for meta-rules with closer connection to the translation rules, we hope to use relatively larger feature values to increase their effect.

We formalize this intuition with the posterior probability that a meta-rule m is used to generate r, as below.

$$\begin{aligned} f(m) &\equiv Pr(m|r;\theta) \\ &= \frac{Pr(m,r;\theta)}{Pr(r;\theta)} \\ &= \frac{\sum_{d \in D_r, m \in M_d} Pr(d;\theta)}{Pr(r;\theta)} \end{aligned} \quad (4)$$

The posterior in (4) could be too sharp. Following the common practice, we smooth the posterior features with a scaling factor α.

$$f(m) \equiv Pr(m|r)^{\alpha}$$

We use *Type 3*(α) to represent the posterior model with a scaling factor of α in experiments. The Type 3 systems also contain the log-likelihood feature in Type 1.

2.4 Parameter Estimation

Now we explain how to obtain parameter θ. With proper definition of the underlying model $\mathcal{M}$, we can estimate θ with the traditional EM algorithm or Bayesian methods.

In the next section, we will present an example of the hidden model. We will employ the EM algorithm to estimate the parameters in θ. Here, translation rules and their frequencies in $\mathcal{G}$ are observed data, and derivation d for each rule r is hidden. At the *Expectation* step, we search all derivations d in D_r of each rule r and calculate their probabilities according to equation (2). At the *Maximization* step, we re-estimate θ on all derivations in proportion to their posterior probability.

3 Case Study

In Section 2, we explored the use of meta-grammars as the underlying model $\mathcal{M}$ and developed three methods to define features. Similar techniques can be applied to finite state machines and other underlying models. Now, we introduce a POS-based underlying model to illustrate the generic model proposed in Section 2. We will show experimental results in Section 4.

3.1 Meta-rules on POS tags

Let $r \in \mathcal{G}$ be a translation rule composed of a pair of source and target word strings (F_w, E_w). Let F_p and E_p be the POS tags for the source and target sides respectively. For the sake of simplicity as the first attempt, we treat non-terminal as a special word X with POS tag X.

Suppose we have a Chinese-to-English translation rule as below.

$$yuehan\ qu\ zhijiage \Rightarrow john\ leaves\ for\ chicago$$

We call

$$NR\ VV\ NR \Rightarrow NNP\ VBZ\ IN\ NNP \quad (5)$$

a translation rule in POS tags.

We will propose an underlying model $\mathcal{M}$ to generate translation rules in POS tags instead of translation rules themselves. For the rest of this section, we take translation rules in POS tags as the target of our generative model. We define meta-rules on pairs of POS tag strings, e.g. $NR\ VV \Rightarrow NNP\ VBZ$.

We can decompose the probability of translation rule in (5) into a product on meta-rule probabilities via various derivations, such as

- $Pr(NR\ VV,\ NNP\ VBZ) \times Pr(NR,\ IN\ NNP)$, and
- $Pr(NR,\ NNP) \times Pr(VV,\ VBZ\ IN) \times Pr(NR,\ NNP)$.

3.2 The Underlying Model and Features

Now, we introduce a generative model M for translation rules in POS tags. We still use the example in (5) as shown in Figure 1, where the top box represents the source side and the bottom box represents the target side. Dotted lines represent word alignments on three pairs of words.

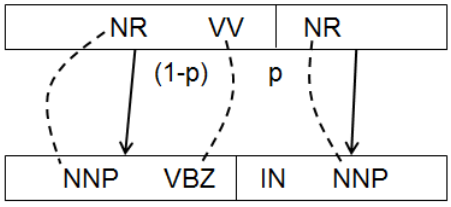


Figure 1: An example

We first generate the number of source tokens of a translation rule with a uniform distribution for up to, for example, 7 tokens.

Then we split the source side into chunks with a binomial distribution with a Bernoulli variable at the gap between each two continuous words, which splits the two words into two chunks with a probability of p. For example, the probability of obtaining two chunks *NR VV* and *NR* is $(1-p)p$, as shown in Figure 1.

Suppose we split the target side into two parts, *NNP VBZ* and *IN NNP*, which respects the word alignments. It generates two meta-rules *NR VV* $\Rightarrow$ *NNP VBZ* and *NR* $\Rightarrow$ *IN NNP*, as shown in Figure 1. The probability for the first meta-rule is

$$Pr(|E| = 2 \mid |F| = 2) \times$$

$$Pr(NR\ VV, NNP\ VBZ \mid |F| = 2, |E| = 2),$$

where $|F|$ represents the number of source tokens, and $|E|$ the number of target tokens. Similarly, the probability of the second one is as follows.

$$Pr(|E| = 2 \mid |F| = 1) \times$$

$$Pr(NR, IN\ NNP \mid |F| = 1, |E| = 2).$$

To sum up, the probability of a derivation d for a translation rule $r : F \Rightarrow E$ is

$$\begin{aligned} Pr(d) \approx\ & Pr_{\theta_1}(|F|) \\ & \times Pr_{\theta_2}(F_s) \\ & \times \prod_{m \in M_d} Pr_{\theta_3}(|E_m| \mid |F_m|) \\ & \times \prod_{m \in M_d} Pr_{\theta_4}(m \mid |F_m|, |E_m|) \end{aligned} \quad (6)$$

where F_m and E_m are source and target sides of a meta-rule m used in derivation d, and F_s is a splitting of the source side. As for the distributions, we

have

$$\begin{aligned}\theta_1 &\sim Uniform \\ \theta_2 &\sim Binomial \\ \theta_3 &\sim Categorical \\ \theta_4 &\sim Categorical\end{aligned}$$

where θ_1 and θ_2 have pre-selected hyperparameters, and θ_3 and θ_4 are estimated with the EM algorithm.

As for sparse features, we will obtain 7 meta-rule features as below.

- *NR* $\Rightarrow$ *NNP*
- *VV* $\Rightarrow$ *VBZ*
- *VV* $\Rightarrow$ *VBZ IN*
- *NR VV* $\Rightarrow$ *NNP VBZ*
- *NR VV* $\Rightarrow$ *NNP VBZ IN*
- *VV NR* $\Rightarrow$ *VBZ IN NNP*
- *NR VV NR* $\Rightarrow$ *NNP VBZ IN NNP*

All of them respect the word alignment, which means that

- there is no alignment that aligns one word in a meta-rule with the other out of the same meta-rule, and
- there is at least one alignment within a meta-rule.

3.3 Implementation Details

Even though the size of all possible meta-rules is much smaller than the space of translation rules, it is still too large to work with existing optimization methods for sparse features in MT, i.e. MIRA (Chiang et al., 2009) or L-BFGS (Matsoukas et al., 2009). In practice, we have to limit the feature space to around 20,000 dimensions.

For this purpose, we first use a frequency based method to filter meta-rule features. Specifically, we first divide all the meta-rules into 100 bins, $(|F|, |E|)$, where $|F|$ is the number of words on the source side, and $|E|$ the target side, $0 < |F|, |E| \leq 10$. For each bin, we keep the same top k-percentile of the meta-rules such that we obtain a total of 20,000 meta-rules as features.

System	BLEU%	TER%	T-B
Baseline	30.35	55.32	24.97
Type 1	30.74	55.48	24.74
Type 2	31.07	55.07	24.00
Type 3 (1)	30.93	55.34	24.41
Type 3 (0.1)	31.05	55.02	23.97
Type 3 (0.01)	31.09	54.96	23.87

Table 1: scores on test-1

A shortcoming of this filtering method is that all these features are positive indicators, while low-frequency negative indicators are discarded. In order to keep the features of various level of frequency, we define class features with a 3-tuple $C(|F|, |E|, q)$, where $|F|$ and $|E|$ are numbers of source and target words as defined above, and q is the integer part of the log_2 value of the feature frequency in the training data.

In this way, each meta-rule feature can be mapped to one of these classes. The value of a class feature equals the sum of the meta-rule features that mapped into this class. We have about 2,000 class features defined in this way. They are applied on both Type 2 and Type 3 features.

4 Experiments

We carry out our experiments on web genre of Chinese-to-English translation. The training set contains about 10 million parallel sentences available to Phase 1 of the DARPA BOLT MT task. The tune set contains 1275 sentences. Each has four references. There are two test sets. Test-1 is from a similar source of the tune set, and it contains 1239 sentences. Test-2 is the web part of the MT08 evaluation data.

Our baseline system is a home-made Hiero (Chiang, 2005) style system. The baseline rule set contains about 17 million rules. It contains about 40 dense features, including a 6-gram LM.

The sparse feature optimization algorithm is similar to the MIRA recipe described in (Chiang et al., 2009). We optimize on TER-BLEU (Snover et al., 2006; Papineni et al., 2001).

The BLEU, TER and T-B scores on the two tests are shown in Tables 1 and 2. It should be noted that, even though our metric of tuning is T-B, the baseline

System	BLEU%	TER%	T-B
Baseline	25.80	56.96	31.16
Type 1	26.18	57.09	30.91
Type 2	26.63	56.64	30.01
Type 3 (1)	26.30	57.00	30.70
Type 3 (0.1)	26.34	56.73	30.39
Type 3 (0.01)	26.50	56.73	30.23

Table 2: scores on test-2 (MT08-WB)

system already provides a very competitive BLEU score on MT08-WB as compared the best system in the evaluation[1], thanks to comprehensive features in the baseline system and more data in training.

All the three types of systems provide consistent improvement on both test sets in terms of T-B, our optimization metric. Type 1 gives marginal improvement of 0.2. This shows the limitation of the generative feature. When we use meta-rules as binary sparse features in Type 2, we obtain about one point improvement on T-B on both sets. This shows the advantage of tuning individual meta-rule weights over a generative model. Type 3 (0.01) and Type 2 are at the same level. Proper smoothing is important to Type 3.

5 Discussion

In the case study of Section 3, we use POS-based rules as hidden states. However, it should be noted that the hidden structures surely do not have to be POS tags. For example, an alternative could be unsupervised NT splitting similar to (Huang et al., 2010).

The meta-grammar based approach was also motivated by the insight acquired on mono-lingual linguistic grammar generation, especially in the TAG related research (Xia, 2001; Prolo, 2002). Meta-grammar was viewed as an effective way to remove redundancy in grammars.

The link between Tree Adjoining Grammar (TAG) (Joshi et al., 1975; Joshi and Schabes, 1997) and MT was first introduced in (Shieber and Schabes, 1990), a pioneer work in tree-to-tree translation. (DeNeefe and Knight, 2009) re-visited the use of adjoining operation in the context of Statistical MT, and reported encouraging results. On the other hand, (Dras, 1999) showed how a meta-level grammar could help in modeling parallel operations in (Shieber and Schabes, 1990). Our work is another effort of statistical modeling of well-recognized linguistic insight in NLP and MT.

[1] http://www.itl.nist.gov/iad/mig/tests/mt/2008/

6 Conclusions and Future Work

In this paper, we introduced a novel method to model translation rules as observed generation output of a compact hidden model. As a case study to capitalize this model, we presented three methods to enrich rule modeling with features defined on a hidden model. Preliminary experiments verified gain of one point on TER-BLEU over a strong baseline in Chinese-to-English translation.

As for future work, we plan to extend this work in the following aspects.

- To try other prior distributions to generate the number of source tokens.
- Unsupervised and semi-supervised learning of hidden models.
- To incorporate rich models into the generative process, e.g. reordering, non-terminals, structural information and lexical models.
- To improve the posterior model with better parameter estimation, e.g. Bayesian methods.
- To replace the exhaustive translation rule set with a compact meta grammar that can create and parameterize new translation rules dynamically, which is the ultimate goal of this line of work.

Acknowledgments

We would like thank the anonymous reviewers for their valuable comments. Haitao Mi and Martin Cmejrek kindly helped on data preparation.

This work was done when the first author was at IBM. The work was supported by DARPA under Grant HR0011-12-C-0015 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the ofcial views or policies, either expressed or implied, of the DARPA.

References

Abhishek Arun and Philipp Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *Proceedings of MT Summit XI*.

D. Chiang, K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of the 2009 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, Ann Arbor, MI.

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 727–736, Singapore.

Mark Dras. 1999. A meta-level grammar: redefining synchronous tag for translation and paraphrase. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhongqiang Huang, Martin Cmejrek, and Bowen Zhou. 2010. Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proceedings of the 2010 Conference of Empirical Methods in Natural Language Processing*.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer-Verlag.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54, Edmonton, Canada.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference of Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia.

Spyros Matsoukas, Antti-Veikko Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*.

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

Kishore Papineni, Salim Roukos, and Todd Ward. 2001. Bleu: a method for automatic evaluation of machine translation. IBM Research Report, RC22176.

Carlos Prolo. 2002. Generating the xtag english grammar using metarules. In *Proceedings of the 19th international conference on Computational linguistics (COLING)*.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Stuart Shieber and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of COLING '90: The 13th Int. Conf. on Computational Linguistics*, pages 253–258, Helsinki, Finland.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA.

T. Watanabe, J. Suzuki, H. Tsukuda, and H. Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*.

F. Xia. 2001. *Automatic Grammar Generation From Two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.

Converting Continuous-Space Language Models into N-gram Language Models for Statistical Machine Translation

Rui Wang[1,2,3], **Masao Utiyama**[2], **Isao Goto**[2], **Eiichro Sumita**[2], **Hai Zhao**[1,3] **and Bao-Liang Lu**[1,3]
1 Center for Brain-Like Computing and Machine Intelligence,
Department of Computer Science and Engineering,
Shanghai Jiao Tong Unviersity, Shanghai, 200240, China
2 Multilingual Translation Laboratory, MASTAR Project,
National Institute of Information and Communications Technology
3-5 Hikaridai, Keihanna Science City, Kyoto, 619-0289, Japan
3 MOE-Microsoft Key Lab. for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong Unviersity, Shanghai 200240 China
wangrui.nlp@gmail.com, mutiyama/igoto/eiichiro.sumita@nict.go.jp, zhaohai@cs.sjtu.edu.cn, bllu@sjtu.edu.cn

Abstract

Neural network language models, or continuous-space language models (CSLMs), have been shown to improve the performance of statistical machine translation (SMT) when they are used for reranking n-best translations. However, CSLMs have not been used in the first pass decoding of SMT, because using CSLMs in decoding takes a lot of time. In contrast, we propose a method for converting CSLMs into back-off n-gram language models (BNLMs) so that we can use converted CSLMs in decoding. We show that they outperform the original BNLMs and are comparable with the traditional use of CSLMs in reranking.

1 Introduction

Language models are important in natural language processing tasks such as speech recognition and statistical machine translation. Traditionally, back-off n-gram language models (BNLMs) (Chen and Goodman, 1996; Chen and Goodman, 1998; Stolcke, 2002) are being widely used for these tasks.

Recently, neural network language models, or continuous-space language models (CSLMs) (Bengio et al., 2003; Schwenk, 2007; Le et al., 2011) are being used in statistical machine translation (SMT) (Schwenk et al., 2006; Son et al., 2010; Schwenk et al., 2012; Son et al., 2012; Niehues and Waibel, 2012). These works have shown that CSLMs can improve the BLEU (Papineni et al., 2002) scores of SMT when compared with BNLMs, on the condition that the training data for language modeling are the same size. However, in practice, CSLMs have not been widely used in SMT.

One reason is that the computational costs of training and using CSLMs are very high. Various methods have been proposed to tackle the training cost issues (Son et al., 2010; Schwenk et al., 2012; Mikolov et al., 2011). However, there has been little work on reducing using costs. Since the using costs of CSLMs are very high, it is difficult to use CSLMs in decoding directly.

A common approach in SMT using CSLMs is the two pass approach, or n-best reranking. In this approach, the first pass uses a BNLM in decoding to produce an n-best list. Then, a CSLM is used to rerank those n-best translations in the second pass. (Schwenk et al., 2006; Son et al., 2010; Schwenk et al., 2012; Son et al., 2012)

Another approach is using restricted Boltzmann machines (RBMs) (Niehues and Waibel, 2012) instead of using multi-layer neural networks (Bengio et al., 2003; Schwenk, 2007; Le et al., 2011). Since probability in a RBM can be calculated very efficiently (Niehues and Waibel, 2012), they can use the RBM language model in SMT decoding. However, the RBM was just used in an adaptation of SMT, not in a large SMT task, because the training costs of RBMs are very high.

The last approach is using a BNLM to simulate a CSLM (Deoras et al., 2011; Arsoy et al., 2013). (Deoras et al., 2011) used a recurrent neural network language model (RNNLM) to generate a large amount of text, which was generated by sampling words from the probability distributions calculated by the RNNLM. Then, they trained the BNLM

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 845–850,
Seattle, Washington, USA, 18-21 October 2013.

from the text using the interpolated Kneser-Ney smoothing method. (Arsoy et al., 2013) converted neural network language models of increasing order to pruned back-off language models, using lower-order models to constrain the n-grams allowed in higher-order models.

Both of these methods were used in decoding for speech recognition. These methods were applied to not-so-large scale experiments (55 million (M) words for training their BNLMs) (Arsoy et al., 2013). In contrast, our method is applied to SMT and can be used to improve a BNLM created from 746 M words by using a CSLM trained from 42 M words.

Because BNLMs can be trained from much larger corpora than those that can be used for training CSLMs, improving a BNLM by using a CSLM trained from a smaller corpus is very important. Actually, a CSLM trained from a smaller corpus can improve the BLEU scores of SMT if it is used in the n-best reranking (Schwenk, 2010; Huang et al., 2013). In contrast, we will demonstrate that a BNLM simulating a CSLM can improve the BLEU scores of SMT in the first pass decoding.

Our approach is as follows: (1) First, we train a CSLM (Schwenk, 2007) from a corpus. (2) Second, we also train a BNLM from the same corpus or larger corpus. (3) Finally, we rewrite the probability of each n-gram of the BNLM with that probability calculated from the CSLM. We also re-normalize the probabilities of the BNLM, then use the re-written BNLM in SMT decoding.

In Section 2, we describe the BNLM and CSLM (Schwenk, 2010) used for re-writing BNLMs. In Section 3, we describe the method of converting a CSLM into a BNLM. In Sections 4 and 5, we evaluate our method and conclude.

2 Language Models

In this section, we will introduce the standard BNLM and CSLM structure and probability calculation.

2.1 Standard back-off ngram language model

A BNLM predicts the probability of a word w_i given its preceding $n-1$ words $h_i = w_{i-n+1}^{i-1}$. But it will suffer from data sparseness if the context, h_i, does not appear in the training data. So an estimation by "backing-off" to models with smaller histories is necessary. In the case of the modified Kneser-Ney smoothing (Chen and Goodman, 1998), the probability of w_i given h_i under a BNLM, $P_b(w_i|h_i)$, is:

$$P_b(w_i|h_i) = \hat{P}_b(w_i|h_i) + \gamma(h_i)P_b(w_i|w_{i-n+2}^{i-1}) \quad (1)$$

where $\hat{P}_b(w_i|h_i)$ is a discounted probability and $\gamma(h_i)$ is the back-off weight. A BNLM is used with a CSLM as shown below.

2.2 CSLM structure and probability calculation

The main structure of a CSLM using a multi-layer neural network contains four layers: the input layer projects all words in the context h_i onto the projection layer (the first hidden layer); the second hidden layer and the output layer achieve the non-liner probability estimation and calculate the language model probability $P(w_i|h_i)$ for the given context. (Schwenk, 2007).

The CSLM calculates the probabilities of all words in the vocabulary of the corpus given the context at once. However, because the computational complexity of calculating the probabilities of all words is quite high, the CSLM is only used to calculate the probabilities of a subset of the whole vocabulary. This subset is called a *short-list*, which consists of the most frequent words in the vocabulary. The CSLM also calculates the sum of the probabilities of all words not in the short-list by assigning a neuron for that purpose. The probabilities of other words not in the short-list are obtained from a BNLM (Schwenk, 2007; Schwenk, 2010).

Let w_i, h_i be the current word and history. The CSLM with a BNLM calculates the probability of w_i given h_i, $P(w_i|h_i)$, as follows:

$$P(w_i|h_i) = \begin{cases} \frac{P_c(w_i|h_i)}{1-P_c(o|h_i)} P_s(h_i) & \text{if } w_i \in \text{short-list} \\ P_b(w_i|h_i) & \text{otherwise} \end{cases} \quad (2)$$

where $P_c(\cdot)$ is the probability calculated by the CSLM, $P_c(o|h_i)$ is the probability of the neuron for the words not in the short-list, $P_b(\cdot)$ is the probability calculated by the BNLM as in Eq. 1, and

$$P_s(h_i) = \sum_{v \in \text{short-list}} P_b(v|h_i). \quad (3)$$

It can be considered that the CSLM redistributes the probability mass of all words in the short-list. This probability mass is calculated by using the BNLM.

3 Conversion of CSLM into BNLM

As described in the introduction, we first train a CSLM from a corpus. We also train a BNLM from the same corpus or a larger corpus. Then, we rewrite the probability of each ngram in the BNLM with the probability calculated from the CSLM.

First, we use the probabilities of 1-grams in the BNLM as they are. Next, we rewrite the probabilities of n-grams (n=2,3,4,5) in the BNLM with the probabilities calculated by using the n-gram CSLM, respectively. Note that the n-gram CSLM means that the length of its history is $n-1$. Note also that we only need to rewrite the probabilities of n-grams ending with a word in the short-list. Finally, we re-normalize the probabilities of the BNLM using the SRILM's '-renorm' option.

When we rewrite a BNLM trained from a larger corpus, the ngrams in the BNLM often contain unknown words for the CSLM. In that case, we use the probabilities in the BNLM as they are.

4 Experiments

4.1 Common settings

We used the patent data for the Chinese to English patent translation subtask from the NTCIR-9 patent translation task (Goto et al., 2011). The parallel training, development, and test data consisted of 1 M, 2,000, and 2,000 sentences, respectively.

We followed the settings of the NTCIR-9 Chinese to English translation baseline system (Goto et al., 2011) except that we used various language models to compare them. We used the MOSES phrase-based SMT system (Koehn et al., 2003), together with Giza++ (Och and Ney, 2003) for alignment and MERT (Och, 2003) for tuning on the development data. The translation performance was measured by the case-insensitive BLEU scores on the tokenized test data. We used `mteval-v13a.pl` for calculating BLEU scores.[1]

We used the 14 standard SMT features: five translation model scores, one word penalty score, seven distortion scores and one language model score. Each of the different language models was used to calculate the language model score.

As the baseline BNLM, we trained a 5-gram BNLM with modified Kneser-Ney smoothing using the English side of the 1 M sentences training data, which consisted of 42 M words. We did not discard any n-grams in training this model. That is, we did not use count cutoffs. We call this BNLM as *BNLM42*.

A 5-gram CSLM was trained on the same 1 M training sentences using the CSLM toolkit (Schwenk, 2010). The settings for the CSLM were: projection layer of dimension 256 for each word, hidden layer of dimension 384 and output layer (short-list) of dimension 8192, which were recommended in the CSLM toolkit. We call this CSLM *CSLM42*. CSLM42 used BNLM42 as the background BNLM.

We also trained a larger 5-gram BNLM with modified Kneser-Ney smoothing by adding sentences from the 2005 US patent data distributed in the NTCIR-8 patent translation task (Fujii et al., 2010) to the 42 M words. The data consisted of 746 M words. We call this BNLM *BNLM746*. We discarded 3,4,5-grams that occurred only once when we created BNLM746.

Next, we re-wrote BNLM42 with CSLM42 by using the method described in Section 3. This re-written BNLM was interpolated with BNLM42. The interpolation weight was determined by the grid search. That is, we changed the interpolation weight to 0.1, 0.3, 0.5, 0.7, 0.9 to create an interpolated BNLM. Then we used that BNLM in the SMT system to tune the weight parameters on the first half of the development data. Next, we selected the interpolation weight that obtained the highest BLEU score on the second half of the development data. After we selected the interpolation weight, we applied MERT again to the 2,000 sentence development data to tune the weight parameters.[2] We call this BNLM *CONV42*. We also obtained *CONV746* by re-writing BNLM746 with CSLM42

[1] It is available at `http://www.itl.nist.gov/iad/mig/tests/mt/2009/`

[2] We aware that the interpolation weight might be determined by minimizing the perplexity on the development data. However, we opted to directly maximize the BLEU score.

in the same way.

The vocabulary of these language models was the same, which was extracted from the 1 M training sentences.

4.2 Experimental results

Table 1 shows the percent BLEU scores on the test data. The figures in the "1st pass" column show the BLEU scores in the first pass decoding when we changed the language model. The figures in the "reranking" column show the BLEU scores when we applied CSLM42 to rerank the 100-best lists for the different language models. When we applied CSLM42 for reranking, we added the CSLM42 score as the additional 15th feature. The weight parameters were tuned by using Z-MERT (Zaidan, 2009).

LMs	1st pass	rerank
BNLM42	31.60	32.44
CONV42	32.58	32.98
BNLM746	32.83	33.36
CONV746	33.22	33.54

Table 1: Comparison of BLEU scores

We also performed the paired bootstrap resampling test (Koehn, 2004).[3] We sampled 2000 samples for each significance test.

Table 2 shows the results of a statistical significance test, in which the "1st" is short for the "1st pass". The marks indicate whether the LM to the left of a mark is significantly better than that above the mark at a certain level. ("$\gg$": significantly better at $\alpha = 0.01$, "$>$": $\alpha = 0.05$, "$-$": not significantly better at $\alpha = 0.05$)

First, as shown in the tables, the reranking by applying CSLM42 increased the BLEU scores for all language models. This observation is in accordance with those of previous work (Schwenk, 2010; Huang et al., 2013).

Second, the reranking results of BNLM42 (32.44) were not better than those of the first pass of BNLM746 (32.83). This indicates that if the underlying BNLM is made from a small corpus, the reranking using CSLM can not compensate for it.

[3]We used the code available at `http://www.ark.cs.cmu.edu/MT/`.

	BNLM746 (rerank)	CONV746 (1st)	CONV42 (rerank)	BNLM746 (1st)	CONV42 (1st)	BNLM42 (rerank)	BNLM42 (1st)
CONV746 (rerank)	$-$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
BNLM746 (rerank)		$-$	$\gg$	$>$	$\gg$	$\gg$	$\gg$
CONV746 (1st)			$\gg$	$-$	$\gg$	$\gg$	$\gg$
CONV42 (rerank)				$-$	$\gg$	$\gg$	$\gg$
BNLM746 (1st)					$-$	$\gg$	$\gg$
CONV42 (1st)						$-$	$\gg$
BNLM42 (rerank)							$\gg$

Table 2: Significance tests for systems with different LMs

Third, CONV42 was better than BNLM42 for both first-pass and reranking. This also holds in the case of CONV746 and BNLM746. This indicated that our conversion method improved the BNLMs, even if the underlying BNLM was trained on a larger corpus than that used for training the CSLM. As described in the introduction, this is very important because BNLMs can be trained from much larger corpora than those that can be used for training CSLMs. This observation has not been found in the previous work.

In addition, the first-pass of CONV42 and CONV746 (32.58 and 33.22) were comparable with those of the reranking results of BNLM42 and BNLM746 (32.44 and 33.36), respectively. That is, there were no significant differences between these results. This indicates that our conversion method preserves the performance of the reranking using CSLM.

5 Conclusion

We have proposed a method for converting CSLMs into BNLMs. The method can be used to improve a BNLM by using a CSLM trained from a smaller corpus than that used for training the BNLM. We have also shown that BNLMs created by our method performs as good as the reranking using CSLMs.

Our future work is to compare our conversion method with that of (Arsoy et al., 2013).[4]

[4]We aware that (Arsoy et al., 2013) compared their method with the one that is identical with our method. However, the experiments were conducted on a speech recognition task and the scale of the experiment was not so large. Since we noticed their work just before the submission of our paper, we did not have time to compare their method with our method in SMT.

Acknowledgments

We appreciate the helpful discussion with Andrew Finch and Paul Dixon, and three anonymous reviewers for many invaluable comments and suggestions to improve our paper. This work is supported by the National Natural Science Foundation of China (Grant No. 60903119, No. 61170114 and No. 61272248), the National Basic Research Program of China (Grant No. 2013CB329401) and the Science and Technology Commission of Shanghai Municipality (Grant No. 13511500200).

References

Ebru Arsoy, Stanley F. Chen, Bhuvana Ramabhadran, and Abhinav Sethy. 2013. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. In *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2013)*, Vancouver, Canada, May. IEEE.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3:1137–1155, March.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Santa Cruz, California, June. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard Univ.

A. Deoras, T. Mikolov, S. Kombrink, M. Karafiat, and Sanjeev Khudanpur. 2011. Variational approximation of long-span language models for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5532–5535, Prague, Czech Republic, May. IEEE.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2010. Overview of the patent translation task at the ntcir-8 workshop. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pages 293–302, Tokyo, Japan, June.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 559–578, Tokyo, Japan, December.

Zhongqiang Huang, Jacob Devlin, and Spyros Matsoukas. 2013. Bbn's systems for the chinese-english sub-task of the ntcir-10 patentmt evaluation. In *NTCIR-10*, Tokyo, Japan, June.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Edmonton, Canada. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Hai-Son Le, I. Oparin, A. Allauzen, J. Gauvain, and F. Yvon. 2011. Structured output layer neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5524–5527, Prague, Czech Republic, May. IEEE.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernock. 2011. Strategies for training large scale neural network language models. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 196–201, Prague, Czech Republic, May. IEEE.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using restricted boltzmann machines. In *Proceedings of the International Workshop for Spoken Language Translation*, IWSLT 2012, pages 311–318, Hong Kong.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–

318, Philadelphia, Pennsylvania, June. Association for Computational Linguistics.

Holger Schwenk, Daniel Dchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 723–730, Sydney, Australia, July. Association for Computational Linguistics.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, WLM '12, pages 11–19, Montreal, Canada, June. Association for Computational Linguistics.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Holger Schwenk. 2010. Continuous-space language models for statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, pages 137–146.

Le Hai Son, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon. 2010. Training continuous space language models: some practical issues. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 778–788, Cambridge, Massachusetts, October. Association for Computational Linguistics.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 39–48, Montreal, Canada, June. Association for Computational Linguistics.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

A Corpus Level MIRA Tuning Strategy for Machine Translation

Ming Tan, Tian Xia, Shaojun Wang
Wright State University
3640 Colonel Glenn Hwy,
Dayton, OH 45435 USA
{tan.6, xia.7, shaojun.wang}
@wright.edu

Bowen Zhou
IBM T.J. Watson Research Center
1101 Kitchawan Rd,
Yorktown Heights, NY 10598 USA
zhou@us.ibm.com

Abstract

MIRA based tuning methods have been widely used in statistical machine translation (SMT) system with a large number of features. Since the corpus-level BLEU is not decomposable, these MIRA approaches usually define a variety of heuristic-driven sentence-level BLEUs in their model losses. Instead, we present a new MIRA method, which employs an exact corpus-level BLEU to compute the model loss. Our method is simpler in implementation. Experiments on Chinese-to-English translation show its effectiveness over two state-of-the-art MIRA implementations.

1 Introduction

Margin infused relaxed algorithm (MIRA) has been widely adopted for the parameter optimization in SMT with a large feature size (Watanabe et al., 2007; Chiang et al., 2008; Chiang et al., 2009; Chiang, 2012; Eidelman, 2012; Cherry and Foster, 2012). Since BLEU is defined on the corpus, and not decomposed into sentences, most MIRA approaches consider a variety of sentence-level BLEUs for the model losses, many of which are heuristic-driven (Watanabe et al., 2007; Chiang et al., 2008; Chiang et al., 2009; Chiang, 2012; Cherry and Foster, 2012). The sentence-level BLEU appearing in the objective is generally based on a pseudo-document, which may not precisely reflect the corpus-level BLEU. We believe that this mismatch could potentially harm the performance. To avoid the sentence BLEU, the work in (Haddow et al., 2011) proposed to process sentences in small batches. The authors adopted a Gibbs sampling (Arun et al., 2009) technique to search the *hope* and *fear* hypotheses, and they did not compare with MIRA. Watanabe (2012) also tuned the parameters with small batches of sentences and optimized a hinge loss not explicitly related to BLEU using stochastic gradient descent. Both approaches introduced additional complexities over baseline MIRA approaches.

In contrast, we propose a remarkably simple but efficient batch MIRA approach which exploits the exact corpus-level BLEU to compute model losses. We search for a *hope* and a *fear* hypotheses for the corpus with a straightforward approach and minimize the structured hinge loss defined on them. The experiments show that our method consistently outperforms two state-of-the-art MIRAs in Chinese-to-English translation tasks with a moderate margin.

2 Margin Infused Relaxed Algorithm

We optimize the model parameters based on *N*-best lists. Our development (*dev*) set is a set of triples $\{(f_i, \mathbf{e}_i, \mathbf{r}_i)\}_{i=1}^{M}$, where f_i is a source-language sentence, corresponded by a list of target-language hypotheses $\mathbf{e}_i = \{e_{ij}\}_{j=1}^{N(f_i)}$, with a number of references $\mathbf{r}_i$. $\mathbf{h}(e_{ij})$ is a feature vector. Generally, most decoders return a top-1 candidate as the translation result, such that $\bar{e}_i(\mathbf{w}) = \arg\max_j \mathbf{w} \cdot \mathbf{h}(e_{ij})$, where $\mathbf{w}$ are the model parameters. In this paper, we aim at optimizing the BLEU score (Papineni et al., 2002).

MIRA is an instance of online learning which assumes an overlap of the decoding procedure and the parameter optimization procedure. For example in (Crammer et al., 2006; Chiang et al., 2008), MIRA

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 851–856,
Seattle, Washington, USA, 18-21 October 2013.

is performed after an input sentence are decoded, and the next sentence is decoded with the updated parameters. The objective for each sentence i is,

$$\min_{\mathbf{w}} \quad \frac{1}{2}||\mathbf{w}-\mathbf{w}'||^2 + C \cdot l_i(\mathbf{w}) \tag{1}$$

$$l_i(\mathbf{w}) = \max_{e_{ij}} \{\mathrm{b}(e_i^*) - \mathrm{b}(e_{ij}) - \mathbf{w}\cdot[\mathbf{h}(e_i^*) - \mathbf{h}(e_{ij})]\} \tag{2}$$

where $e_i^* \in \mathbf{e_i}$ is a *hope* candidate, $\mathbf{w}'$ is the parameter vector from the last sentence. Since MIRA defines its objective only based on the current sentence, $\mathrm{b}(\cdot)$ is a sentence-level BLEU.

Most MIRA algorithms need a deliberate definition of $\mathrm{b}(\cdot)$, since BLEU cannot be decomposed into sentences. The types of the sentence BLEU calculation includes: (a) a smoothed version of BLEU for e_{ij} (Liang et al., 2006), (b) fit e_{ij} into a pseudo-document considering the history (Chiang et al., 2008; Chiang, 2012), (c) use e_{ij} to replace the corresponding hypothesis in the oracles (Watanabe et al., 2007). The sentence-level BLEU sometimes perplexes the algorithms and results in a mismatch with the corpus-level BLEU.

3 Corpus-level MIRA

3.1 Algorithm

We propose a batch tuning strategy, corpus-level MIRA (c-MIRA), in which an objective is not built upon a hinge loss of a single sentence, but upon that of the entire corpus.

The online MIRAs are difficult to parallelize. Therefore, similar to the batch MIRA in (Cherry and Foster, 2012), we conduct the batch tuning by repeating the following steps: (a) Decode source sentences (in parallel) and obtain $\{\mathbf{e_i}\}_{i=1}^M$, (b) Merge $\{\mathbf{e_i}\}_{i=1}^M$ with the one from the previous iteration, (c) Invoke Algorithm 1.

We define $\mathcal{E} = (e_{\mathcal{E},1}, e_{\mathcal{E},2}, ..., e_{\mathcal{E},M})$ as a corpus hypothesis, with $\mathbf{H}(\mathcal{E}) = \frac{1}{M}\sum_{i=1}^{M}\mathbf{h}(e_{\mathcal{E},i})$. $e_{\mathcal{E},i}$ is the hypothesis of the source sentence f_i covered by $\mathcal{E}$. $\mathcal{E}$ is corresponded to a corpus-level BLEU, which we ultimately want to optimize. Following MIRA formulated in (Crammer et al., 2006; Chiang et al., 2008), c-MIRA repeatedly optimizes,

$$\min_{\mathbf{w}} \quad \frac{1}{2}||\mathbf{w}-\mathbf{w}'||^2 + C \cdot l_{corpus}(\mathbf{w}) \tag{3}$$

$$l_{corpus}(\mathbf{w}) = \max_{\mathcal{E}} \{\mathrm{B}(\mathcal{E}^*) - \mathrm{B}(\mathcal{E}) - \mathbf{w}\cdot[\mathbf{H}(\mathcal{E}^*) - \mathbf{H}(\mathcal{E})]\} \tag{4}$$

where $\mathrm{B}(\cdot)$ is a corpus-level BLEU. $\mathcal{E}^*$ is a *hope* hypothesis. $\mathcal{E} \in \mathcal{L}$, where $\mathcal{L}$ is the hypothesis space of the entire corpus, and $|\mathcal{L}| = |\mathbf{e_1}|\cdots|\mathbf{e_M}|$.

Algorithm 1 Corpus-Level MIRA

Require: $\{(f_i, \mathbf{e}_i, \mathbf{r}_i)\}_{i=1}^M$, $\mathbf{w_0}$, C

for $t = 1 \cdots T$ **do**
  $\mathcal{E}^* = \{\}$,$\mathcal{E}' = \{\}$ ▷ Initialize the *hope* and *fear*
  for $i = 1 \cdots M$ **do**
    $e_{\mathcal{E}^*,i} = \arg\max_{e_{ij}} [\mathbf{w_{t-1}} \cdot \mathbf{h}(e_{ij}) + \mathrm{b}'(e_{ij})]$
    $e_{\mathcal{E}',i} = \arg\max_{e_{ij}} [\mathbf{w_{t-1}} \cdot \mathbf{h}(e_{ij}) - \mathrm{b}'(e_{ij})]$
    $\mathcal{E}^* \leftarrow \mathcal{E}^* + \{e_{\mathcal{E}^*,i}\}$ ▷ Build the *hope*
    $\mathcal{E}' \leftarrow \mathcal{E}' + \{e_{\mathcal{E}',i}\}$ ▷ Build the *fear*
  end for
  $\triangle_{\mathrm{B}} = \mathrm{B}(\mathcal{E}^*) - \mathrm{B}(\mathcal{E}')$ ▷ the BLEU difference
  $\triangle_{\mathbf{H}} = \mathbf{H}(\mathcal{E}') - \mathbf{H}(\mathcal{E}^*)$ ▷ the feature difference
  $\alpha = \min\left[C, \frac{\triangle_{\mathrm{B}} + \mathbf{w_{t-1}}\cdot\triangle_{\mathbf{H}}}{||\triangle_{\mathbf{H}}||^2}\right]$
  $\mathbf{w_t} = \mathbf{w_{t-1}} - \alpha \cdot \triangle_{\mathbf{H}}$
  $\bar{\mathbf{w}}_t = \frac{1}{t+1}\sum_{t=0}^{t}\mathbf{w}_t$
end for
return $\bar{\mathbf{w}}_t$ with the optimal BLEU on the *dev* set.

c-MIRA can be regarded as a standard MIRA, in which there is only one single triple $(\mathcal{F}, \mathcal{L}, \mathcal{R})$, where $\mathcal{F}$ and $\mathcal{R}$ are the source and reference of the corpus respectively. Eq. 3 is equivalent to a quadratic programming with $|\mathcal{L}|$ constraints. Crammer et al. (2006) show that a single constraint with one *hope* $\mathcal{E}^*$ and one *fear* $\mathcal{E}'$ admits a closed-form update and performs well. We denote one execution of the outer loop as an *epoch*. The *hope* and *fear* are updated in each *epoch*. Similar to (Chiang et al., 2008), the *hope* and *fear* hypotheses are defined as following,

$$\mathcal{E}^* = \max_{\mathcal{E}}[\mathbf{w}\cdot\mathbf{H}(\mathcal{E}) + \mathrm{B}(\mathcal{E})] \tag{5}$$

$$\mathcal{E}' = \max_{\mathcal{E}}[\mathbf{w}\cdot\mathbf{H}(\mathcal{E}) - \mathrm{B}(\mathcal{E})] \tag{6}$$

Eq. 5 and 6 find the hypotheses with the best and worse BLEU that the decoder can easily achieve. It is unnecessary to search the entire space of $\mathcal{L}$ for precise solution $\mathcal{E}^*$and $\mathcal{E}'$, because MIRA only at-

tempts to separate the *hope* from the *fear* by a margin proportional to their BLEU differentials (Cherry and Foster, 2012). We just construct $\mathcal{E}^*$and $\mathcal{E}'$ respectively by,

$$e_{\mathcal{E}^*,i} = \max_{e_{i,j}}[\mathbf{w}\cdot\mathbf{h}(e_{i,j}) + \mathrm{b}'(e_{i,j})]$$
$$e_{\mathcal{E}',i} = \max_{e_{i,j}}[\mathbf{w}\cdot\mathbf{h}(e_{i,j}) - \mathrm{b}'(e_{i,j})]$$

where b' is simply a BLEU with add one smoothing (Lin and Och, 2004). A smoothed BLEU is good enough to pick up a "satisfying" pair of *hope* and *fear*. However, the updating step (Line 11) uses the corpus-level BLEU.

3.2 Justification

c-MIRA treats a corpus as one sentence for decoding, while conventional decoders process sentences one by one. We show the optimal solutions from the two methods are equivalent theoretically.

We follow the notations in (Och and Ney, 2002). We search a hypothesis on corpus $\mathcal{E} = \{e_{1,k_1}, e_{2,k_2}, ..., e_{M,k_M}\}$ with the highest probability given the source corpus $\mathcal{F} = \{\mathrm{f}_1, \mathrm{f}_2, ..., \mathrm{f}_\mathrm{M}\}$,

$$\begin{aligned}\overline{\mathcal{E}} &= \arg\max_{\mathcal{E}} \log P(\mathcal{E}|\mathcal{F}) \\ &= \arg\max_{\mathcal{E}} \left(\mathbf{w}\cdot\sum_{i=1}^{M}\mathbf{h}(e_{i,k_i}) - \sum_{\mathrm{i}=1}^{\mathrm{M}}\log(\mathrm{Z_i})\right) \quad (7) \\ &= \{\arg\max_{e_{i,k_i}} \mathbf{w}\cdot\mathbf{h}(e_{i,k_i})\}_{i=1}^{M} \quad (8)\end{aligned}$$

where $Z_i = \sum_{j=1}^{N(f_i)}\exp(\mathbf{w}\cdot\mathbf{h}(e_{i,j}))$, which is a constant with respective to $\mathcal{E}$. Eq. 7 shows that the feature vector of $\mathcal{E}$ is determined by the sum of each candidate's feature vectors. Also, the model score can be decomposed into each sentence in Eq. 8, which shows that decoding all sentences together equals to decoding one by one.

We also show that if the metric is decomposable, the loss in c-MIRA is actually the sum of the hinge loss $l_i(\mathbf{w})$ in structural SVM (Tsochantaridis et al., 2004; Cherry and Foster, 2012). We assume $B(e_{ij})$ to be the metric of a sentence hypothesis, then the loss of c-MIRA in Eq. 4 is,

$$\begin{aligned}l_{corpus}(\mathbf{w}) &\propto \max_{\mathcal{E}'}\sum_{i=1}^{M}[B(e_{i,k_{\mathcal{E}^*}}) - B(e_{i,k_{\mathcal{E}'}}) \\ &\quad -\mathbf{w}\cdot\mathbf{h}(e_{i,k_{\mathcal{E}^*}}) + \mathbf{w}\cdot\mathbf{h}(e_{i,k_{\mathcal{E}'}})] \\ &= \sum_{i=1}^{M}\max_{e_{ij}}[B(e_{i,k_{\mathcal{E}^*}}) - B(e_{ij}) \\ &\quad -\mathbf{w}\cdot\mathbf{h}(e_{i,k_{\mathcal{E}^*}}) + \mathbf{w}\cdot\mathbf{h}(e_{ij})] = \sum_{i=1}^{M} l_i(\mathbf{w})\end{aligned}$$

Instead of adopting a cutting-plane algorithm (Tsochantaridis et al., 2004), we optimize the same loss with a MIRA pattern in a simpler way. However, since BLEU is not decomposable, the structural SVM (Cherry and Foster, 2012) uses an interpolated sentence BLEU (Liang et al., 2006). Although Algorithm 1 has an outlook similar to the batch-MIRA algorithm in (Cherry and Foster, 2012), their loss definitions differ fundamentally. Batch MIRA basically uses a sentence-level loss, and they also follow the sentence-by-sentence tuning pattern. In the future work, we will compare structural SVM and c-MIRA under decomposable metrics like WER or SSER (Och and Ney, 2002).

4 Experiments and Analysis

We first evaluate c-MIRA in a iterative batch tuning procedure in a Chinese-to-English machine translation system with 228 features. Second, we show c-MIRA is also effective in the re-ranking task with more than 50,000 features.

In both experiments, we compare c-MIRA and three baselines: (1) MERT (Och, 2003), (2) Chiang et al.'s MIRA (MIRA_1) in (Chiang et al., 2008). (3) batch-MIRA (MIRA_2) in (Cherry and Foster, 2012). Here, we roughly choose C with the best BLEU on *dev* set, from $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$. We convert Chiang et al.'s MIRA to the batch mode described in section 3.1. So the only difference between MIRA_1 and MIRA_2 is: MIRA_1 obtains multiple constraints before optimization, while MIRA_2 only uses one constraint. We implement MERT and MIRA_1, and directly use MIRA_2 from Moses (Koehn et al., 2007). We conduct experiments in a server of 8-cores with 2.5GHz Opteron. We set the maximum number of *epochs* as we generally do not observe an obvious increase on the *dev* set BLEU.

		MERT	MIRA_1	MIRA_2	c-MIRA
	C		0.0001	0.001	0.0001
8	*dev*	**34.80**	34.70	34.73	34.70
feat.	04	**31.92**	31.81	31.73	31.83
	05	28.85	**28.94**	28.71	28.92
	C		0.001	0.001	0.001
all	*dev*	34.61	35.24	35.14	**35.56**
feat.	04	31.76	32.25	32.04	**32.57+**
	05	28.85	**29.43**	29.37	29.41
	06news	30.91	31.43	31.24	**31.82+**
	06others	27.43	28.01	28.13	**28.45**
	08news	25.62	26.11	26.03	**26.40**
	08others	16.22	16.66	16.46	**17.10+**

Table 1: BLEUs (%) on the *dev* and *test* sets with 8 dense features only and all features. The significant symbols (+ at 0.05 level) are compared with MIRA_2

The *epoch* size for MIRA_1 and MIRA_2 is 40, while the one for c-MIRA is 400. c-MIRA runs more *epochs*, because we update the parameters by much fewer times. However, we can implement Line 3$\sim$8 in Algorithm 1 in multi-thread (we use eight threads in the following experiments), which makes our algorithm much faster. Also, we increase the *epoch* sizes of MIRA_1 and MIRA_2 to 400, and find there is no improvement on their performance.

4.1 Iterative Batch Training

In this experiment, we conduct the batch tuning procedure shown in section 3. We align the FBIS data including about 230K sentence pairs with GIZA++ for extracting grammar, and train a 4-gram language model on the Xinhua portion of Gigaword corpus. A hierarchical phrase-based model (Chiang, 2007) is tuned on NIST MT 2002, which has 878 sentences, and tested on MT 2004, 2005, 2006, and 2008. All features used here, besides eight basic ones in (Chiang, 2007), consists of an extra 220 group features. We design such feature templates to group grammar by the length of source side and target side, $(feat\_type, a \leq src\_side \leq b, c \leq tgt\_side \leq d)$, where $feat\_type$ denotes any of relative frequency, reversed relative frequency, lexical probability and reversed lexical probability, and $[a, b]$, $[c, d]$ enumerate all possible subranges of $[1, 10]$, as the maximum length on each side of a hierarchical grammar is limited to 10. There are 4×55 extra group features. We also set the size of *N*-best list per sentence before merge as 200.

	MERT	MIRA_1	MIRA_2	c-MIRA
R. T.	25.8min	16.0min	7.3min	7.8min

Table 2: Running time.

All methods use 30 decoding iterations. We select the iteration with the best BLEU of the *dev* set for testing. We present the BLEU scores in Table 1 on two feature settings: (1) 8 basic features only, and (2) all 228 features. In the first case, due to the small feature size, MERT can get a better BLEU of the *dev* set, and all MIRA algorithms fails to generally beat MERT on the *test* set. However, as the feature size increase to 228, MERT degrades on the *dev*-set BLEU, and also become worse on *test* sets, while MIRA algorithms improve on the *dev* set expectedly. MIRA_1 performs better than MIRA_2, probably because of more constraints. c-MIRA can moderately improve BLEU by 0.2$\sim$0.4 from MIRA_1 and 0.2$\sim$0.6 from MIRA_2. This might indicate that a loss defined on corpus is more accurate than the one defined on sentence. Table 2 lists the running time. Only MIRA_2 is fairly faster than c-MIRA because of more *epochs* in c-MIRA.

4.2 Re-ranking Experiments

The baseline system is a state-of-the-art hierarchical phrase-based system, and trained on six million parallel sentences corpora available to the DARPA BOLT Chinese-English task. This system includes 51 dense features (including translation probabilities, provenance features, etc.) and about 50k sparse features (mostly lexical and fertility-based). The language model is a six-gram model trained on a 10 billion words monolingual corpus, including the English side of our parallel corpora plus other corpora such as Gigaword (LDC2011T07) and Google News. We use 1275 sentences for tuning and 1239 sentences for testing from the LDC2010E30 corpus respectively. There are four reference translations for each input sentence in both tuning and testing datasets.

We use a N-best list which is an intermediate out-

		MIRA_1	MIRA_2	c-MIRA
dense	dev	31.90	31.78	**32.00**
only	test	30.89	30.89	**31.07**
dense	dev	32.29	32.20	**32.49**
+sparse	test	31.12	31.00	**31.39**

Table 3: BLEUs (%) on re-ranking experiments.

MIRA_1	MIRA_2	c-MIRA
about 1,966,720	35,120	400

Table 4: Times of updating model parameters.

put of the baseline system optimized on TER-BLEU instead of BLEU. Before the re-ranking task, the initial BLEUs of the top-1 hypotheses on the tuning and testing set are 31.45 and 30.56. The average numbers of hypotheses per sentence are about 200 and 500, respectively for the tuning and testing sets. Again, we use the best *epoch* on the tuning set for testing. The BLEUs on *dev* and *test* sets are reported in Table 3. We observe that the effectiveness of c-MIRA is not harmed as the feature size is scaled up.

4.3 Analysis

To examine the simple search for *hopes* and *fears* (Line 3∼8 in Alg. 1), we use two *hope/fear* building strategies to get $\mathcal{E}^*$ and $\mathcal{E}'$: (1) simply connect each e_i^* and e_i' in Line 4∼5 of Algorithm 1, (2) conduct a slow beam search among the N-best lists of all foreign sentences from $\mathbf{e_1}$ to $\mathbf{e_M}$ and use Eq. 5 and 6 to prune the stack. The stack size is 10. We observe that there is no significant difference between the two strategies on the BLEU of the *dev* set. But the second strategy is about 10 times slower.

We also consider more constraints in Eq. 3. By beam search, we obtain one corpus-level oracle and 29 other hypotheses similar to (Chiang et al., 2008), and optimize with SMO (Platt, 1998). Unfortunately, experiments show that more constraints lead to an overfitting and no improved performance.

As shown in Table 4, in one execution, our method updates the parameters by only 400 times; MIRA_2 updates by $40 \times 878 = 35120$ times; and MIRA_1 updates much more (about 1,966,720 times) due to the SMO procedure. We are surprised to find c-MIRA gets a higher training BLEU with such few parameter updates. This probably suggests that there is a gap between sentence-level BLEU and corpus-level BLEU, so standard MIRAs need to update the parameters more often.

Regarding simplicity, MIRA_1 uses a strongly-heuristic definition of a sentence BLEU, and MIRA_2 needs a pseudo-document with a decay rate of $\gamma = 0.9$. In comparison, c-MIRA avoids both the sentence level BLEU and the pseudo-document, thus needs fewer variables.

5 Conclusion

We present a simple and effective MIRA batch tuning algorithm without the heuristic-driven calculation of sentence-level BLEU, due to the indecomposability of a corpus-level BLEU. Our optimization objective is directly defined on the corpus-level hypotheses. This work simplifies the tuning process, and avoid the mismatch between the sentence-level BLEU and the corpus-level BLEU. This strategy can be potentially applied to other optimization paradigms, such as the structural SVM (Cherry and Foster, 2012), SGD and AROW (Chiang, 2012), and other forms of samples, such as forests (Chiang, 2012) and lattice (Cherry and Foster, 2012).

6 Acknowledgments

The key idea and a part of the experimental work of this paper were developed in collaboration with the IBM researcher when the first author was an intern at IBM T.J. Watson Research Center. This research is partially supported by Air Force Office of Scientific Research under grant FA9550-10-1-0335, the National Science Foundation under grant IIS RI-small 1218863 and a Google research award.

References

A. Arun, C. Dyer, B. Haddow, P. Blunsom, A. Lopez, and P. Koehn. 2009. Monte Carlo inference and maximization for phrase-based translation. *In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*, 102-110.

C. Cherry and G. Foster. 2012. Batch tuning strategies for statistical machine translation. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 427-436.

D. Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research (JMLR)*, 1159-1187.

D. Chiang, K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 218-226.

D. Chiang, Y. Marton, and P. Resnik. 2008. Online large-margin training of syntactic and structural translation features. *In Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 224-233.

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

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551-585.

V. Eidelman. 2012. Optimization strategies for online large-margin learning in machine translation. *Proceedings of the Seventh Workshop on Statistical Machine Translation*, 480-489.

B. Haddow, A. Arun, and P. Koehn. 2011. SampleRank training for phrase-based machine translation. *Proceedings of the Sixth Workshop on Statistical Machine Translation. Association for Computational Linguistics*, 261-271.

P. Koehn, H. Hoang, A. Birch, C. Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 177-180.

P. Liang, A. Bouchard-Cote, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. *In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 761-768.

C. Lin and F. Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. *In Proc. of International Conference on Computational Linguistics (COLING)*, No. 501.

F. Och. 2003. Minimum error rate training in statistical machine translation. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, 160-167.

F. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, 295-302.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics (ACL)*, 311-318.

J. Platt. 1998. Sequetial minimal optimization: A fast algorithm for training support vector machines. *In Technical Report MST-TR-98-14. Microsoft Research*.

I. Tsochantaridis, T. Hofman, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. *International Conference on Machine Learning (ICML)*, 823-830.

T. Watanabe. 2012. Optimized online rank learning for machine translation. *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 253-262.

T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki. 2007. Online large-margin training for statistical machine translation. *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 764-773.

Word Level Language Identification in Online Multilingual Communication

Dong Nguyen[1] **A. Seza Doğruöz**[23]
(1) Human Media Interaction, University of Twente, Enschede, The Netherlands
(2) Tilburg School of Humanities, Tilburg University, Tilburg, The Netherlands
(3) Language Technologies Institute, Carnegie Mellon University, Pittsburgh, USA
dong.p.ng@gmail.com, a.s.dogruoz@gmail.com

Abstract

Multilingual speakers switch between languages in online and spoken communication. Analyses of large scale multilingual data require automatic language identification at the word level. For our experiments with multilingual online discussions, we first tag the language of individual words using language models and dictionaries. Secondly, we incorporate context to improve the performance. We achieve an accuracy of 98%. Besides word level accuracy, we use two new metrics to evaluate this task.

1 Introduction

There are more multilingual speakers in the world than monolingual speakers (Auer and Wei, 2007). Multilingual speakers switch across languages in daily communication (Auer, 1999). With the increasing use of social media, multilingual speakers also communicate with each other in online environments (Paolillo, 2011). Data from such resources can be used to study code switching patterns and language preferences in online multilingual conversations. Although most studies on multilingual online communication rely on manual identification of languages in relatively small datasets (Danet and Herring, 2007; Androutsopoulos, 2007), there is a growing demand for automatic language identification in larger datasets. Such a system would also be useful for selecting the right parsers to process multilingual documents and to build language resources for minority languages (King and Abney, 2013).

In this paper, we identify Dutch (NL) en Turkish (TR) at the word level in a large online forum for Turkish-Dutch speakers living in the Netherlands. The users in the forum frequently switch languages within posts, for example:

<TR> *Sariyi ver* </TR>
<NL> *Wel mooi doelpunt* </NL>

So far, language identification has mostly been modeled as a document classification problem. Most approaches rely on character or byte n-grams, by comparing n-gram profiles (Cavnar and Trenkle, 1994), or using various machine learning classifiers. While McNamee (2005) argues that language identification is a solved problem, classification on a more fine-grained level (instead of document level) remains a challenge (Hughes et al., 2006). Furthermore, language identification is more difficult for short texts (Baldwin and Lui, 2010; Vatanen et al., 2010), such as queries and tweets (Bergsma et al., 2012; Carter et al., 2012; Ceylan and Kim, 2009). Tagging individual words (without context) has been done using dictionaries, affix statistics and classifiers using character n-grams (Hammarström, 2007; Gottron and Lipka, 2010). Although Yamaguchi and Tanaka-Ishii (2012) segmented text by language, their data was artificially created by randomly sampling and concatenating text segments (40-160 characters) from monolingual texts. Therefore, the language switches do not reflect realistic switches as they occur in natural texts. Most related to ours is the work by King and Abney (2013) who labeled languages of words in multilingual web pages, but evaluated the task only using word level accuracy.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 857–862,
Seattle, Washington, USA, 18-21 October 2013.

Our paper makes the following contributions: 1) We explore two new ways to evaluate the task for analyzing multilingual communication and show that only word accuracy gives a limited view 2) We are the first to apply this task on a conversational and larger dataset 3) We show that features using the context improve the performance 4) We present a new public dataset to support research on language identification.

In the rest of the paper, we first discuss the related work and describe our dataset. Secondly, we present our experiments. We finally conclude with a summary and suggestions for future work.

2 Corpus

Our data[1] comes from one of the largest online communities in The Netherlands for Turkish-Dutch speakers. All posts from May 2006 until October 2012 were crawled. Although Dutch and Turkish dominate the forum, English fixed phrases (e.g. *no comment, come on*) are also occasionally observed. Users switch between languages within and across posts. Examples 1 and 2 illustrate switches between Dutch and Turkish within the same post. Example 1 is a switch at sentence level, example 2 is a switch at word level.

Example 1:
<NL>Mijn dag kan niet stuk :) </NL>
<TR> Cok guzel bir haber aldim </TR>
Translation: <NL> This made my day:)
</NL><TR> I received good news
</TR>

Example 2:
<TR>kahvalti</TR><NL>met
vriendinnen by my thuis </NL>
Translation: <TR>breakfast </TR>
<NL> with my girlfriends at my home
</NL>

The data is highly informal with misspellings, lengthening of characters (e.g. *hotttt*), replacement of Turkish characters (*kahvalti* instead of *kahvaltı*) and spelling variations (*tankyu* instead of *thank you*). Dutch and Turkish sometimes share common spellings (e.g. *ben* is *am* in Dutch and *I* in Turkish), making this a challenging task.

[1]Available at http://www.dongnguyen.nl/data-langid-emnlp2013.html

Annotation
For this research, we classify words as either Turkish or Dutch. Since Dutch and English are typologically more similar to each other than Turkish, the English phrases (less than 1%) are classified as Dutch. Posts were randomly sampled and annotated by a native Turkish speaker who is also fluent in Dutch. A native Dutch speaker annotated a random set of 100 posts (Cohen's kappa = 0.98). The following tokens were ignored for language identification:

- Smileys (as part of the forum markup, as well as textual smileys such as ":)").
- Numeric tokens and punctuation.
- Forum tags (e.g. *[u]* to underline text).
- Links, images, embedded videos etc.
- Turkish and Dutch first names and place names[2].
- Usernames when indicated with special forum markup.
- Chat words, such as *hahaha*, *ooooh* and *lol* recognized using regular expressions.

Posts for which all tokens are ignored, are not included in the corpus.

Statistics
The dataset was randomly divided into a training, development and test set. The statistics are listed in Table 1. The statistics show that Dutch is the majority language, although the difference between Turkish and Dutch is not large. We also find that the documents (i.e. posts) are short, with on average 18 tokens per document. The data represents realistic texts found in online multilingual communication. Compared to previously used datasets (Yamaguchi and Tanaka-Ishii, 2012; King and Abney, 2013), the data is noisier and the documents are much shorter.

	#NL tokens	**#TR tokens**	**#Posts/(BL%)**
Train	14900 (54%)	12737 (46%)	1603 (15%)
Dev	8590 (51%)	8140 (49%)	728 (19%)
Test	5895 (53%)	5293 (47%)	735 (17%)

Table 1: Number of tokens and posts for Dutch (NL) and Turkish (TR), including % of bilingual (BL) posts

[2]Based on online name lists and Wikipedia pages

3 Experimental Setup

3.1 Training Corpora

We used the following corpora to extract dictionaries and language models.

- *GenCor*: Turkish web pages (Sak et al., 2008).
- *NLCOW2012*: Dutch web pages (Schäfer and Bildhauer, 2012).
- *Blog authorship corpus*: English blogs (Schler et al., 2006).

Each corpus was chunked into large segments which were then selected randomly until 5M tokens were obtained for each language. We tokenized the text and kept the punctuation.

3.2 Baselines

As baselines, we use *langid.py*[3] (Lui and Baldwin, 2012) and van Noord's *TextCat* implementation[4] of the algorithm by Cavnar and Trenkle (1994). TextCat is based on the comparison of n-gram profiles and langid.py on Naive Bayes with n-gram features. For both baselines, words were entered individually to each program. Words for which no language could be determined were assigned to Dutch. These models were developed to identify the languages of the documents instead of words and we did not retrain them. Therefore, these models are not expected to perform well on this task.

3.3 Models

We start with models that assign languages based on only the current word. Next, we explore models and features that can exploit the context (the other words in the post). Words with the highest probability for English were assigned to Dutch for evaluation.

Dictionary lookup (`DICT`)
We extract dictionaries with word frequencies from the training corpora. This approach looks up the words in the dictionaries and chooses the language for which the word has the highest probability. If the word does not occur in the dictionaries, Dutch is chosen as the language.

Language model (`LM`)
We build a character n-gram language model for each language (max. n-gram length is 5). We use Witten-Bell smoothing and include word boundaries for calculating the probabilities.

Dictionary + Language model (`DICT+LM`)
We first use the dictionary lookup approach (`DICT`). If the word does not occur in dictionaries, a decision is made using the language models (`LM`).

Logistic Regression (`LR`)
We use a logistic regression model that incorporates context with the following features:

- (Individual word) Label assigned by the `DICT+LM` model.
- (Context) The results of the `LM` model based on previous + current token, and current token + next token (e.g. the sequence "*ben thuis*" (*am home*) as a whole if *ben* is the current token). This gives the language model more context for estimation. We compare the use of the assigned labels (`LAB`) with the use of the log probability values (`PROB`) as feature values.

Conditional Random Fields (`CRF`)
We treat the task as a sequence labeling problem and experiment with linear-chain Conditional Random Fields (Lafferty et al., 2001) in three settings:

- (Individual word) A CRF with only the tags assigned by the `DICT+LM` to the individual tokens as a feature (`BASE`).
- (Context). CRFs using the `LAB` or `PROB` as additional features (same features as in the logistic regression model) to capture additional context.

3.4 Implementation

Language identification was not performed for texts within quotes. To handle the alphabetical lengthening (e.g. *lollll*), words are normalized by trimming same character sequences of three characters or more. We use the Lingpipe[5] and Scikit-learn (Pedregosa et al., 2011) toolkits for our experiments.

[3] https://github.com/saffsd/langid.py
[4] http://www.let.rug.nl/∼vannoord/TextCat/
[5] http://alias-i.com/lingpipe/

	Word classification					Fraction				Post classification	
	TR		NL				MAE				
Run	P	R	P	R	Acc.	ρ	All	Mono.	BL	F_1	Acc.
Textcat	0.872	0.647	0.743	0.915	0.788	0.739	0.251	0.264	0.188	0.386	0.396
LangIDPy	0.954	0.387	0.641	**0.983**	0.701	0.615	0.364	0.371	0.333	0.413	0.475
DICT	0.955	0.733	0.802	0.969	0.858	0.827	0.196	0.200	0.175	0.511	0.531
LM	0.950	0.930	0.938	0.956	0.944	0.926	0.074	0.076	0.065	0.699	0.703
DICT + LM	0.951	0.934	0.942	0.957	0.946	0.943	0.067	0.067	**0.063**	0.711	0.717
LR + LAB	0.965	0.952	0.958	0.969	0.961	0.917	0.066	0.066	0.068	0.791	0.808
LR + PROB	0.956	0.976	0.978	0.959	0.967	0.945	0.048	0.044	0.064	0.826	0.849
CRF + BASE	**0.973**	0.974	0.977	0.976	0.975	0.940	0.043	0.027	0.119	**0.858**	**0.898**
CRF + LAB	0.964	0.977	0.979	0.967	0.972	0.933	0.046	0.033	0.111	0.855	0.891
CRF + PROB	0.970	**0.980**	**0.982**	0.973	**0.976**	**0.946**	**0.039**	**0.025**	0.103	0.853	0.895

Table 2: Results of language identification experiments.

3.5 Evaluation

The assigned labels can be used for computational analysis of multilingual data in different ways. For example, these labels can be used to analyze language preferences in multilingual communication or the direction of the switches (from Turkish to Dutch or the other way around). Therefore, we evaluate the methods from different perspectives.

The evaluation at word and post levels is done with the following metrics:

- *Word classification* precision (P), recall (R) and accuracy. Although this is the most straightforward approach to evaluate the task, it ignores the document boundaries.
- *Fraction of language in a post*: Pearson's correlation (ρ) and Mean Absolute Error (MAE) of proportion of Turkish in a post. This evaluates the measured proportion of languages in a post when the actual tags for individual words are not needed. For example, such information is useful for analyzing the language preferences of users in the online forum. Besides reporting the MAE over all posts, we also separate the performance over monolingual and bilingual posts (BL).
- *Post classification*: Durham (2003) analyzed the switch between languages in terms of the amount of monolingual and bilingual posts. Our posts are classified as NL, TR or bilingual (BL) if all words are tagged in the particular language or both. We report F_1 and accuracy.

4 Results

The results are presented in Table 2. Significance tests were done by comparing the results of the word and post classification measures using McNemar's test, and comparing the MAEs using paired t-tests. All runs were significantly different from each other based on these tests ($p < 0.05$), except the MAEs of the `DICT+LM` and `LR+LAB` runs and the MAEs and post classification metrics between the CRFs runs.

The difficulty of the task is illustrated by examining the coverage of the tokens by the dictionaries. 24.6% of the tokens (dev + test set) appear in both dictionaries, 31.1% only in the Turkish dictionary, 30.5% only in the Dutch dictionary and 13.9% in none of the dictionaries.

The baselines do not perform well. This confirms that language identification at the word level needs different approaches than identification at the document level. Using language models result in a better performance than dictionaries. They can handle unseen words and are more robust against the noisy spellings. The combination of language models and dictionaries is more effective than the individual models. The results improve when context was added using a logistic regression model, especially with the probability values as feature values.

CRFs improve the results but the improvement on the correlation and MAE is less. More specifically, CRFs improve the performance on monolingual posts, especially when a single word is tagged in the wrong language. However, when the influence of the context is too high, CRFs reduce the performance in bilingual posts.

This is also illustrated with the results of the post classification. The LR+PROB run has a high recall (0.905), but a low precision (0.559) for bilingual posts, while the CRF+PROB approach has a low recall (0.611) and a high precision (0.828).

The fraction of Dutch and Turkish in posts varies widely, providing additional challenges to the use of CRFs for this task. Classifying posts first as monolingual/bilingual and tagging individual words afterwards for bilingual posts might improve the performance.

The evaluation metrics highlight different aspects of the task whereas word level accuracy gives a limited view. We suggest using multiple metrics to evaluate this task for future research.

Dictionaries versus Language Models

The results reported in Table 2 were obtained by sampling 5M tokens of each language. To study the effect of the number of tokens on the performance of the DICT and LM runs, we vary the amount of data. The performance of both methods increases consistently with more data (Figure 1). We also find that language models achieve good performance with only a limited amount of data, and consistently outperform the approach using dictionaries. This is probably due to the highly informal and noisy nature of our data.

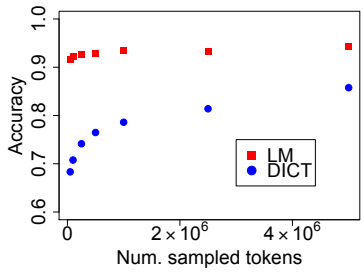


Figure 1: Effect of sampling size

Post classification

We experimented with classifying posts into TR, NL and bilingual using the results of the word level language identification (Table 2: post classification). Posts were classified as a particular language if all words were tagged as belonging to that language, and bilingual otherwise. Runs using CRFs achieved the best performance.

We now experiment with allowing a margin (e.g. a margin of 0.10 classifies posts as TR if at least 90% of the words are classified as TR). Allowing a small margin already increases the results of simpler approaches (such as the LR-PROB run, Table 3) by making it more robust against errors. However, allowing a margin reduces the performance of the CRF runs.

Margin	0.0	0.05	0.10	0.15	0.20
Accuracy	0.849	0.873	0.876	0.878	0.865

Table 3: Effect of margin on post classification (LR-PROB run)

Error analysis

The manual analysis of the results revealed three main challenges: 1) Our data is highly informal with many spelling variations (e.g. *moimoimoi*, *gooooooooooollllll*) and noise (e.g. *asdfghjfgshahaha*) 2) Words sharing spelling in Dutch and Turkish are difficult to identify especially when there is no context available (e.g. a post with only one word). These words are annotated based on their context. For example, the word *super* in *"Seyma, super"* is annotated as Turkish since *Seyma* is also a Turkish word. 3) Named entity recognition is necessary to improve the performance of the system and decrease the noise in evaluation. Based on precompiled lists, our system ignores named entities. However, some names still remain undetected (e.g. usernames).

5 Conclusion

We presented experiments on identifying the language of individual words in multilingual conversational data. Our results reveal that language models are more robust than dictionaries and adding context improves the performance. We evaluate our methods from different perspectives based on how language identification at word level can be used to analyze multilingual data. The highly informal spelling in online environments and the occurrences of named entities pose challenges.

Future work could focus on cases with more than two languages, and languages that are typologically less distinct from each other or dialects (Trieschnigg et al., 2012).

6 Acknowledgements

The first author was supported by the Netherlands Organization for Scientific Research (NWO) grant 640.005.002 (FACT) and the second author through a postdoctoral research grant in E-Humanities (Digital Humanities) by Tilburg University (NL). The authors would like to thank Mariët Theune and Dolf Trieschnigg for feedback.

References

J. Androutsopoulos, 2007. *The multilingual internet. Language, Culture and communication online*, chapter Language choice and code-switching in German-based diasporic web-forums., pages 340–361. Oxford: Oxford University Press.

P. Auer and L. Wei. 2007. Introduction: Multilingualism as a problem? Monolingualism as a problem? In *Handbook of Multilingualism and Multilingual Communication*, volume 5 of *Handbooks of Applied Linguistics*, pages 1–14. Mouton de Gruyter.

P. Auer. 1999. From codeswitching via language mixing to fused lects toward a dynamic typology of bilingual speech. *International Journal of Bilingualism*, 3(4):309–332.

T. Baldwin and M. Lui. 2010. Language identification: the long and the short of the matter. In *Proceedings of NAACL 2010*.

S. Bergsma, P. McNamee, M. Bagdouri, C. Fink, and T. Wilson. 2012. Language identification for creating language-specific twitter collections. In *Proceedings of the Second Workshop on Language in Social Media*.

S. Carter, W. Weerkamp, and M. Tsagkias. 2012. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, pages 1–21.

W.B. Cavnar and J. M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval*.

H. Ceylan and Y. Kim. 2009. Language identification of search engine queries. In *Proceedings of ACL 2009*.

B. Danet and S. C. Herring. 2007. *The multilingual Internet: Language, culture, and communication online*. Oxford University Press Oxford.

M. Durham. 2003. Language choice on a Swiss mailing list. *Journal of Computer-Mediated Communication*, 9(1).

T. Gottron and N. Lipka. 2010. A comparison of language identification approaches on short, query-style texts. In *Proceedings of ECIR 2010*.

H. Hammarström. 2007. A fine-grained model for language identification. In *Proceedings of iNEWS-07 Workshop at SIGIR 2007*.

B. Hughes, T. Baldwin, S. Bird, J. Nicholson, and A. Mackinlay. 2006. Reconsidering language identification for written language resources. In *Proceedings of LREC 2006*.

B. King and S. Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of NAACL 2013*.

J. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*.

M. Lui and T. Baldwin. 2012. langid.py: an off-the-shelf language identification tool. In *Proceedings of ACL 2012*.

P. McNamee. 2005. Language identification: a solved problem suitable for undergraduate instruction. *Journal of Computing Sciences in Colleges*, 20(3):94–101.

J.C. Paolillo. 2011. "Conversational" codeswitching on Usenet and Internet Relay Chat. *Language@Internet*, 8(3).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

H. Sak, T. Güngör, and M. Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *GoTAL 2008*, volume 5221 of *LNCS*, pages 417–427. Springer.

R. Schäfer and F. Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proceedings of LREC 2012*.

J. Schler, M. Koppel, S. Argamon, and J. Pennebaker. 2006. Effects of age and gender on blogging. In *Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*.

D. Trieschnigg, D. Hiemstra, M. Theune, F. Jong, and T. Meder. 2012. An exploration of language identification techniques for the Dutch folktale database. In *Adaptation of Language Resources and Tools for Processing Cultural Heritage workshop (LREC 2012)*.

T. Vatanen, J. J. Väyrynen, and S. Virpioja. 2010. Language identification of short text segments with n-gram models. In *Proceedings of LREC 2010*.

H. Yamaguchi and K. Tanaka-Ishii. 2012. Text segmentation by language using minimum description length. In *Proceedings of ACL 2012*.

Microblog Entity Linking by Leveraging Extra Posts

Yuhang Guo, Bing Qin*, Ting Liu , Sheng Li
Research Center for Social Computing and Information Retrieval
School of Computer Science and Technology
Harbin Institute of Technology, China
{yhguo, bqin*, tliu, sli}@ir.hit.edu.cn

Abstract

Linking name mentions in microblog posts to a knowledge base, namely microblog entity linking, is useful for text mining tasks on microblog. Entity linking in long text has been well studied in previous works. However few work has focused on short text such as microblog post. Microblog posts are short and noisy. Previous method can extract few features from the post context. In this paper we propose to use extra posts for the microblog entity linking task. Experimental results show that our proposed method significantly improves the linking accuracy over traditional methods by 8.3% and 7.5% respectively.

1 Introduction

Microblogging services (e.g. Twitter) are attracting millions of users to share and exchange their ideas and opinions. Millions of new microblog posts are generated on such open broadcasting platforms every day [1]. Microblog provides a fruitful and instant channel of global information publication and acquisition.

A necessary step for the information acquisition on microblog is to identify which entities a post is about. Such identification can be challenging because the entity mention may be ambiguous. Let's begin with a real post from Twitter.

(1) No excuse for floods tax, says Abbott URL

This post is about an Australia political leader, `Tony Abbot`, and his opinion on flood tax policy. To understand that this post mentions Tony Abbot is not trivial because the name Abbot can refer to many people and organizations. In the Wikipedia page of *Abbott*, there lists more than 20 *Abbott*s, such as baseball player `Jim Abbott`, actor `Bud Abbott` and company `Abbott Laboratories`, etc..

Given a knowledge base (KB) (e.g. Wikipedia), entity linking is the task to identify the referent KB entity of a target name mention in plain text. Most current entity linking techniques are designed for long text such as news/blog articles (Mihalcea and Csomai, 2007; Cucerzan, 2007; Milne and Witten, 2008; Han and Sun, 2011; Zhang et al., 2011; Shen et al., 2012; Kulkarni et al., 2009; Ratinov et al., 2011). Entity linking for microblog posts has not been well studied.

Comparing with news/blog articles, microblog posts are:

short each post contains no more than 140 characters;

fresh the new entity-related content may have not been included in the knowledge base;

informal acronyms and spoken language writing style are common.

Due to these properties, few feature can be extracted from a post. Without enough features, previous entity linking methods may fail. In order to overcome the feature sparseness, we turn to another property of microblog:

*Corresponding author

[1] See http://blog.twitter.com/2011/06/ 200-million-tweets-per-day.html.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 863–868,
Seattle, Washington, USA, 18-21 October 2013.

redundancy For each day, over 340M short messages are posted in twitter. Similar information may be posted in different expressions.

For example, we find the following post,

> *(2) Julia Gillard and Tony Abbott on the flood levy just after 8.30am on @612brisbane!*

The content of post (2) is highly related to post (1). In contrast to the confusing post (1), the text in post (2) explicitly indicates that the *Abbott* here refers to the Australian political leader. This inspires us to bridge the confusing post and the knowledge base with other posts.

In this paper, we approach the microblog entity linking by leveraging extra posts. A straightforward method is to expand the post context with similar posts, which we call Context-Expansion-based Microblog Entity Linking (CEMEL). In this method, we first construct a query with the given post and then search for it in a collection of posts. From the search result, we select the most similar posts for the context expansion. The disambiguation will benefit from the extra posts if, hopefully, they are related to the given post in content and include explicit features for the disambiguation.

Furthermore, we propose a Graph-based Microblog Entity Linking (GMEL) method. In contrast to CEMEL, the extra posts in GMEL are not directly added into the context. Instead, they are represented as nodes in a graph, and weighted by their similarity with the target post. We use an iterative algorithm in this graph to propagate the entity weights through the edges between the post nodes.

We conduct experiments on real microblog data which we harvested from Twitter. Current entity linking corpus, such as the TAC-KBP data (McNamee and Dang, 2009), mainly focuses on long text. And few microblog entity linking corpus is publicly available. In this work, we manually annotated a microblog entity linking corpus. This corpus inherit the target names from TAC-KBP2009. So it is comparable with the TAC-KBP2009 corpus.

Experimental results show that the performance of previous methods drops on microblog posts comparing with on long text. Both of CEMEL and GMEL can significantly improve the performance over baselines, which means that entity linking system on microblog can be improved by leveraging extra posts. The results also show that GMEL outperforms CEMEL significantly.

We summarize our contributions as follows.

- We propose a context-expansion-based and a graph-based method for microblog entity linking by leveraging extra posts.
- We annotate a microblog entity linking corpus which is comparable to an existing long text corpus.
- We show the inefficiency of previous method on the microblog corpus and our method can significantly improve the results.

2 Task defination

The microblog entity linking task is that, for a name mention in a microblog post, the system is to find the referent entity of the name in a knowledge base, or return a NIL mark if the entity is absence from the knowledge base. This definition is close to the entity linking task in the TAC-KBP evaluation (Ji and Grishman, 2011) except for the context of the target name is microblog post whereas in TAC-KBP the context is news article or web log.

Several related tasks have been studied on microblog posts. In Meij et al. (2012)'s work, they link a post, rather than a name mention in the post, to relevant Wikipedia concepts. Guo et al. (2013a) and Liu et al. (2013) define entity linking as to first detect all the mentions in a post and then link the mentions to the knowledge base. In contrast, our definition (as well as the TAC-KBP definition) focuses on a concerned name mention across different posts/documents.

3 Method

A typical entity linking system can be broken down into two steps:

candidate generation This step narrows down the candidate entity range from any entity in the world to a limited set.

candidate ranking This step ranks the candidates and output the top ranked entity as the result.

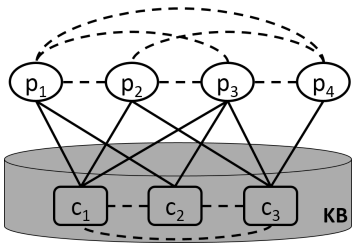


Figure 1: An example of the GMEL graph. $p_1 \dots p_4$ are post nodes and $c_1 \dots c_3$ are candidate entity nodes. Each post node is connected to the corresponding candidate nodes from the knowledge base. The edges between the nodes are weighted by the similarity between them.

In this paper, we use the candidate generation method described in Guo et al.(2013). For the candidate ranking, we use a Vector Space Model (VSM) and a Learning to Rank (LTR) as baselines. VSM is an unsupervised method and LTR is a supervised method. Both of them have achieved the state-of-the-art performances in the TAC-KBP evaluations.

The major challenge in microblog entity linking is the lack of context in the post. An ideal solution is to expand the context with the posts which contain the same entity. However, automatically judging whether a name mention in two documents refers to the same entity, namely cross document co-reference, is not trivial. Here our solution is to rank the posts by their possibility of co-reference to the target one and select the most possible co-referent posts for the expansion.

CEMEL is based on the assumption that, given a name and two posts where the name is mentioned, the higher similarity between the posts the higher possibility of their co-reference and that the co-referent posts may contains useful features for the disambiguation. However, two literally similar posts may not be co-referent. If such non co-referent post is expanded to the context, noises may be included.

Take the following post as an example.

(3) AG Abbott says that bullets have crossed the border from Mexico to Texas at least four times. URL

This post is similar to post (1) because they both contains "says" and "URL". But the Abbott in post (3) refers to the Texas Attorney General Greg Abbott. In this mean, the expanded context in post (3) could mislead the disambiguation for post (1). Such noise can be controlled by setting a strict number of posts to expand the context or weighting the contribution of this post to the target one.

Our CEMEL method consists of the following steps: First we construct a query with the terms from the target post. Second we search for the query in a microblog post collection using a common information retrieval model such as the vector space model. Note that here we limit the searched posts must contain the target name mention. Then we expand the target post with top N similar posts and use a typical entity linking method (such as VSM and LTR) with the expanded context.

Figure 1 illustrates the graph of GMEL. Each node of this graph represents an candidate entity (e.g. $c_1 \dots c_3$) or a post of the given target name (e.g. $p_1 \dots p_4$) In this graph, each node represents an entity or a post of the given target name. Between each pair of post nodes, each pair of entity nodes and each post node and its candidate entity nodes, there is an edge. The edge is weighted by the similarity between the two linked nodes. Entity nodes are labeled by themselves and candidate nodes are initialized as unlabeled nodes. For the edges between post node pairs and entity node pairs, we use cosine similarity. For the edges between a post node and its candidate entity nodes, we use the score given by traditional entity linking methods. We use an iterative algorithm on this graph to propagate the labels from the entity nodes to the post nodes. We adapt Label Propagation (LP) (Zhu and Ghahramani, 2002) and Modified Adsorption (MAD) (Talukdar and Pereira, 2010) for the iteration over the graph.

4 Experiment

4.1 Data Annotation

Till now, few microblog entity linking data is publicly available. In this work, we manually annotate a data set on microblog posts[2]. We collect 15.6 million microblog posts in Twitter dated from January 23 to February 8, 2011. In order to compare with existing entity linking on long text, we select a subset of target names from TAC-KBP2009 and inherit the knowledge base in the TAC-KBP evaluation. The

[2]We published this data so that researchers can reproduce our results.

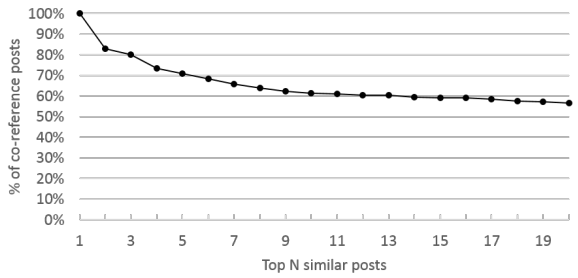


Figure 2: Percentage of the co-reference posts in the top N similar posts

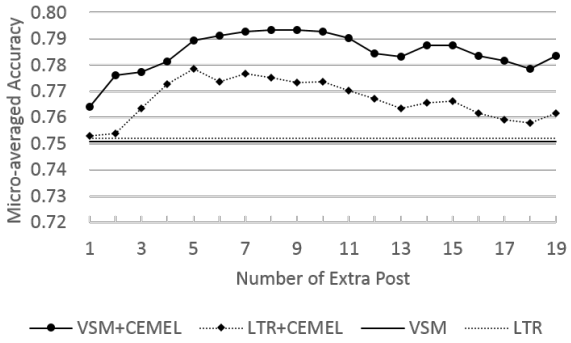


Figure 3: Impact of expansion post number in CEMEL

TAC-KBP2009 data set includes 513 target names. We search for all the target names in the post collection and get 26,643 matches. We randomly sample 120 posts for each of the top 30 most frequently matched target names and filter out non-English and overly short (i.e. less than 3 words) posts. Then we get 2,258 posts for 25 target names and manually link the target name mentions in the posts to the TAC-KBP knowledge base.

In order to evaluate the assumption in CEMEL: similar posts tend to co-reference, we randomly select 10 posts for 5 target names respectively and search for the posts in the post collection. From the search result of each of the 50 posts, we select the top 20 posts and manually annotate if they co-reference with the query post.

4.2 Settings

We generate candidates with the method described in (Guo et al., 2013b) and use Vector Space Model (VSM) (Varma et al., 2009) and Learning to Rank (LTR) (Zheng et al., 2010) as the ranking model. We

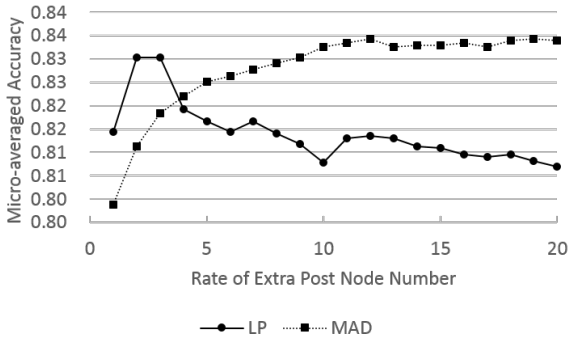


Figure 4: Accuracy of GMEL with different rate of extra post nodes

use Lucene and ListNet with default settings for the VSM and LTR implementation respectively. We use bigram feature for VSM and the feature set of (Chen et al., 2011) for LTR. LTR is evaluated with 10-fold cross validation. Given a target name, the GMEL graph includes all the evaluation posts as well as a set of extra post nodes searched from the post collection with the query of the target name. We filter out determiners, interjections, punctuations, emoticons, discourse markers and URLs in the posts with a twitter part-of-speech tagger (Owoputi et al., 2013). The similarity between a post and its candidate entities is set with the score given by VSM or LTR and the similarity between other nodes is set with the corresponding cosine similarity. We employ junto[3] with default settings for the iterative algorithm implementation .

4.3 Results

Figure 2 shows the relationship between similarity and co-reference. From this figure we can see that the percentage decreases with the growth of N. When the N is up to 10, about 60% of the similar posts co-reference with the query post and the decrease speed slows down. The Pearson correlation coefficient between the percentage and the number of top N is -0.843, which shows a significant correlation between the two variables (with p-value 0.01 under t-test).

Figure 3 shows the impact of the extra post number for the context expansion in CEMEL. We can see that the accuracies of VSM and LTR are improved

[3]See https://github.com/parthatalukdar/junto

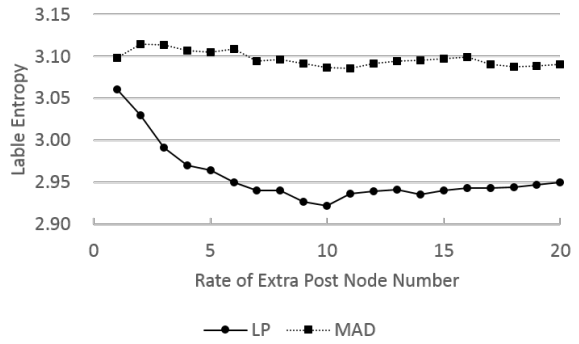


Figure 5: Label entropy of GMEL with different rate of extra post nodes

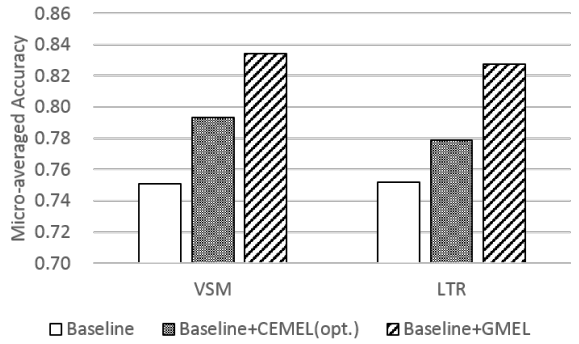


Figure 6: Accuracy of the systems

by CEMEL. The improvements peak with 5-10 extra posts. Then more extra posts will pull down the accuracy.

Figure 4 shows the accuracy of GMEL. The x-axis is the rate of the extra post number over the evaluation post number. We can see that the accuracy of MAD increases with the number of extra post nodes at first and then turns to be stable. The accuracy of LP increases at first and drops when more extra posts are added into the graph.

Figure 5 shows the information entropy of the labels in LP and MAD. The curves show that the prediction of LP tends to converge into a small number of labels. This is because LP prefers smoothing labelings over the graph (Talukdar and Pereira, 2010).

We also evaluate our baselines on TAC-KBP2009 data set (LTR is trained on TAC-KBP2010 data set). The accuracy of VSM and LTR are 0.8338 and 0.8372 respectively, which are comparable with the state-of-the-art result (Hachey et al., 2013).

Figure 6 shows the performances of the systems on the microblog data. We set the optimal expansion post number of CEMEL and use MAD algorithm for GMEL with all searched extra post nodes. From this figure we can see that the results of VSM and LTR baselines are comparable and both of them are significantly lower than that on TAC-KBP2009 data. CEMEL improves the VSM and LTR baselines by 4.3% and 2.7% respectively. GMEL improves VSM and LTR by 8.3% and 7.5% respectively. The results of GMEL are also significantly better than CEMEL. All of the improvements are significant under Z-test with $p < 0.05$.

5 Conclusion

In this paper we approach microblog entity linking by leveraging extra posts. We propose a context-expansion-based and a graph-based method. Experimental results on our data set show that the performance of traditional method drops on the microblog data. The graph-based method outperforms the context-expansion-based method and both of them significantly improve the accuracy of traditional methods. In the graph-based method the modified adsorption algorithm performs better than the label propagation algorithm.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) via grant 61273321, 61073126, 61133012 and the National 863 Leading Technology Research Project via grant 2012AA011102. We would like to thank to Wanxiang Che, Ruiji Fu, Yanyan Zhao, Wei Song and several anonymous reviewers for their constructive comments and suggestions.

References

Zheng Chen, Suzanne Tamang, Adam Lee, and Heng Ji. 2011. A toolkit for knowledge base population. In *SIGIR*, pages 1267–1268.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013a. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1020–1030, Atlanta, Georgia, June. Association for Computational Linguistics.

Yuhang Guo, Bing Qin, Yuqin Li, Ting Liu, and Sheng Li. 2013b. Improving candidate generation for entity linking. In Elisabeth Mtais, Farid Meziane, Mohamad Saraee, Vijayan Sugumaran, and Sunil Vadera, editors, *Natural Language Processing and Information Systems*, volume 7934 of *Lecture Notes in Computer Science*, pages 225–236. Springer Berlin Heidelberg.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194(0):130 – 150. ¡ce:title¿Artificial Intelligence, Wikipedia and Semi-Structured Resources¡/ce:title¿.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 945–954, Portland, Oregon, USA, June. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1148–1158, Portland, Oregon, USA, June. Association for Computational Linguistics.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 457–466, New York, NY, USA. ACM.

Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity linking for tweets. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

P. McNamee and H.T. Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Proceedings of the Second Text Analysis Conference (TAC2009)*.

Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 563–572, New York, NY, USA. ACM.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242, New York, NY, USA. ACM.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 509–518, New York, NY, USA. ACM.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL2013*, pages 380–390, Atlanta, Georgia, June. Association for Computational Linguistics.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384, Portland, Oregon, USA, June. Association for Computational Linguistics.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 449–458, New York, NY, USA. ACM.

Partha Pratim Talukdar and Fernando Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1473–1481, Uppsala, Sweden, July. Association for Computational Linguistics.

Vasudeva Varma, Vijay Bharat, Sudheer Kovelamudi, Praveen Bysani, Santosh GSK, Kiran Kumar N, Kranthi Reddy, Karuna Kumar, and Nitin Maganti. 2009. Iiit hyderabad at tac 2009. In *Proceedings of the Second Text Analysis Conference (TAC 2009)*, Gaithersburg, Maryland, USA, November.

Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. 2011. Entity linking with effective acronym expansion, instance selection, and topic modeling. In Toby Walsh, editor, *IJCAI 2011*, pages 1909–1914.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *NAACL2010*, pages 483–491, Los Angeles, California, June. Association for Computational Linguistics.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University.

Automatic Domain Partitioning for Multi-Domain Learning

Di Wang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
diwang@cs.cmu.edu

Chenyan Xiong
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
cx@cs.cmu.edu

William Yang Wang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ww@cmu.edu

Abstract

Multi-Domain learning (MDL) assumes that the domain labels in the dataset are known. However, when there are multiple metadata attributes available, it is not always straightforward to select a single best attribute for domain partition, and it is possible that combining more than one metadata attributes (including continuous attributes) can lead to better MDL performance. In this work, we propose an automatic domain partitioning approach that aims at providing better domain identities for MDL. We use a supervised clustering approach that learns the domain distance between data instances , and then cluster the data into better domains for MDL. Our experiment on real multi-domain datasets shows that using our automatically generated domain partition improves over popular MDL methods.

1 Introduction

Instead of assuming data are *i.i.d*, Multi-domain learning (MDL) methods assumes that data come from several domains and make use of domain labels to improve modeling performance (Daumé III, 2007). The motivation of using MDL is that datasets from different domains could be different, in two ways. First, the feature distribution $p(x)$ could be domain specific, meaning that the importance of each feature is different across domains. Second, the distribution of label Y given X, $p(y|x)$, of different domains could be different. These differences could create problems for traditional machine learning methods: models learned from one domain might not be generalizable to other domains (Ben-David et al., 2006; Ben-David et al., 2010).

One common assumption of MDL methods is that the domain identities are pre-defined. For example, in the multi-domain Amazon product review dataset (Finkel and Manning, 2009), the product categories are typically used as the domain identities. However, a question raised by Joshi et al. (2012) is that, in real-world data sets, there could be many ways to split data into domains, and it is hard to decide which one to use. Consider the Amazon product reviews, where we have multiple attributes attached to each review: for example, product category, reviewer location, price, and number of feedback. Which attribute is the most informative domain label? Or we should use all of these meta-data and partition the data into many small domains?

In this paper, we investigate the problem of automatic domain partitioning. We propose an empirical domain difference testing method to examine whether two groups of data are *i.i.d*, or generated from different distributions, and how different they are. Using this approach, we generate data pairs that belong to the same distribution, and data pairs that should be partitioned into different domains. These pairs are then used as training data for a supervised clustering algorithm, which automatically partitions the dataset into several domains. In the evaluation, we show that our automatically-partitioned domains improve the performances of two popular MDL methods on real sentiment analysis data sets.

Note that Joshi et al. (2013) proposed a Multi-Attribute Multi-Domain learning (MAMD) method, which also exploited multiple dimensions of meta-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 869–873,
Seattle, Washington, USA, 18-21 October 2013.

data and provided extensions to two traditional MDL methods. However, extensions to the MAMD setting may not be trivial for every MDL algorithm, while our method serves as a pre-processing step and can be easily used for all MDL approaches. In addition to this, MAMD only works with categorical metadata, and can not fully utilize information in the form of continuous metadata values.

2 Automatic Domain Partitioning

In this section, we introduce the Automatic Domain Partitioning (ADP) problem: given data X, metadata M and label Y, find a function $g : M \mapsto I$ such that the common MDL methods perform better with data X and domain identity I in the prediction of Y. For example, on Amazon sentiment analysis data, X is the feature matrix extracted from reviews, Y is the positive or negative label vector, and M is the metadata matrix associated with reviews (e.g. product price and category).

Our approach works as follows: in training, we first use an empirical domain difference testing method to detect whether two groups of data should be considered as different domains; after that we apply supervised clustering to learn the distance metric between two data points, i.e. how different thay are in MDL view, from training data generated by our domain difference test method; finally, based on the distance metric learned, we cluster our data into several clusters, and train MDL models with those clusters as domain labels; in testing, we assign data instance to its nearest cluster and use that cluster as its domain identity, and then apply the trained MDL models for prediction.

2.1 Empirical Domain Difference Test

The key motivation of MDL is that a model fits for one domain may not fit well for other domains. Following the same motivation, we propose an empirical method for domain difference test called Domain Model Loss (DML) that provides us the domain difference score $d(G_1, G_2)$ between two groups of data $G_1 = \{X_1, Y_1\}$ and $G_2 = \{X_2, Y_2\}$.

Domain Model Loss If the mapping functions $f_1 : X_1 \mapsto Y_1$ and $f_2 : X_2 \mapsto Y_2$ are different for two data groups, we could directly use the disagreement of f_1 and f_2 as domain difference score. More specifically, if we train two classifiers $\hat{f}_1 : X_1 \mapsto Y_1, \hat{f}_2 : X_2 \mapsto Y_2$ individually on G_1 and G_2, we could have the K-fold empirical loss:

$$\hat{l}(f_1, G_1) = \frac{1}{K} \sum_i \text{Error of } f_1 \text{ on i-th fold of } G_1,$$

$$\hat{l}(f_2, G_2) = \frac{1}{K} \sum_i \text{Error of } f_2 \text{ on i-th fold of } G_2.$$

And we could also apply the trained model f_1 on G_2, and f_2 on G_1 to get:

$$\hat{l}(f_1, G_2) = \text{Error of } f_1 \text{ on } G_2,$$
$$\hat{l}(f_2, G_1) = \text{Error of } f_2 \text{ on } G_1.$$

Then, if G_1 and G_2 are actually the same with each other, then both models will have same empirical loss on either data set, but if they are not, we will have a positive DML score:

$$DML(G_1, G_2) = \frac{1}{2}(\hat{L}(f_1, G_2) + \hat{L}(f_2, G_1)),$$

where:

$$\hat{L}(f_1, G_2) = \frac{\hat{l}(f_1, G_2) - \hat{l}(f_1, G_1)}{\hat{l}(f_1, G_1)},$$
$$\hat{L}(f_2, G_1) = \frac{\hat{l}(f_2, G_1) - \hat{l}(f_2, G_2)}{\hat{l}(f_2, G_2)}.$$

2.2 Supervised Clustering for Domain Partitioning

Our domain difference test method calculates the distance between two partitioned data groups. However, to directly use it for domain partitioning, we must go through all possible combinations of domain assignments in exponential time, which is infeasible. Our solution is to use a polynomial-time supervised clustering method developed by Xing et al. (2002) to learn a distance function that calculates the distance between any two data points. Formally, given a set of data pairs D, which belong to different domains, and a set of data pairs S, which belong to the same domain, it learns a distance metric A by:

$$\begin{aligned} \max_A g(A) &= \sum_{(i,j)\in D} \sqrt{(m_i - m_j)^T A (m_i - m_j)} \\ s.t. f(A) &= \sum_{(i,j)\in S} (m_i - m_j)^T A (m_i - m_j) \le 1 \\ A &\succeq 0, \end{aligned}$$

where m_i, m_j are meta data of i and j.

The metadata M are preprocessed as follows: 1) Each categorical attribute was converted to several binary questions, one per category, and each binary question was considered as one metadata dimension in ADP method. For example, if categorical attribute "Product Type" has two values "Music" and "Electronics", then there will be two dimensions of metadata corresponding to "Product Type" in ADP. Two metadata dimensions correspond to binary questions: "Is Product Type Music" and "Is Product Type Electronics". 2) Each continuous attribute was normalized by scaling between 0 and 1.

The training data S, D for metric learning are generated as follows:

1. For each dimension M_k of M, split data at value 0.5, sample two equally sized groups, apply our domain difference testing method and find the difference between these data groups.
2. Assign distance to each pair of instances by the average distance of all partitions that partitions the pair into different groups.
3. Select top n similar pairs as S and top n different pairs as D.

The learned distance metric A now conveys the domain difference information obtained from our domain distance test results: which meta attributes are important for domain partitioning and which are not as important. Following Xing et al. (2002), we transfer the instance's metadata feature M by MB^T, where $B^TB = A$. Then we use a clustering method on MB^T, and the output is our domain partitioning result.

3 Experiment Methodology

Datasets To evaluate our methods, we used two subsets of Amazon review corpus (Jindal and Liu, 2008), which originally contain 5.8 million reviews with a variety of metadata about products and users. The first subset (`BOOK`) contains 20,000 reviews on books published by eleven most popular publishers, while the second (`PROD`) is reviews about products within seven most common product categories. We randomly split each dataset into training and testing sets with equal size. The task is to predict a positive or negative label for each review. Case insensitive unigrams excluding stop words are used as features, and all features appear less than 500 times are removed for efficient experiment processing. Reviews of 4 or 5 stars are considered positive and 1 or 2 stars are considered negative, while 3 stars reviews are excluded. Each review has multiple metadata such as book's publisher, product's type, user's state location, product price, review year, and number of other user feedback. Reviews with missing metadata are filtered out.

MDL Methods Our first MDL algorithm is the Frustratingly Easy Domain Adaptation (**FEDA**) (Daumé III, 2007) which is easy to implement and achieved competitive performance on many applications. It creates an augmented feature space as the Cartesian product of the input features and the original domains plus a shared domain. Then it uses a SVM classifier over the augmented feature space to obtain classification result. Specifically, our FEDA methods use L_2-regularized SVM with linear kernel by LIBLINEAR package[1]. The parameters C = 0.01 was selected using five-fold cross-validation on training set.

Our second MDL algorithm is Multi-Domain Regularization (**MDR**) (Dredze and Crammer, 2008), which is a classifier combination approach based on Confidence-Weighted (CW) learning (Dredze et al., 2008). The CW learning is an online update method that maintains probabilistic confidence for each parameter by keeping track of its variance. In our experiments, we use the CW implementation provided by its authors and choose the best performing configurations described in (Dredze and Crammer, 2008).

Domain Partition Methods We evaluated the domain partition results provided by our ADP on the two MDL methods (FEDA & MDR). For simplicity and efficiency, we use Naive Bayes as our base prediction model f_1 and f_2 to generate the domain model loss score, described in section 2.1. In training data generation, we choose top 10% similar pairs as S and top 10% different pairs as D. And given the learnt distance metric A, we use K-means to do the clustering. The number of clusters is selected by five-fold cross-validation on training set.

[1] http://www.csie.ntu.edu.tw/∼cjlin/liblinear

We compare our domain partition quality with three other methods: 1) **1-Best** chooses best performing categorical metadata on a validation set as domain indicators, where the original training set was splitted equally to train and validate the performance of each categorical attribute; 2) **Random** partition that assigns domain identities to instances randomly with same number of domains as 1-Best. We run each random partition ten times and took the average; 3) **MAMD** proposed by Joshi et al. (2013). However, the original version of MAMD does not support continuous attribute such as price. So we made an extension that sorts these values to ten bins and then treats them as categorical values.

4 Results and Discussions

Partition + MDL	PROD	BOOK
ADP + FEDA	82.02 $*$	86.22 ‡$*$
MAMD + FEDA	81.04	86.08
1-Best + FEDA	82.00	85.85
Random + FEDA	79.36	84.72
ADP + MDR	82.10 ♯ ‡ $*$	86.62 ♯ ‡ $*$
MAMD + MDR	80.17	84.37
1-Best + MDR	79.79	83.68
Random + MDR	74.65	81.16

Table 1: Overall accuracies on PROD and BOOK datasets. ADP results that are statistically significantly better than MAMD are marked with ♯, and better than 1-Best and Random are indicated by ‡ and $*$ respectively, using a paired t-test, with $p < 0.05$.

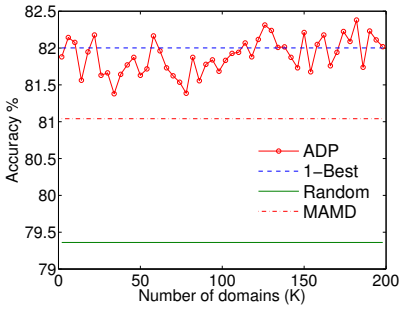
Table 1 shows the overall experimental results of four domain partition methods with two MDL methods on PROD and BOOK datasets. One could see that when using MDR method, ADP could significantly outperform all baselines on both data sets, with relatively more than 2% gains. For FEDA, on PROD data, ADP performs the same with MAMD and 1-Best; on BOOK data, ADP outperforms 1-Best significantly, but is just slightly better than MAMD. One possible reason is that the best numbers of cluster selected by cross-validation are around 150. With such large number of none-perfect domains, FEDA will generate huge dimension of features and perhaps require more training data to provide better performances. Another possible reason is that FEDA and the SVM underlying FEDA are very robust against bad domain partition results. This might be the reason of high FEDA baselines. In general, our ADP method helps existing MDL approaches achieve better performance, while bad (Random) partitioning does hurt.

Figure 1(a) and 1(b) shows the performances of applying FEDA on different domain partitioning methods on PROD and BOOK, while Figure 1(c) and 1(d) shows experiment results with MDR. The x-axis is the size of the output domains (the K in our K-means clustering), and y-axis is the accuracy of models. With our domain partitioning approach, MDR can perform consistently higher than all the three baselines on both dataset when $k > 50$. As we discussed for Table 1, FEDA might be less sensitive to domain partition results, which causes high baseline performance and high ADP+FEDA performance with small K. Since the performance trends to increase along with k until 50 in three figures (1(b), 1(c) and 1(d)), we believe that the ground-truth domain size is likely larger than 50. These results clearly indicate ADP does provide more desirable domain assignments for MDL. The domain selected by 1-Best such as publishers has only 11 domains, which limits the ability of 1-Best to completely express domain information. And our generated domains integrate multiple metadata attributes, lead to more detailed domain partitions, and enhance the ability of MDL methods to capture the difference between different groups of data. Although accuracies are growing with k in general, we also see that there are fluctuations on curves especially when curves are zoomed to a small range. To get smoother results, we can sample more data to calculate domain similarity and repeat the K-means clustering with more different initializations.

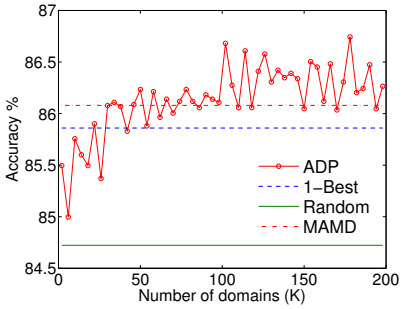
5 Conclusions

In this paper, we propose an Automatic Domain Partition (ADP) method that provides better domain identities for multi-domain learning methods. We first propose a new approach to identify whether two data groups should be considered as different domains, by comparing the differences using Domain Model Loss. We use a supervised clustering approach to train our model with labels generated by domain difference tests, and cluster the re-weighted

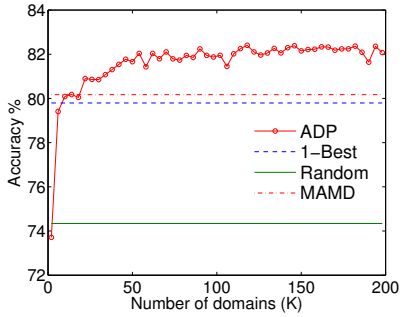
metadata as our domain partition by K-means. Experiments on real world multi-domain data show that the domain identities generated by our method can improve the performance of MDL models.



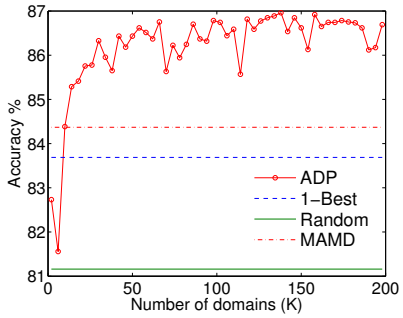
(a) FEDA results on PROD



(b) FEDA results on BOOK



(c) MDR results on PROD



(d) MDR results on BOOK

Figure 1: Accuracies over different size of the output domains (K)

References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 137–144.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 689–697.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML)*, pages 264–271.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 602–610.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Web Data Mining (WSDM)*, pages 219–230.

Mahesh Joshi, Mark Dredze, William W. Cohen, and Carolyn Penstein Rosé. 2012. Multi-domain learning: When do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, (EMNLP-CoNLL)*, pages 1302–1312.

Mahesh Joshi, Mark Dredze, William W. Cohen, and Carolyn P. Rosé. 2013. Whats in a domain? multi-domain learning for multi-attribute data. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 685–690, Atlanta, Georgia, June. Association for Computational Linguistics.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. 2002. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*, pages 505–512.

Decipherment with a Million Random Restarts

Taylor Berg-Kirkpatrick **Dan Klein**
Computer Science Division
University of California, Berkeley
{tberg,klein}@cs.berkeley.edu

Abstract

This paper investigates the utility and effect of running numerous random restarts when using EM to attack decipherment problems. We find that simple decipherment models are able to crack homophonic substitution ciphers with high accuracy if a large number of random restarts are used but almost completely fail with only a few random restarts. For particularly difficult homophonic ciphers, we find that big gains in accuracy are to be had by running upwards of 100K random restarts, which we accomplish efficiently using a GPU-based parallel implementation. We run a series of experiments using millions of random restarts in order to investigate other empirical properties of decipherment problems, including the famously uncracked Zodiac 340.

1 Introduction

What can a million restarts do for decipherment? EM frequently gets stuck in local optima, so running between ten and a hundred random restarts is common practice (Knight et al., 2006; Ravi and Knight, 2011; Berg-Kirkpatrick and Klein, 2011). But, how important are random restarts and how many random restarts does it take to saturate gains in accuracy?

We find that the answer depends on the cipher. We look at both Zodiac 408, a famous homophonic substitution cipher, and a more difficult homophonic cipher constructed to match properties of the famously unsolved Zodiac 340. Gains in accuracy saturate after only a hundred random restarts for Zodiac 408, but for the constructed cipher we see large gains in accuracy even as we scale the number of random restarts up into the hundred thousands. In both cases the difference between few and many random restarts is the difference between almost complete failure and successful decipherment.

We also find that millions of random restarts can be helpful for performing exploratory analysis. We look at some empirical properties of decipherment problems, visualizing the distribution of local optima encountered by EM both in a successful decipherment of a homophonic cipher and in an unsuccessful attempt to decipher Zodiac 340. Finally, we attack a series of ciphers generated to match properties of Zodiac 340 and use the results to argue that Zodiac 340 is likely not a homophonic cipher under the commonly assumed linearization order.

2 Decipherment Model

Various types of ciphers have been tackled by the NLP community with great success (Knight et al., 2006; Snyder et al., 2010; Ravi and Knight, 2011). Many of these approaches learn an encryption key by maximizing the score of the decrypted message under a language model. We focus on homophonic substitution ciphers, where the encryption key is a 1-to-many mapping from a plaintext alphabet to a cipher alphabet. We use a simple method introduced by Knight et al. (2006): the EM algorithm (Dempster et al., 1977) is used to learn the emission parameters of an HMM that has a character trigram language model as a backbone and the ciphertext as the observed sequence of emissions. This means that we learn a multinomial over cipher symbols for each plaintext character, but do not learn transition

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 874–878,
Seattle, Washington, USA, 18-21 October 2013.

parameters, which are fixed by the language model. We predict the deciphered text using posterior decoding in the learned HMM.

2.1 Implementation

Running multiple random restarts means running EM to convergence multiple times, which can be computationally intensive; luckily, restarts can be run in parallel. This kind of parallelism is a good fit for the Same Instruction Multiple Thread (SIMT) hardware paradigm implemented by modern GPUs. We implemented EM with parallel random restarts using the CUDA API (Nickolls et al., 2008). With a GPU workstation,[1] we can complete a million random restarts roughly a thousand times more quickly than we can complete the same computation with a serial implementation on a CPU.

3 Experiments

We ran experiments on several homophonic substitution ciphers: some produced by the infamous Zodiac killer and others that were automatically generated to be similar to the Zodiac ciphers. In each of these experiments, we ran numerous random restarts; and in all cases we chose the random restart that attained the highest model score in order to produce the final decode.

3.1 Experimental Setup

The specifics of how random restarts are produced is usually considered a detail; however, in this work it is important to describe the process precisely. In order to generate random restarts, we sampled emission parameters by drawing uniformly at random from the interval $[0, 1]$ and then normalizing. The corresponding distribution on the multinomial emission parameters is mildly concentrated at the center of the simplex.[2]

For each random restart, we ran EM for 200 iterations.[3] We found that smoothing EM was important for good performance. We added a smoothing constant of 0.1 to the expected emission counts before each M-step. We tuned this value on a small held out set of automatically generated ciphers.

In all experiments we used a trigram character language model that was linearly interpolated from character unigram, bigram, and trigram counts extracted from both the Google N-gram dataset (Brants and Franz, 2006) and a small corpus (about 2K words) of plaintext messages authored by the Zodiac killer.[4]

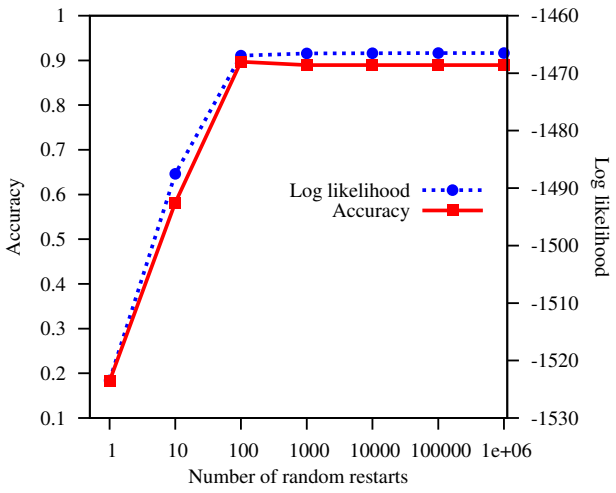


Figure 1: Zodiac 408 cipher. Accuracy by best model score and best model score vs. number of random restarts. Bootstrapped from 1M random restarts.

3.2 An Easy Cipher: Zodiac 408

Zodiac 408 is a homophonic cipher that is 408 characters long and contains 54 different cipher symbols. Produced by the Zodiac killer, this cipher was solved, manually, by two amateur code-breakers a week after its release to the public in 1969. Ravi and Knight (2011) were the first to crack Zodiac 408 using completely automatic methods.

In our first experiment, we compare a decode of Zodiac 408 using one random restart to a decode using 100 random restarts. Random restarts have high

[1] We used a single workstation with three NVIDIA GTX 580 GPUs. These are consumer graphics cards introduced in 2011.

[2] We also ran experiments where emission parameters were drawn from Dirichlet distributions with various concentration parameter settings. We noticed little effect so long as the distribution did not favor the corners of the simplex. If the distribution did favor the corners of the simplex, decipherment results deteriorated sharply.

[3] While this does not guarantee convergence, in practice 200 iterations seems to be sufficient for the problems we looked at.

[4] The interpolation between n-gram orders is uniform, and the interpolation between corpora favors the Zodiac corpus with weight 0.9.

variance, so when we present the accuracy corresponding to a given number of restarts we present an average over many bootstrap samples, drawn from a set of one million random restarts. If we attack Zodiac 408 with a single random restart, on average we achieve an accuracy of 18%. If we instead use 100 random restarts we achieve a much better average accuracy of 90%. The accuracies for various numbers of random restarts are plotted in Figure 1. Based on these results, we expect accuracy to increase by about 72% when using 100 random restarts instead of a single random restart; however, using more than 100 random restarts for this particular cipher does not appear to be useful.

Also in Figure 1, we plot a related graph, this time showing the effect that random restarts have on the achieved model score. By construction, the (maximum) model score must increase as we increase the number of random restarts. We see that it quickly saturates in the same way that accuracy did.

This raises the question: have we actually achieved the globally optimal model score or have we only saturated the usefulness of random restarts? We can't prove that we have achieved the global optimum,[5] but we can at least check that we have surpassed the model score achieved by EM when it is initialized with the gold encryption key. On Zodiac 408, if we initialize with the gold key, EM finds a local optimum with a model score of -1467.4. The best model score over 1M random restarts is -1466.5, which means we have surpassed the gold initialization.

The accuracy after gold initialization was 92%, while the accuracy of the best local optimum was only 89%. This suggests that the global optimum may not be worth finding if we haven't already found it. From Figure 1, it appears that large increases in likelihood are correlated with increases in accuracy, but small improvements to high likelihoods (e.g. the best local optimum versus the gold initialization) may not to be.

[5]ILP solvers can be used to globally optimize objectives corresponding to short 1-to-1 substitution ciphers (Ravi and Knight, 2008) (though these objectives are slightly different from the likelihood objectives faced by EM), but we find that ILP encodings for even the shortest homophonic ciphers cannot be optimized in any reasonable amount of time.

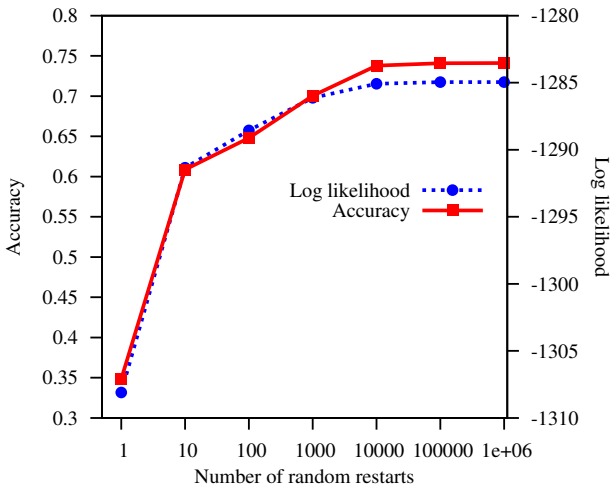


Figure 2: Synth 340 cipher. Accuracy by best model score and best model score vs. number of random restarts. Bootstrapped from 1M random restarts.

3.3 A Hard Cipher: Synth 340

What do these graphs look like for a harder cipher? Zodiac 340 is the second cipher released by the Zodiac killer, and it remains unsolved to this day. However, it is unknown whether Zodiac 340 is actually a homophonic cipher. If it were a homophonic cipher we would certainly expect it to be harder than Zodiac 408 because Zodiac 340 is shorter (only 340 characters long) and at the same time has *more* cipher symbols: 63. For our next experiment we generate a cipher, which we call Synth 340, to match properties of Zodiac 340; later we will generate multiple such ciphers.

We sample a random consecutive sequence of 340 characters from our small Zodiac corpus and use this as our message (and, of course, remove this sequence from our language model training data). We then generate an encryption key by assigning each of 63 cipher symbols to a single plain text character so that the number of cipher symbols mapped to each plaintext character is proportional to the frequency of that character in the message (this balancing makes the cipher more difficult). Finally, we generate the actual ciphertext by randomly sampling a cipher token for each plain text token uniformly at random from the cipher symbols allowed for that token under our generated key.

In Figure 2, we display the same type of plot, this time for Synth 340. For this cipher, there is an abso-

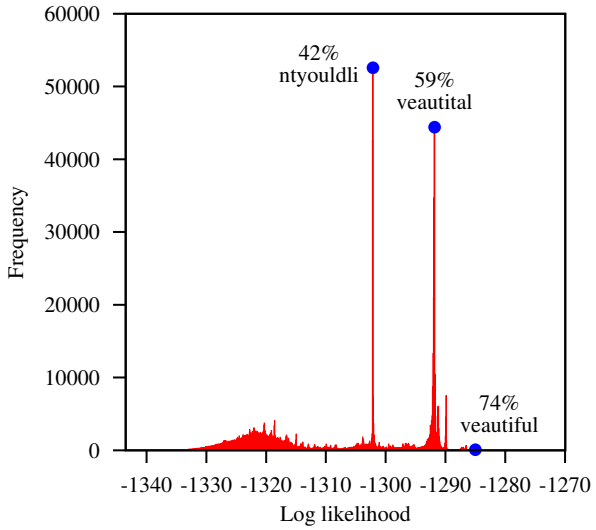


Figure 3: Synth 340 cipher. Histogram of the likelihoods of the local optima encountered by EM across 1M random restarts. Several peaks are labeled with their average accuracy and a snippet of a decode. The gold snippet is "beautiful."

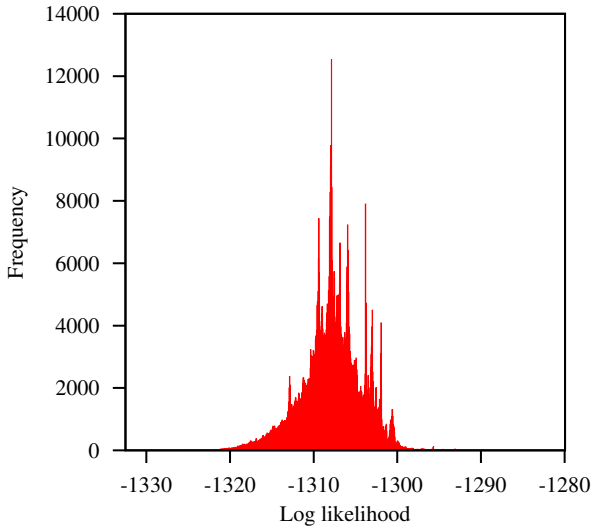


Figure 4: Zodiac 340 cipher. Histogram of the likelihoods of the local optima encountered by EM across 1M random restarts.

lute gain in accuracy of about 9% between 100 random restarts and 100K random restarts. A similarly large gain is seen for model score as we scale up the number of restarts. This means that, even after tens of thousands of random restarts, EM is still finding new local optima with better likelihoods. It also appears that, even for a short cipher like Synth 340, likelihood and accuracy are reasonably coupled.

We can visualize the distribution of local optima encountered by EM across 1M random restarts by plotting a histogram. Figure 3 shows, for each range of likelihood, the number of random restarts that led to a local optimum with a model score in that range. It is quickly visible that a few model scores are substantially more likely than all the rest. This kind of sparsity might be expected if there were a small number of local optima that EM was extremely likely to find. We can check whether the peaks of this histogram each correspond to a single local optimum or whether each is composed of multiple local optima that happen to have the same likelihood. For the histogram bucket corresponding to a particular peak, we compute the average relative difference between each multinomial parameter and its mean. The average relative difference for the highest peak in Figure 3 is 0.8%, and for the second highest peak is 0.3%. These values are much smaller than the average relative difference between the means of these two peaks, 40%, indicating that the peaks do correspond to single local optima or collections of extremely similar local optima.

There are several very small peaks that have the highest model scores (the peak with the highest model score has a frequency of 90 which is too small to be visible in Figure 3). The fact that these model scores are both high and rare is the reason we continue to see improvements to both accuracy and model score as we run numerous random restarts. The two tallest peaks and the peak with highest model score are labeled with their average accuracy and a small snippet of a decode in Figure 3. The gold snippet is the word "beautiful."

3.4 An Unsolved Cipher: Zodiac 340

In a final experiment, we look at the Zodiac 340 cipher. As mentioned, this cipher has never been cracked and may not be a homphonic cipher or even a valid cipher of any kind. The reading order of the cipher, which consists of a grid of symbols, is unknown. We make two arguments supporting the claim that Zodiac 340 is not a homophonic cipher with row-major reading order: the first is statistical, based on the success rate of attempts to crack similar synthetic ciphers; the second is qualitative, comparing distributions of local optimum likelihoods.

If Zodiac 340 is a homophonic cipher should we

expect to crack it? In order to answer this question we generate 100 more ciphers in the same way we generated Synth 340. We use 10K random restarts to attack each cipher, and compute accuracies by best model score. The average accuracy across these 100 ciphers was 75% and the minimum accuracy was 36%. All but two of the ciphers were deciphered with more than 51% accuracy, which is usually sufficient for a human to identify a decode as partially correct.

We attempted to crack Zodiac 340 using a row-major reading order and 1M random restarts, but the decode with best model score was nonsensical. This outcome would be unlikely if Zodiac 340 were like our synthetic ciphers, so Zodiac 340 is probably not a homophonic cipher with a row-major order. Of course, it could be a homophonic cipher with a different reading order. It could also be the case that a large number of salt tokens were inserted, or that some other assumption is incorrect.

In Figure 4, we show the histogram of model scores for the attempt to crack Zodiac 340. We note that this histogram is strikingly different from the histogram for Synth 340. Zodiac 340's histogram is not as sparse, and the range of model scores is much smaller. The sparsity of Synth 340's histogram (but not Zodiac 340's histogram) is typical of histograms corresponding to our set of 100 generated ciphers.

4 Conclusion

Random restarts, often considered a footnote of experimental design, can indeed be useful on scales beyond that generally used in past work. In particular, we found that the initializations that lead to the local optima with highest likelihoods are sometimes very rare, but finding them can be worthwhile; for the problems we looked at, local optima with high likelihoods also achieved high accuracies. While the present experiments are on a very specific unsupervised learning problem, it is certainly reasonable to think that large-scale random restarts have potential more broadly.

In addition to improving search, large-scale restarts can also provide a novel perspective when performing exploratory analysis, here letting us argue in support for the hypothesis that Zodiac 340 is not a row-major homophonic cipher.

References

Taylor Berg-Kirkpatrick and Dan Klein. 2011. Simple effective decipherment via combinatorial optimization. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. Linguistic Data Consortium, Catalog Number LDC2009T25.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the 2006 Annual Meeting of the Association for Computational Linguistics*.

John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. 2008. Scalable parallel programming with CUDA. *Queue*.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

Sujith Ravi and Kevin Knight. 2011. Bayesian inference for Zodiac and other homophonic ciphers. In *Proceedings of the 2011 Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proceedings of the 2010 Annual Meeting of the Association for Computational Linguistics*.

Russian Stress Prediction using Maximum Entropy Ranking

Keith Hall **Richard Sproat**
Google, Inc
New York, NY, USA
{kbhall,rws}@google.com

Abstract

We explore a model of stress prediction in Russian using a combination of local contextual features and linguistically-motivated features associated with the word's stem and suffix. We frame this as a ranking problem, where the objective is to rank the pronunciation with the correct stress above those with incorrect stress. We train our models using a simple Maximum Entropy ranking framework allowing for efficient prediction. An empirical evaluation shows that a model combining the local contextual features and the linguistically-motivated non-local features performs best in identifying both primary and secondary stress.

1 Introduction

In many languages, one component of accurate word pronunciation prediction is predicting the placement of lexical stress. While in some languages (e.g. Spanish) the lexical stress system is relatively simple, in others (e.g. English, Russian) stress prediction is quite complicated. Much as with other work on pronunciation prediction, previous work on stress assignment has fallen into two camps, namely systems based on linguistically motivated rules (Church, 1985, for example) and more recently data-driven techniques where the models are derived directly from labeled training data (Dou et al., 2009). In this work, we present a machine-learned system for predicting Russian stress which incorporates both data-driven contextual features as well as linguistically-motivated word features.

2 Previous Work on Stress Prediction

Pronunciation prediction, of which stress prediction is a part, is important for many speech applications including automatic speech recognition, text-to-speech synthesis, and transliteration for, say, machine translation. While there is by now a sizable literature on pronunciation prediction from spelling (often termed "grapheme-to-phoneme" conversion), work that specifically focuses on stress prediction is more limited. One of the best-known early pieces of work is (Church, 1985), which uses morphological rules and stress pattern templates to predict stress in novel words. Another early piece of work is (Williams, 1987).

The work we present here is closer in spirit to data-driven approaches such as (Webster, 2004; Pearson et al., 2000) and particularly (Dou et al., 2009), whose features we use in the work described below.

3 Russian Stress Patterns

Russian stress preserves many features of Indo-European accenting patterns (Halle, 1997). In order to know the stress of a morphologically complex word consisting of a stem plus a suffix, one needs to know if the stem has an accent, and if so on what syllable; and similarly for the suffix. For words where the stem is accented,

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 879–883,
Seattle, Washington, USA, 18-21 October 2013.

	acc	unacc	postacc
Dat Sg	гор'оху	г'ороду	корол'ю
	gor'oxu	g'orodu	korolj'u
Dat Pl	гор'охам	город'ам	корол'ям
	gor'oxam	gorod'am	korolj'am
	'pea'	'town'	'king'

Table 1: Examples of accented, unaccented and postaccented nouns in Russian, for dative singular and plural forms.

this accent overrides any accent that may occur on the suffix. With unaccented stems, if the suffix has an accent, then stress for the whole word will be on the suffix; if there is also no stress on the suffix, then a default rule places stress on the first syllable of the word. In addition to these patterns, there are also postaccented words, where accent is placed uniformly on the first syllable of the suffix — an innovation of East and South Slavic languages (Halle, 1997). These latter cases can be handled by assigning an accent to the stem, indicating that it is associated with the syllable *after* the stem. Some examples of each of these classes, from (Halle, 1997, example 11), are given in Table 1. According to Halle (1997), considering just nouns, 91.6% are accented (on the stem), 6.6% are postaccented and 0.8% are unaccented, with about 1.0% falling into other patterns.

Stress placement in Russian is important for speech applications since over and above the phonetic effects of stress itself (prominence, duration, etc.), the position of stress strongly influences vowel quality. To take an example of the lexically unaccented noun город *gorod* 'city', the genitive singular г'орода *g'oroda* /gˈɔrədə/ contrasts with the nominative plural город'а *gorod'a* /gərʌdˈa/. All non-stressed /a/ are reduced to schwa — or by most accounts if before the stressed syllable to /ʌ/; see (Wade, 1992).

The stress patterns of Russian suggest that useful features for predicting stress might include (string) prefix and suffix features of the word in order to capture properties of the stem, since some stems are (un)accented, or of the suffix, since some suffixes are accented.

4 Maximum Entropy Rankers

Similarly to Dou et al. (2009), we frame the stress prediction problem as a ranking problem. For each word, we identify stressable vowels and generate a set of alternatives, each representing a different primary stress placement. Some words also have secondary stress which, if it occurs, always occurs before the primary stressed syllable. For each primary stress alternative, we generate all possible secondary stressed alternatives, including an alternative that has no secondary stress. (In the experiments reported below we actually consider two conditions: one where we ignore secondary stress in training and evaluation; and one where we include it.)

Formally, we model the problem using a Maximum Entropy ranking framework similar to that presented in Collins and Koo (2005). For each example, x_i, we generate the set of possible stress patterns $\mathcal{Y}_i$. Our goal is to rank the items in $\mathcal{Y}_i$ such that all of the valid stress patterns $\mathcal{Y}_i^*$ are above all of the invalid stress patterns. Our objective function is the likelihood, $\mathcal{L}$ of this conditional distribution:

$$\mathcal{L} = \prod_i p(\mathcal{Y}_i^* | \mathcal{Y}_i, x_i) \quad (1)$$

$$\log \mathcal{L} = \sum_i \log p(\mathcal{Y}_i^* | \mathcal{Y}_i, x_i) \quad (2)$$

$$= \sum_i \log \frac{\sum_{y' \in \mathcal{Y}_i^*} e^{\sum_k \theta_k f_k(y', x)}}{Z} \quad (3)$$

Z is defined as the sum of the conditional likelihood over all hypothesized stress predictions for example x_i:

$$Z = \sum_{y'' \in \mathcal{Y}_i} e^{\sum_k \theta_k f_k(y'', x)} \quad (4)$$

The objective function in Equation 3 can be optimized using a gradient-based optimization. In our case, we use a variety of stochastic gradient descent (SGD) which can be parallelized for efficient training.

During training, we provide all plausibly correct primary stress patterns as the *positive* set

$\mathcal{Y}_i^*$. At prediction-time, we evaluate all possible stress predictions and pick the one with the highest score under the trained model Θ:

$$\arg\max_{y' \in \mathcal{Y}_i} p(y'|\mathcal{Y}_i) = \arg\max_{y' \in \mathcal{Y}_i} \sum_k \theta_k f_k(y', x) \quad (5)$$

The primary motivation for using Maximum Entropy rather the ranking-SVM is for efficient training and inference. Under the above Maximum Entropy model, we apply a linear model to each hypothesis (i.e., we compute the dot-product) and sort according to this score. This makes inference (prediction) fast in comparison to the ranking SVM-based approach proposed in Dou et al. (2009).

All experiments presented in this paper used the Iterative Parameter Mixtures distributed SGD training optimizer (Hall et al., 2010). Under this training approach, per-iteration averaging has a regularization-like effect for sparse feature spaces. We also experimented with L1-regularization, but it offered no additional improvements.

5 Features

The features used in (Dou et al., 2009) are based on trigrams consisting of a vowel letter, the preceding consonant letter (if any) and the following consonant letter (if any). Attached to each trigram is the stress level of the trigram's vowel — 1, 2 or 0 (for no stress). For the English word *overdo* with the stress pattern 2-0-1, the basic features would be *ov:2*, *ver:0*, and *do:1*. Notating these pairs as $s_i : t_i$, where s_i is the triple, t_i is the stress pattern and i is the position in the word, the complete feature set is given in Table 2, where the stress pattern for the whole word is given in the last row as $t_1 t_2 ... t_N$. Dou and colleagues use an SVM-based ranking approach, so they generated features for all possible stress assignments for each word, assigning the highest rank to the correct assignment. The ranker was then trained to associate feature combinations to the correct ranking of alternative stress possibilities.

Given the discussion in Section 3, plausible additional features are all prefixes and suffixes of the word, which might be expected to better capture some of the properties of Russian stress patterns discussed above, than the much more local features from (Dou et al., 2009). In this case for all stress variants of the word we collect prefixes of length 1 through the length of the word, and similarly for suffixes, except that for the stress symbol we treat that together with the vowel it marks as a single symbol. Thus for the word *gorod'a*, all prefixes of the word would be *g*, *go*, *gor*, *goro*, *gorod*, *gorod'a*.

In addition, we include prefixes and suffixes of an "abstract" version of the word where most consonants and vowels have been replaced by a phonetic class. The mappings for these are shown in Table 3.

Note that in Russian the vowel ё /jɔ/ is *always* stressed, but is rarely written in text: it is usually spelled as е, whose stressed pronunciation is /(j)ɛ/. Since written е is in general ambiguous between е and ё, when we compute stress variants of a word for the purpose of rank-

Substring	s_i, t_i
	s_i, i, t_i
Context	s_{i1}, t_i
	$s_{i1}s_i, t_i$
	s_{i+1}, t_i
	$s_i s_{i+1}, t_i$
	$s_{i1}s_i s_{i+1}, ti$
Stress Pattern	$t_1 t_2 ... t_N$

Table 2: Features used in (Dou et al., 2009, Table 2).

vowel	а,е,и,о,у,э,ю,я,ы
stop	б,д,г,п,т,к
nasal	м,н
fricative	ф,с,ш,щ,х,з,ж
hard/soft	ъ,ь
yo	ё
semivowel	й,в
liquid	р,л
affricate	ц,ч

Table 3: Abstract phonetic classes used for constructing "abstract" versions of a word. Note that etymologically, and in some ways phonologically, в v behaves like a semivowel in Russian.

ing, we include both variants that have е and ё.

6 Data

Our data were 2,004,044 fully inflected words with assigned stress expanded from Zaliznyak's *Grammatical Dictionary of the Russian Language* (Zaliznyak, 1977). These were split randomly into 1,904,044 training examples and 100,000 test examples. The 100,000 test examples obviously contain no *forms* that were found in the training data, but most of them are word forms that derive from lemmata from which some training data forms are also derived. Given the fact that Russian stress is lexically determined as outlined in Section 3, this is perfectly reasonable: in order to know how to stress a form, it is often necessary to have seen other words that share the same lemma. Nonetheless, it is also of interest to know how well the system works on words that do not share any lemmata with words in the training data. To that end, we collected a set of 248 forms that shared no lemmata with the training data. The two sets will be referred to in the next section as the "shared lemmata" and "no shared lemmata" sets.

7 Results

Table 4 gives word accuracy results for the different feature combinations, as follows: Dou et al's features (Dou et al., 2009); our affix features; our affix features plus affix features based on the abstract phonetic class versions of words; Dou et al's features plus our affix features; Dou et al's features plus our affix features plus the abstract affix features.

When we consider only primary stress (column 2 in Table 4, for the shared-lemmata test data, Dou et al's features performed the worst at 97.2% accuracy, with all feature combinations that include the affix features performing at the same level, 98.7%. For the no-shared-lemmata test data, using Dou et al's features alone achieved an accuracy of 80.6%. The affix features alone performed worse, at 79.8%, presumably because it is harder for them to generalize to unseen cases, but using the abstract affix features increased the performance to 81.0%, better than that of using Dou et al's features alone. As can be seen combining Dou et al's features with various combinations of the affix features improved the performance further.

Features	1 stress	1+2 stress
shared lemmata		
Dou et al	0.972	0.965
Aff	0.987	0.985
Aff+Abstr Aff	0.987	0.985
Dou et al+Aff	0.987	0.986
Dou et al+Aff+Abstr Aff	0.987	0.986
no shared lemmata		
Dou et al	0.806	0.798
Aff	0.798	0.782
Aff+Abstr	0.810	0.790
Dou et al+Aff	0.823	0.810
Dou et al+Aff+Abstr Aff	0.839	0.815

Table 4: Word accuracies for various feature combinations for both shared lemmata and no-shared lemmata conditions. The second column reports results where we consider only primary stress, the third column results where we also predict secondary stress.

For primary *and* secondary stress prediction (column 3 in the table), the results are overall degraded for most conditions but otherwise very similar in terms of ranking of the features to what we find with primary stress alone. Note though that for the shared-lemmata condition the results with affix features are almost as good as for the primary-stress-only case, whereas there is a significant drop in performance for the Dou et al. features. For the no-shared-lemmata condition, Dou et al.'s features fare rather better compared to the affix features. On the other hand there is a substantial benefit to combining the features, as the results for "Dou et al+Aff" and "Dou et al+Aff+Abstr Aff" show. Note that in the no-shared-lemmata condition, there is only one word that is marked with a secondary stress, and that stress is actually correctly predicted by all methods. Much of the difference between the Dou et al. features and the affix condition can be accounted for by three cases involving the same root, which the affix condition misas-

signs secondary stress to.

For the shared-lemmata task however there were a substantial number of differences, as one might expect given the nature of the features. Comparing just the Dou et al. features and the all-features condition, systematic benefit for the all-features condition was found for secondary stress assignment for productive prefixes where secondary stress is typically found. For example, the prefix аэро ('aero-') as in а`эродина'мика ('aerodynamics') typically has secondary stress. This is usually missed by the Dou et al. features, but is uniformly correct for the all-features condition.

Since the no-shared-lemmata data set is small, we tested significance using two permutation tests. The first computed a distribution of scores for the test data where successive single test examples were removed. The second randomly permuted the test data 248 times, after each random permutation, removing the first ten examples, and computing the score. Pairwise t-tests between all conditions for the primary-stress-only and for the primary plus secondary stress predictions, were highly significant in all cases.

We also experimented with a postaccent feature to model the postaccented class of nouns described in Section 3. For each *prefix* of the word, we record whether the following vowel is stressed or unstressed. This feature yielded only very slight improvements, and we do not report these results here.

8 Discussion

In this paper we have presented a Maximum Entropy ranking-based approach to Russian stress prediction. The approach is similar in spirit to the SVM-based ranking approach presented in (Dou et al., 2009), but incorporates additional affix-based features, which are motivated by linguistic analyses of the problem. We have shown that these additional features generalize better than the Dou et al. features in cases where we have seen a related form of the test word, and that combing the additional features with the Dou et al. features always yields an improvement.

References

Kenneth Church. 1985. Stress assignment in letter to sound rules for speech synthesis. In *Association for Computational Linguistics*, pages 246–253.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31:25–69, March.

Qing Dou, Shane Bergsma, Sittichai Jiampojamarn, and Grzegorz Kondrak. 2009. A ranking approach to stress prediction for letter-to-phoneme conversion. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 118–126, Suntec, Singapore, August. Association for Computational Linguistics.

Keith B. Hall, Scott Gilpin, and Gideon Mann. 2010. Mapreduce/bigtable for distributed optimization. In *Neural Information Processing Systems Workshop on Leaning on Cores, Clusters, and Clouds*.

Morris Halle. 1997. On stress and accent in Indo-European. *Language*, 73(2):275–313.

Steve Pearson, Roland Kuhn, Steven Fincke, and Nick Kibre. 2000. Automatic methods for lexical stress assignment and syllabification. In *International Conference on Spoken Language Processing*, pages 423–426.

Terence Wade. 1992. *A Comprehensive Russian Grammar*. Blackwell, Oxford.

Gabriel Webster. 2004. Improving letter-to-pronunciation accuracy with automatic morphologically-based stress prediction. In *International Conference on Spoken Language Processing*, pages 2573–2576.

Briony Williams. 1987. Word stress assignment in a text-to-speech synthesis system for British English. *Computer Speech and Language*, 2:235–272.

Andrey Zaliznyak. 1977. *Grammaticheskij slovar' russkogo jazyka*. Russkiy Yazik, Moscow.

Scaling to Large^3 Data: An efficient and effective method to compute Distributional Thesauri

Martin Riedl and **Chris Biemann**
FG Language Technology
Computer Science Department, Technische Universität Darmstadt
Hochschulstrasse 10, D-64289 Darmstadt, Germany
{riedl,biem}@cs.tu-darmstadt.de

Abstract

We introduce a new highly scalable approach for computing Distributional Thesauri (DTs). By employing pruning techniques and a distributed framework, we make the computation for very large corpora feasible on comparably small computational resources. We demonstrate this by releasing a DT for the whole vocabulary of Google Books syntactic n-grams. Evaluating against lexical resources using two measures, we show that our approach produces higher quality DTs than previous approaches, and is thus preferable in terms of speed and quality for large corpora.

1 Introduction

Using larger data to estimate models for machine learning applications as well as for applications of Natural Language Processing (NLP) has repeatedly shown to be advantageous, see e.g. (Banko and Brill, 2001; Brants et al., 2007). In this work, we tackle the influence of corpus size for building a distributional thesaurus (Lin, 1998). Especially, we shed light on the interaction of similarity measures and corpus size, as well as aspects of scalability.

We shortly introduce the JoBimText framework for distributional semantics and show its scalability for large corpora. For the computation of the data we follow the MapReduce (Dean and Ghemawat, 2004) paradigm. The computation of similarities between terms becomes challenging on large corpora, as both the numbers of terms to be compared and the number of context features increases. This makes standard similarity calculations as proposed in (Lin, 1998; Curran, 2002; Lund and Burgess, 1996; Weeds et al., 2004) computationally infeasible. These approaches first calculate an information measure between each word and the according context and then calculate the similarity between all words, based on the information measure for all shared contexts.

2 Related Work

A variety of approaches to compute DTs have been proposed to tackle issues regarding size and runtime. The reduction of the feature space seems to be one possibility, but still requires the computation of such reduction cf. (Blei et al., 2003; Golub and Kahan, 1965). Other approaches use randomised indexing for storing counts or hashing functions to approximate counts and measures (Gorman and Curran, 2006; Goyal et al., 2010; Sahlgren, 2006). Another possibility is the usage of distributed processing like MapReduce. In (Pantel et al., 2009; Agirre et al., 2009) a DT is computed using MapReduce on 200 quad core nodes (for 5.2 billion sentences) respectively 2000 cores (1.6 Terawords), an amount of hardware only available to commercial search engines. Whereas Agirre uses a χ^2 test to measure the information between terms and context, Pantel uses the Pointwise Mutual Information (PMI). Then, both approaches use the cosine similarity to calculate the similarity between terms. Furthermore, Pantel describes an optimization for the calculation of the cosine similarity. Whereas Pantel and Lin (2002) describe a method for sense clustering, they also use a method to calculate similarities between terms. Here, they propose a pruning scheme similar to ours, but do not explicitly evaluate its effect.

The evaluation of DTs has been performed in extrinsic and intrinsic manner. Extrinsic evaluations have been performed using e.g. DTs for automatic

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 884–890,
Seattle, Washington, USA, 18-21 October 2013.

set expansion (Pantel et al., 2009) or phrase polarity identification (Goyal and Daumé, 2011). In this work we will concentrate on intrinsic evaluations: Lin (1997; 1998) introduced two measures using WordNet (Miller, 1995) and Roget's Thesaurus. Using WordNet, he defines context features (synsets a word occurs in Wordnet or subsets when using Roget's Thesaurus) and then builds a gold standard thesaurus using a similarity measure. Then he evaluates his generated Distributional Thesaurus (DT) with respect to the gold standard thesauri. Weeds et al. (2004) evaluate various similarity measures based on 1000 frequent and 1000 infrequent words. Curran (2004) created a gold standard thesaurus by manually extracting entries from several English thesauri for 70 words. His automatically generated DTs are evaluated against this gold standard thesaurus using several measures. We will report on his measure and additionally propose a measure based on WordNet paths.

3 Building a Distributional Thesaurus

Here we present our scalable DT algorithm using the MapReduce paradigm, which is divided into two parts: The holing system and a computational method to calculate distributional similarities. A more detailed description, especially for the MapReduce steps, can be found in (Biemann and Riedl, 2013).

3.1 Holing System

The holing operation splits an observation (e.g. a dependency relation) into a pair of two parts: a term and a context feature. This captures their first-order relationship. These pairs are subsequently used for the computation of the similarities between terms, leading to a second-order relation. The representation can be formalized by the pair <x,y> where x is the term and y represents the context feature. The position of x in y is denoted by the hole symbol '@'. As an example the dependency relation *(nsub;gave$_2$;I$_1$)* could be transferred to *<gave$_2$,(nsub;@;I$_1$)>* and *<I$_1$,(nsub;gave$_2$;@)>*. This representation scheme is more generic then the schemes introduced in (Lin, 1998; Curran, 2002), as it allows to characterise pairs by several holes, which could be used to learn analogies, cf. (Turney and Littman, 2005).

3.2 Distributional Similarity

First, we count the frequency for each first-order relation and remove all features that occur with more than w terms, as these context features tend to be too general to characterise the similarity between other words (Rychlý and Kilgarriff, 2007; Goyal et al., 2010, cmp.). From this. we calculate a significance score for all first-order relations. For this work, we implemented two different significance measures: Pointwise Mutual Information (PMI): $PMI(term, feature) = \log_2(\frac{f(term,feature)}{f(term)f(feature)})$ (Church and Hanks, 1990) and Lexicographer's Mutual Information (LMI): $LMI(term, feature) = f(term, feature)\log_2(\frac{f(term,feature)}{f(term)f(feature)})$ (Evert, 2005).

We then prune all negatively correlated pairs ($s<0$). The maximum number of context features per term are defined with p, as we argue that it is sufficient to keep only the p most salient (ordered descending by their significance score) context features per term. Features of low saliency generally should not contribute much to the similarity of terms and also could lead to spurious similarity scores. Afterwards, all terms are aggregated by their features, which allows us to compute similarity scores between all terms that share at least one such feature.

Whereas the method introduced by (Pantel and Lin, 2002) is very similar to the one proposed in this paper (the similarity between terms is calculated solely by the number of features two terms share), they use PMI to rank features and do not use pruning to scale to large corpora, as they use a rather small corpus. Additionally, they do not evaluate the effect of such pruning.

In contrast to the best measures proposed by Lin (1998; Curran (2002; Pantel et al. (2009; Goyal et al. (2010) we do not calculate any information measure using frequencies of features and terms (we use significance ranking instead), as shown in Table 1.

Additionally, we avoid any similarity measurement using the information measure, as also done in these approaches, to calculate the similarity over the feature counts of each term: we merely count how many salient features two terms share. All these constraints makes this approach more scalable to larger corpora, as we do not need to know the full list of

Information Measures	
Lin's formula	$I(term, feature) = lin(term, feature) = \log \frac{f(term,feature)*f(relation(feature))}{\sum(f(word,relation(feature))f(word)}$
Curran's TTest	$I(term, feature) = ttest(term, feature) = \frac{p(term,feature)-p(feature)*p(term)}{\sqrt{p(feature)*p(term)}}$
Similarity Measures	
Lin's formula	$sim(t_1, t_2) = \frac{\sum_{f \in features(t_1) \cap features(t_2)}(I(t_1,f)+I(t_2,f))}{\sum_{f \in features(t_1)} I(t_1,f)+\sum_{f \in features(w_2)} I(w_2,f)}$
Curran's Dice	$sim(t_1, t_2) = \frac{\sum_{f \in features(t_1) \cap features(t_2)} min(I(t_1,f),I(t_2,f))}{\sum_{f \in features(t_1) \cap features(t_2)}(I(t_1,f)+I(t_2,f))}$ with $I(t, f) > 0$
Our Measure	$sim(t_1, t_2) = \sum_{f \in features(t_1) \cap features(t_2)} 1$ with $s > 0$

Table 1: Similarity measures used for computing the distributional similarity between terms.

features for a term pair at any time. While our computations might seem simplistic, we demonstrate its adequacy for large corpora in Section 5.

4 Evaluation

The evaluation is performed using a recent dump of English Wikipedia, containing 36 million sentences and a newspaper corpus, compiled from 120 million sentences (about 2 Gigawords) from Leipzig Corpora Collection (Richter et al., 2006) and the Gigaword corpus (Parker et al., 2011). The DTs are based on collapsed dependencies from the Stanford Parser (Marneffe et al., 2006) in the holing operation. For all DTs we use the pruning parameters s=0, p=1000 and w=1000. In a final evaluation, we use the syntactic n-grams built from Google Books (Goldberg and Orwant, 2013).

To show the impact of corpus size, we downsampled our corpora to 10 million, 1 million and 100,000 sentences. We compare our results against DTs calculated using Lin's (Lin, 1998) measure and the best measure proposed by Curran (2002) (see Table 1).

Our evaluation is performed using the same 1000 frequent and 1000 infrequent nouns as previously employed by Weeds et al. (2004). We create a gold standard, by extracting reasonable entries of these 2000 nouns using Roget's 1911 thesaurus, Moby Thesaurus, Merriam Webster's Thesaurus, the Big Huge Thesaurus and the OpenOffice Thesaurus and employ the inverse ranking measure (Curran, 2002) to evaluate the DTs.

Furthermore, we introduce a WordNet-based method. To calculate the similarity between two terms, we use the WordNet::Similarity path (Pedersen et al., 2004) measure. While its absolute scores are hard to interpret due to inhomogenity in the granularity of WordNet, they are well-suited for relative comparison. The score between two terms is inversely proportional to the shortest path between all the synsets of both terms. The highest possible score is one, if two terms share a synset. We compare the average score of the top five (or ten) entries in the DT for each of the 2000 selected words for our comparison.

5 Results

First, we inspect the results of Curran's measure using the Wikipedia and newspaper corpus for the frequent nouns, shown in Figure 1.

Both graphs show the inverse ranking score against the size of the corpus. Our method scores consistently higher when using LMI instead of PMI for ranking the features per term. The PMI measure declines when the corpus becomes larger. This can be attributed to the fact that PMI favors term-context pairs involving rare contexts (Bordag, 2008). Computing similarities between terms should not be performed on the basis of rare contexts, as these do not generalize well because of their sparseness.

All other measures improve with larger corpora. It is surprising that recent works use PMI to calculate similarities between terms (Goyal et al., 2010; Pantel et al., 2009), who, however evaluate their approach only with respect to their own implementation or extrinsically, and do not prune on saliency. Apart from the PMI measure, Curran's measure leads to the weakest results. We could not confirm that his measure outperforms Lin's measure as stated in (Curran, 2002)[1]. An explanation for this results

[1]Regarding Curran's Dice formula, it is not clear whether to use the intersection or the union of the features. We use an intersection, as it is unclear how to interpret the minimum function otherwise, and the alternatives performed worse.

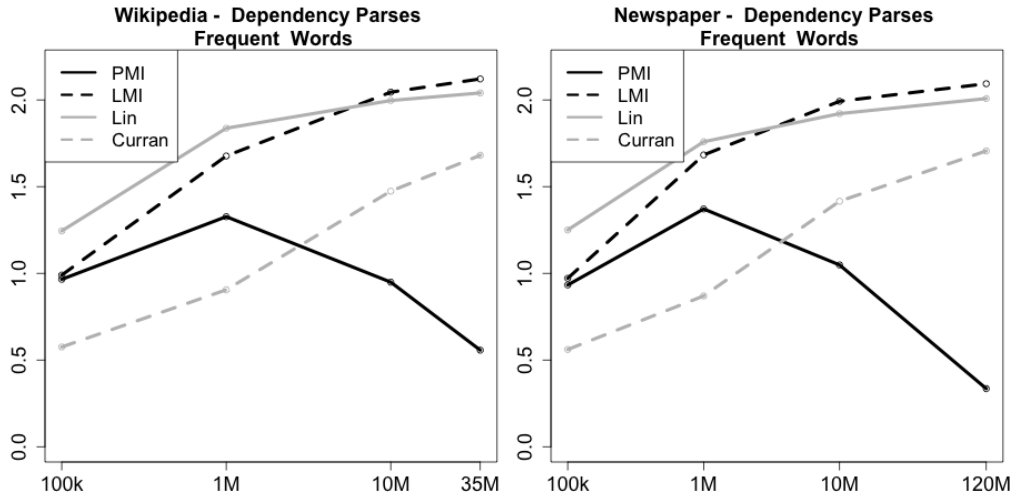


Figure 1: Inverse ranking for 1000 frequent nouns (Wikipedia left, Newspaper right) for different sized corpora. The 4 lines represent the scores of following DTs: our method using LMI (dashed black line) and the PMI significance measure (solid black line) and Curran's (dash bray line) and Lin's measure (solid tray line).

might be the use of a different parser, very few test words and also a different gold standard thesaurus in his evaluation. Comparing our method using LMI to Lin's method, we achieve lower scores with our method using small corpora, but surpass Lin's measure from 10 million sentences onwards.

Next, we show the results of the WordNet evaluation measure in Figure 2. Comparing the top 10 (upper) to the top 5 words (lower) used for the evaluation, we can observe higher scores for the top 5 words, which validates the ranking. These results are highly correlated to the results achieved with the inverse ranking measure. This is a positive result, as the WordNet measure can be performed automatically using a single public resource[2]. In Figure 3, we show results for the 1000 infrequent nouns using the inverse ranking (upper) and the WordNet measure (lower).

We can see that our method using PMI does not decline for larger corpora, as the limit on first-order features is not reached and frequent features are still being used. Comparing our LMI DT is en par with Lin's measure for 10 million sentences, and makes better use of large data when using the complete dataset. Again, the inverse ranking and the WordNet Path measure are highly correlated.

[2]Building a gold standard thesaurus following Curran (2002) needs access to all the used thesauri. Whereas for some, programming interfaces exist, often with limited access and licence restrictions, others have to be extracted manually.

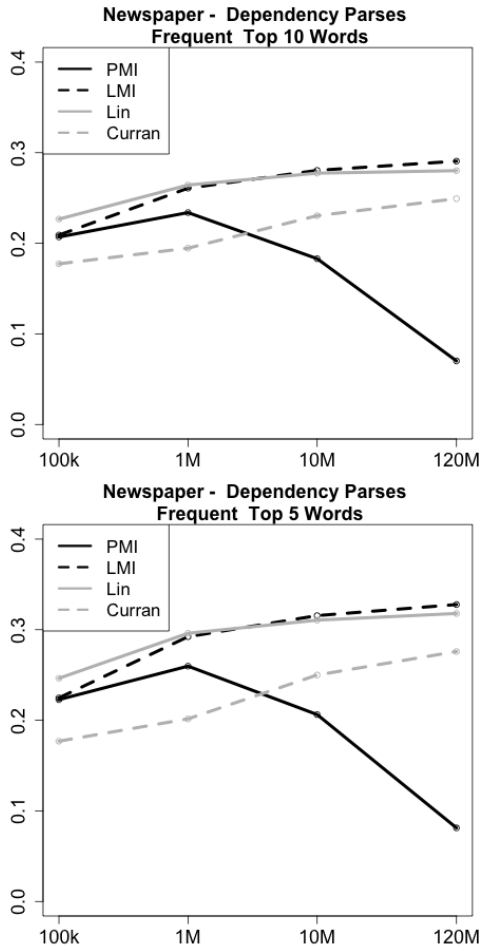


Figure 2: Results, using the WordNet:Path measure for frequent nouns using the newspaper corpus.

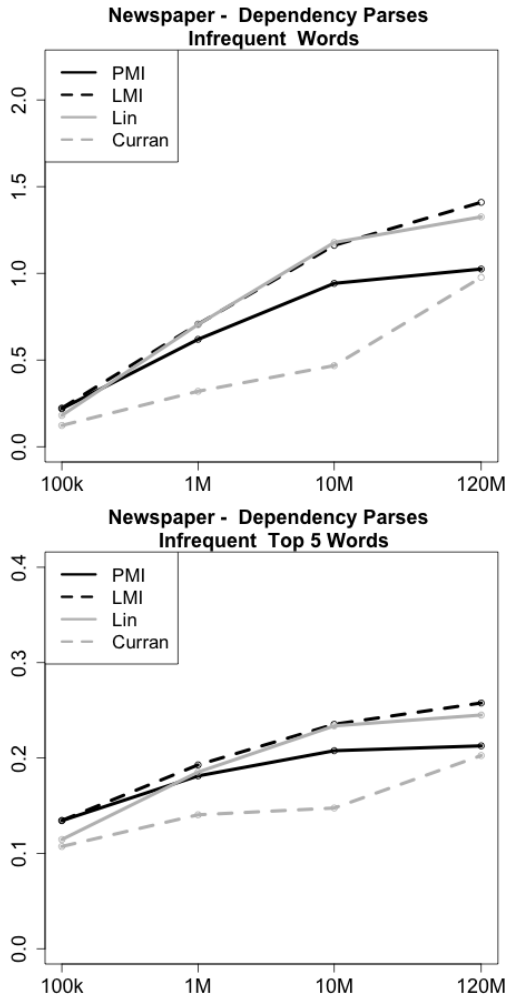


Figure 3: WordNet::Path results for 1000 infrequent nouns

The results shown here validate our pruning approach. Whereas Lin and Curran propose approaches to filter features that have low word feature scores, they do not remove features that occur with too many words, which is done in this work. Using these pruning steps, a simplistic similarity measure does not only lead to reduced computation times, but also to better results, when using larger corpora.

5.1 Using a large[3] corpus

We demonstrate the scalability of our method using the very large Google Books dataset (Goldberg and Orwant, 2013), consisting of dependencies extracted from 17.6 billion sentences. The evaluation results, using different measures, are given in Table 2.

Comparing the results for the Google Books DT to the ones achieved using Wikipedia and the newspaper, we can observe a boost in the performance, both for the inverse ranking and the WordNet measures. Additionally, we show results for the P@1 measure, which indicates the percentage of entries, whose first entry is in the gold standard thesaurus. Remarkably, we get a P@1 against our gold standard thesaurus of 76% for frequent and 64% for infrequent nouns using the Google Books DT.

	Corpus	Inv.	P@1	Path@5	Path@10
frequent nouns	Newspaper	2.0935	0.709	0.3277	0.2906
	Wikipedia	2.1213	0.703	0.3365	0.2968
	Google Books	2.3171	0.764	0.3712	0.3217
infrequent nouns	Newspaper	1.4097	0.516	0.2577	0.2269
	Wikipedia	1.3832	0.514	0.2565	0.2265
	Google Books	1.8125	0.641	0.2989	0.2565

Table 2: Comparing results for different corpora.

The most computation time was needed for the dependency parsing and took two weeks on a small cluster (64 cores on 8 nodes) for the 120 million Newspaper sentences. The DT for the Google Books was calculated in under 30 hours on a Hadoop cluster (192 cores on 16 nodes) and could be calculated within 10 hours for the Newspaper corpus. The computation of a DT using this huge corpus would be intractable with standard vector-based measurements. Even computing Lin's and Curran's vector-based similarity measure for the whole vocabulary of the newspaper corpus was not possible with our Hadoop cluster, as too much memory would have been required and thus we computed similarities only for the 2000 test nouns on a server with 92GB of main memory.

6 Conclusion

We have introduced a highly scalable approach to DT computation and showed its adequacy for very large corpora. Evaluating against thesauri and WordNet, we demonstrated that our similarity measure yields better-quality DTs and scales to corpora of billions of sentences, even on comparably small compute clusters. We achieve this by a number of pruning operations, and distributed processing. The framework and the DTs for Google Books, Newspaper and Wikipedia are available online[3] under the ASL 2.0 licence.

[3] https://sf.net/projects/jobimtext/

Acknowledgments

This work has been supported by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-konomischer Exzellenz" (LOEWE) as part of the research center "Digital Humanities". We would also thank the anonymous reviewers for their comments, which greatly helped to improve the paper.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27, Boulder, Colorado, USA.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 26–33, Toulouse, France.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Stefan Bordag. 2008. A comparison of co-occurrence and similarity measures as simulations of context. In *CICLing'08 Proceedings of the 9th international conference on Computational linguistics and intelligent text processing*, pages 52–63, Haifa, Israel.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

James R. Curran. 2002. Ensemble methods for automatic thesaurus extraction. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 222–229, Philadelphia, PA, USA.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of Operating Systems, Desing & Implementation (OSDI) '04*, pages 137–150, San Francisco, CA, USA.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences: Word Pairs and Collocations.* Ph.D. thesis, Institut für maschinelle Sprachverarbeitung, University of Stuttgart.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 241–247, Atlanta, Georgia, USA.

Gene H. Golub and William M. Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *J. Soc. Indust. Appl. Math.: Ser. B, Numer. Anal.*, 2:205–224.

James Gorman and James R. Curran. 2006. Scaling distributional similarity to large corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 361–368, Sydney, Australia.

Amit Goyal and Hal Daumé, III. 2011. Generating semantic orientation lexicon using large data and thesaurus. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pages 37–43, Portland, Oregon, USA.

Amit Goyal, Jagadeesh Jagarlamudi, Hal Daumé, III, and Suresh Venkatasubramanian. 2010. Sketch techniques for scaling distributional similarity to the web. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, GEMS '10, pages 51–56, Uppsala, Sweden.

Dekang Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98, pages 64–71, Madrid, Spain.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics - Volume 2*, COLING '98, pages 768–774, Montreal, Quebec, Canada.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods*, 28(2):203–208.

Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2006*, Genova, Italy.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 613–619, Edmonton, Alberta, Canada.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 938–947, Singapore.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. *English Gigaword Fifth Edition*. Linguistic Data Consortium, Philadelphia, PA, USA.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, Boston, Massachusetts, USA.

Matthias Richter, Uwe Quasthoff, Erla Hallsteinsdóttir, and Chris Biemann. 2006. Exploiting the leipzig corpora collection. In *Proceedings of the IS-LTC 2006*, Ljubljana, Slovenia.

Pavel Rychlý and Adam Kilgarriff. 2007. An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 41–44, Prague, Czech Republic.

Magnus Sahlgren. 2006. *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* Ph.D. thesis, Stockholm University.

Peter D. Turney and Michael L. Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251–278.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, pages 1015–1021, Geneva, Switzerland.

Discriminative Improvements to Distributional Sentence Similarity

Yangfeng Ji
School of Interactive Computing
Georgia Institute of Technology
jiyfeng@gatech.edu

Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
jacobe@gatech.edu

Abstract

Matrix and tensor factorization have been applied to a number of semantic relatedness tasks, including paraphrase identification. The key idea is that similarity in the latent space implies semantic relatedness. We describe three ways in which labeled data can improve the accuracy of these approaches on paraphrase classification. First, we design a new discriminative term-weighting metric called TF-KLD, which outperforms TF-IDF. Next, we show that using the latent representation from matrix factorization as features in a classification algorithm substantially improves accuracy. Finally, we combine latent features with fine-grained n-gram overlap features, yielding performance that is 3% more accurate than the prior state-of-the-art.

1 Introduction

Measuring the semantic similarity of short units of text is fundamental to many natural language processing tasks, from evaluating machine translation (Kauchak and Barzilay, 2006) to grouping redundant event mentions in social media (Petrović et al., 2010). The task is challenging because of the infinitely diverse set of possible linguistic realizations for any idea (Bhagat and Hovy, 2013), and because of the short length of individual sentences, which means that standard bag-of-words representations will be hopelessly sparse.

Distributional methods address this problem by transforming the high-dimensional bag-of-words representation into a lower-dimensional latent space. This can be accomplished by factoring a matrix or tensor of term-context counts (Turney and Pantel, 2010); proximity in the induced latent space has been shown to correlate with semantic similarity (Mihalcea et al., 2006). However, factoring the term-context matrix means throwing away a considerable amount of information, as the original matrix of size $M \times N$ (number of instances by number of features) is factored into two smaller matrices of size $M \times K$ and $N \times K$, with $K \ll M, N$. If the factorization does not take into account labeled data about semantic similarity, important information can be lost.

In this paper, we show how labeled data can considerably improve distributional methods for measuring semantic similarity. First, we develop a new discriminative term-weighting metric called TF-KLD, which is applied to the term-context matrix *before* factorization. On a standard paraphrase identification task (Dolan et al., 2004), this method improves on both traditional TF-IDF and Weighted Textual Matrix Factorization (WTMF; Guo and Diab, 2012). Next, we convert the latent representations of each sentence pair into a feature vector, which is used as input to a linear SVM classifier. This yields further improvements and substantially outperforms the current state-of-the-art on paraphrase classification. We then add "fine-grained" features about the lexical similarity of the sentence pair. The combination of latent and fine-grained features yields further improvements in accuracy, demonstrating that these feature sets provide complementary information on semantic similarity.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 891–896,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Without attempting to do justice to the entire literature on paraphrase identification, we note three high-level approaches: (1) string similarity metrics such as n-gram overlap and BLEU score (Wan et al., 2006; Madnani et al., 2012), as well as string kernels (Bu et al., 2012); (2) syntactic operations on the parse structure (Wu, 2005; Das and Smith, 2009); and (3) distributional methods, such as latent semantic analysis (LSA; Landauer et al., 1998), which are most relevant to our work. One application of distributional techniques is to replace individual words with distributionally similar alternatives (Kauchak and Barzilay, 2006). Alternatively, Blacoe and Lapata (2012) show that latent word representations can be combined with simple element-wise operations to identify the semantic similarity of larger units of text. Socher et al. (2011) propose a syntactically-informed approach to combine word representations, using a recursive auto-encoder to propagate meaning through the parse tree.

We take a different approach: rather than representing the meanings of individual words, we directly obtain a distributional representation for the entire sentence. This is inspired by Mihalcea et al. (2006) and Guo and Diab (2012), who treat sentences as pseudo-documents in an LSA framework, and identify paraphrases using similarity in the latent space. We show that the performance of such techniques can be improved dramatically by using supervised information to (1) reweight the individual distributional features and (2) learn the importance of each latent dimension.

3 Discriminative feature weighting

Distributional representations (Turney and Pantel, 2010) can be induced from a co-occurrence matrix $\mathbf{W} \in \mathbb{R}^{M \times N}$, where M is the number of instances and N is the number of distributional features. For paraphrase identification, each instance is a sentence; features may be unigrams, or may include higher-order n-grams or dependency pairs. By decomposing the matrix $\mathbf{W}$, we hope to obtain a latent representation in which semantically-related sentences are similar. Singular value decomposition (SVD) is traditionally used to perform this factorization. However, recent work has demonstrated the robustness of nonnegative matrix factorization (NMF; Lee and Seung, 2001) for text mining tasks (Xu et al., 2003; Arora et al., 2012); the difference from SVD is the addition of a non-negativity constraint in the latent representation based on non-orthogonal basis.

While $\mathbf{W}$ may simply contain counts of distributional features, prior work has demonstrated the utility of reweighting these counts (Turney and Pantel, 2010). TF-IDF is a standard approach, as the inverse document frequency (IDF) term increases the importance of rare words, which may be more discriminative. Guo and Diab (2012) show that applying a special weight to unseen words can further improvement performance on paraphrase identification.

We present a new weighting scheme, TF-KLD, based on supervised information. The key idea is to increase the weights of distributional features that are discriminative, and to decrease the weights of features that are not. Conceptually, this is similar to Linear Discriminant Analysis, a supervised feature weighting scheme for continuous data (Murphy, 2012).

More formally, we assume labeled sentence pairs of the form $\langle \vec{w}_i^{(1)}, \vec{w}_i^{(2)}, r_i \rangle$, where $\vec{w}_i^{(1)}$ is the binarized vector of distributional features for the first sentence, $\vec{w}_i^{(2)}$ is the binarized vector of distributional features for the second sentence, and $r_i \in \{0, 1\}$ indicates whether they are labeled as a paraphrase pair. Assuming the order of the sentences within the pair is irrelevant, then for k-th distributional feature, we define two Bernoulli distributions:

- $p_k = P(w_{ik}^{(1)} | w_{ik}^{(2)} = 1, r_i = 1)$. This is the probability that sentence $w_i^{(1)}$ contains feature k, given that k appears in $w_i^{(2)}$ and the two sentences are labeled as paraphrases, $r_i = 1$.

- $q_k = P(w_{ik}^{(1)} | w_{ik}^{(2)} = 1, r_i = 0)$. This is the probability that sentence $w_i^{(1)}$ contains feature k, given that k appears in $w_i^{(2)}$ and the two sentences are labeled as not paraphrases, $r_i = 0$.

The Kullback-Leibler divergence $KL(p_k || q_k) = \sum_x p_k(x) \log \frac{p_k(x)}{q_k(x)}$ is then a measure of the discriminability of feature k, and is guaranteed to be non-

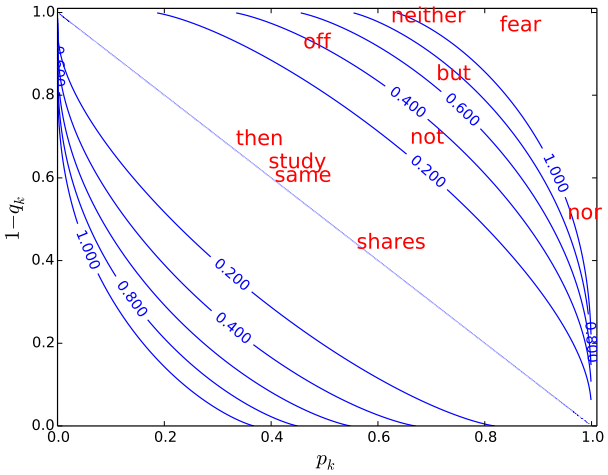


Figure 1: Conditional probabilities for a few hand-selected unigram features, with lines showing contours with identical KL-divergence. The probabilities are estimated based on the MSRPC training set (Dolan et al., 2004).

negative.[1] We use this divergence to reweight the features in $\mathbf{W}$ before performing the matrix factorization. This has the effect of increasing the weights of features whose likelihood of appearing in a pair of sentences is strongly influenced by the paraphrase relationship between the two sentences. On the other hand, if $p_k = q_k$, then the KL-divergence will be zero, and the feature will be ignored in the matrix factorization. We name this weighting scheme TF-KLD, since it includes the term frequency and the KL-divergence.

Taking the unigram feature *not* as an example, we have $p_k = [0.66, 0.34]$ and $q_k = [0.31, 0.69]$, for a KL-divergence of 0.25: the likelihood of this word being shared between two sentence is strongly dependent on whether the sentences are paraphrases. In contrast, the feature *then* has $p_k = [0.33, 0.67]$ and $q_k = [0.32, 0.68]$, for a KL-divergence of 3.9×10^{-4}. Figure 1 shows the distributions of these and other unigram features with respect to p_k and $1 - q_k$. The diagonal line running through the middle of the plot indicates zero KL-divergence, so features on this line will be ignored.

[1] We obtain very similar results with the opposite divergence $KL(q_k||p_k)$. However, the symmetric Jensen-Shannon divergence performs poorly.

1	unigram recall
2	unigram precision
3	bigram recall
4	bigram precision
5	dependency relation recall
6	dependency relation precision
7	BLEU recall
8	BLEU precision
9	Difference of sentence length
10	Tree-editing distance

Table 1: Fine-grained features for paraphrase classification, selected from prior work (Wan et al., 2006).

4 Supervised classification

While previous work has performed paraphrase classification using distance or similarity in the latent space (Guo and Diab, 2012; Socher et al., 2011), more direct supervision can be applied. Specifically, we convert the latent representations of a pair of sentences $\vec{v}_1$ and $\vec{v}_2$ into a sample vector,

$$\vec{s}(\vec{v}_1, \vec{v}_2) = \left[\vec{v}_1 + \vec{v}_2, |\vec{v}_1 - \vec{v}_2|\right], \qquad (1)$$

concatenating the element-wise sum $\vec{v}_1 + \vec{v}_2$ and absolute difference $|\vec{v}_1 - \vec{v}_2|$. Note that $\vec{s}(\cdot, \cdot)$ is symmetric, since $\vec{s}(\vec{v}_1, \vec{v}_2) = \vec{s}(\vec{v}_2, \vec{v}_1)$. Given this representation, we can use any supervised classification algorithm.

A further advantage of treating paraphrase as a supervised classification problem is that we can apply additional features besides the latent representation. We consider a subset of features identified by Wan et al. (2006), listed in Table 1. These features mainly capture fine-grained similarity between sentences, for example by counting specific unigram and bigram overlap.

5 Experiments

Our experiments test the utility of the TF-KLD weighting towards paraphrase classification, using the Microsoft Research Paraphrase Corpus (Dolan et al., 2004). The training set contains 2753 true paraphrase pairs and 1323 false paraphrase pairs; the test set contains 1147 and 578 pairs, respectively.

The TF-KLD weights are constructed from only the training set, while matrix factorizations are per-

formed on the entire corpus. Matrix factorization on both training and (unlabeled) test data can be viewed as a form of *transductive learning* (Gammerman et al., 1998), where we assume access to unlabeled test set instances.[2] We also consider an inductive setting, where we construct the basis of the latent space from only the training set, and then project the test set onto this basis to find the corresponding latent representation. The performance differences between the transductive and inductive settings were generally between 0.5% and 1%, as noted in detail below. We reiterate that the TF-KLD weights are never computed from test set data.

Prior work on this dataset is described in section 2. To our knowledge, the current state-of-the-art is a supervised system that combines several machine translation metrics (Madnani et al., 2012), but we also compare with state-of-the-art unsupervised matrix factorization work (Guo and Diab, 2012).

5.1 Similarity-based classification

In the first experiment, we predict whether a pair of sentences is a paraphrase by measuring their cosine similarity in latent space, using a threshold for the classification boundary. As in prior work (Guo and Diab, 2012), the threshold is tuned on held-out training data. We consider two distributional feature sets: **FEAT**$_1$, which includes unigrams; and **FEAT**$_2$, which also includes bigrams and unlabeled dependency pairs obtained from MaltParser (Nivre et al., 2007). To compare with Guo and Diab (2012), we set the latent dimensionality to $K = 100$, which was the same in their paper. Both SVD and NMF factorization are evaluated; in both cases, we minimize the Frobenius norm of the reconstruction error.

Table 2 compares the accuracy of a number of different configurations. The transductive TF-KLD weighting yields the best overall accuracy, achieving 72.75% when combined with non-negative matrix factorization. While NMF performs slightly better than SVD in both comparisons, the major difference is the performance of discriminative TF-KLD weighting, which outperforms TF-IDF regardless of the factorization technique. When we perform the matrix factorization on only the training data, the accuracy on the test set is 73.58%, with F1 score 80.55%.

[2]Another example of transductive learning in NLP is when Turian et al. (2010) induced word representations from a corpus that included both training and test data for their downstream named entity recognition task.

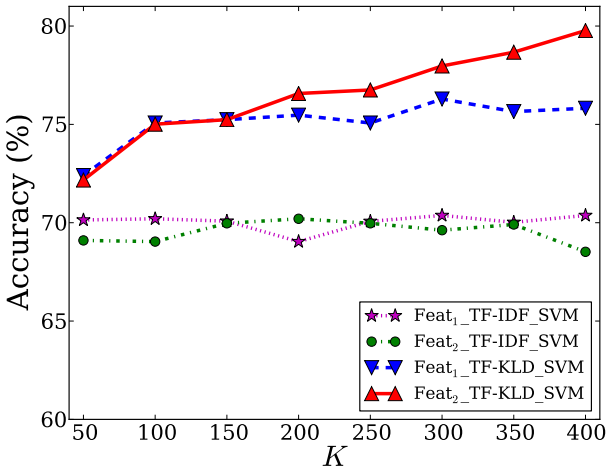


Figure 2: Accuracy of feature and weighting combinations in the classification framework.

5.2 Supervised classification

Next, we apply supervised classification, constructing sample vectors from the latent representation as shown in Equation 1. For classification, we choose a Support Vector Machine with a linear kernel (Fan et al., 2008), leaving a thorough comparison of classifiers for future work. The classifier parameter C is tuned on a development set comprising 20% of the original training set.

Figure 2 presents results for a range of latent dimensionalities. Supervised learning identifies the important dimensions in the latent space, yielding significantly better performance that the similarity-based classification from the previous experiment. In Table 3, we compare against prior published work, using the held-out development set to select the best value of K (again, $K = 400$). The best result is from TF-KLD, with distributional features FEAT$_2$, achieving 79.76% accuracy and 85.87% F1. This is well beyond all known prior results on this task. When we induce the latent basis from only the training data, we get 78.55% on accuracy and 84.59% F1, also better than the previous state-of-art.

Finally, we augment the distributional representation, concatenating the ten "fine-grained" features in Table 1 to the sample vectors described in Equation 1. As shown in Table 3, the accu-

Factorization	Feature set	Weighting	K	Measure	Accuracy (%)	F1
SVD	unigrams	TF-IDF	100	cosine sim.	68.92	80.33
NMF	unigrams	TF-IDF	100	cosine sim.	68.96	80.14
WTMF	unigrams	TF-IDF	100	cosine sim.	71.51	not reported
SVD	unigrams	TF-KLD	100	cosine sim.	72.23	81.19
NMF	unigrams	TF-KLD	100	cosine sim.	**72.75**	**81.48**

Table 2: Similarity-based paraphrase identification accuracy. Results for WTMF are reprinted from the paper by Guo and Diab (2012).

	Acc.	F1
Most common class	66.5	79.9
(Wan et al., 2006)	75.6	83.0
(Das and Smith, 2009)	73.9	82.3
(Das and Smith, 2009) with 18 features	76.1	82.7
(Bu et al., 2012)	76.3	not reported
(Socher et al., 2011)	76.8	83.6
(Madnani et al., 2012)	77.4	84.1
FEAT_2, TF-KLD, SVM	79.76	85.87
FEAT_2, TF-KLD, SVM, Fine-grained features	**80.41**	**85.96**

Table 3: Supervised classification. Results from prior work are reprinted.

racy now improves to 80.41%, with an F1 score of 85.96%. When the latent representation is induced from only the training data, the corresponding results are 79.94% on accuracy and 85.36% F1, again better than the previous state-of-the-art. These results show that the information captured by the distributional representation can still be augmented by more fine-grained traditional features.

6 Conclusion

We have presented three ways in which labeled data can improve distributional measures of semantic similarity at the sentence level. The main innovation is TF-KLD, which discriminatively reweights the distributional features *before* factorization, so that discriminability impacts the induction of the latent representation. We then transform the latent representation into a sample vector for supervised learning, obtaining results that strongly outperform the prior state-of-the-art; adding fine-grained lexical features further increases performance. These ideas may have applicability in other semantic similarity tasks, and we are also eager to apply them to new, large-scale automatically-induced paraphrase corpora (Ganitkevitch et al., 2013).

Acknowledgments

We thank the reviewers for their helpful feedback, and Weiwei Guo for quickly answering questions about his implementation. This research was supported by a Google Faculty Research Award to the second author.

References

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning Topic Models - Going beyond SVD. In *FOCS*, pages 1–10.

Rahul Bhagat and Eduard Hovy. 2013. What Is a Paraphrase? *Computational Linguistics*.

William Blacoe and Mirella Lapata. 2012. A Comparison of Vector-based Representations for Semantic Composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fan Bu, Hang Li, and Xiaoyan Zhu. 2012. String Rewriting kernel. In *Proceedings of ACL*, pages 449–458. Association for Computational Linguistics.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference*

of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, pages 468–476, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *COLING*.

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

Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. 1998. Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 148–155. Morgan Kaufmann Publishers Inc.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL*, pages 758–764. Association for Computational Linguistics.

Weiwei Guo and Mona Diab. 2012. Modeling Sentences in the Latent Space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 864–872, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of NAACL*, pages 455–462. Association for Computational Linguistics.

Thomas Landauer, Peter W. Foltz, and Darrel Laham. 1998. Introduction to Latent Semantic Analysis. *Discource Processes*, 25:259–284.

Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)*.

Nitin Madnani, Joel R. Tetreault, and Martin Chodorow. 2012. Re-examining Machine Translation Metrics for Paraphrase Identification. In *HLT-NAACL*, pages 182–190. The Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*.

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proceedings of HLT-NAACL*, pages 181–189. Association for Computational Linguistics.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling And Unfolding Recursive Autoencoders For Paraphrase Detection. In *Advances in Neural Information Processing Systems (NIPS)*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word Representation: A Simple and General Method for Semi-Supervised Learning. In *ACL*, pages 384–394.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *JAIR*, 37:141–188.

Ssephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2006. Using Dependency-based Features to Take the "Para-farce" out of Paraphrase. In *Proceedings of the Australasian Language Technology Workshop*.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 25–30. Association for Computational Linguistics.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document Clustering based on Non-Negative Matrix Factorization. In *SIGIR*, pages 267–273.

Is Twitter A Better Corpus for Measuring Sentiment Similarity?

Shi Feng[1], **Le Zhang**[1], **Binyang Li**[2,3], **Daling Wang**[1], **Ge Yu**[1], **Kam-Fai Wong**[3]
[1]Northeastern University, Shenyang, China
[2]University of International Relations, Beijing, China
[3]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{fengshi,wangdaling,yuge}@ise.neu.edu.cn, zhang777le@gmail.com
{byli,kfwong}@se.cuhk.edu.hk

Abstract

Extensive experiments have validated the effectiveness of the corpus-based method for classifying the word's sentiment polarity. However, no work is done for comparing different corpora in the polarity classification task. Nowadays, Twitter has aggregated huge amount of data that are full of people's sentiments. In this paper, we empirically evaluate the performance of different corpora in sentiment similarity measurement, which is the fundamental task for word polarity classification. Experiment results show that the Twitter data can achieve a much better performance than the Google, Web1T and Wikipedia based methods.

1 Introduction

Measuring semantic similarity for words and short texts has long been a fundamental problem for many applications such as word sense disambiguation, query expansion, search advertising and so on.

Determining the word's polarity plays a critical role in opinion mining and sentiment analysis task. Usually we can detect the word's polarity by measuring it's semantic similarity with a positive seed word se_p and a negative seed word se_n respectively, as shown in Formula (1):

$$SO(w) = sim(w, se_p) - sim(w, se_n) \quad (1)$$

where $sim(w_i, w_j)$ is the semantic similarity measurement method for the given word w_i and w_j. A lot of papers have been published for designing appropriate similarity measurements. One direction is to learn similarity from the knowledge base or concept taxonomy (Lin, 1998; Resnik, 1999). Another direction is to learn semantic similarity with the help of large corpus such as Web or Wikipedia data (Sahami and Heilman, 2006; Yih and Meek, 2007; Bollegala et al., 2011; Gabrilovich and Markovitch, 2007). The basic assumption of this kind of methods is that the word with similar semantic meanings often co-occur in the given corpus. Extensive experiments have validated the effectiveness of the corpus-based method in polarity classification task (Turney, 2002; Kaji and Kitsuregawa, 2007; Velikovich et al., 2010). For example, PMI is a well-known similarity measurement (Turney, 2002), which makes use of the whole Web as the corpus, and utilizes the search engine hits number to estimate the co-occurrence probability of the give word pairs. The PMI based method has achieved promising results. However, according to Kanayama's investigation, only 60% co-occurrences in the same window in Web pages reflect the same sentiment orientation (Kanayama and Nasukawa, 2006). Therefore, we may ask the question whether the choosing of corpus can change the performance of sim and is there any better corpus than the Web page data for measuring the sentiment similarity?

Everyday, enormous numbers of tweets that contain people's rich sentiments are published in Twitter. The Twitter may be a good source for measuring the sentiment similarity. Compared with the Web page data, the tweets have a higher rate of subjective text posts. The length limitation can guarantee the polarity consistency of each tweet. Moreover, the tweets contain graphical emoticons, which can be

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 897–902,
Seattle, Washington, USA, 18-21 October 2013.

considered as natural sentiment labels for the corresponding tweets in Twitter. In this paper, we attempt to empirically evaluate the performance of different corpora in sentiment similarity measurement task. As far as we know, no work is done on this topic.

2 The Characteristics of Twitter Data

As the world's second largest SNS website, at the end of 2012 Twitter had aggregated more than 500 million registered users, among which 200 million were active users . More than 400 million tweets are posted every day.

Several examples of typical posts from Twitter are shown below.

(1) *She had a headache and feeling light headed with no energy.* :(

(2) *@username Nice work! Looks like you had a fun day. I'm headed there Sat or Sun.* :)

(3) *I seen the movie on Direc Tv. I ordered it and I really liked it. I can't wait to get it for blu ray! Excellent work Rob!*

We observe that comparing with the other corpus, the Twitter data has several advantages in measuring the sentiment similarity.

Large. Users like to record their personal feelings and talk about the trend topics in Twitter (Java et al., 2007; Kwak et al., 2010). So there are huge amount of subjective texts with various topics generated in the millions of tweets everyday. Further more, the flexible Twitter API makes these data easy to access and collect.

Length Limitation. Twitter has a length limitation of 140 characters. Users have limited space to express their feelings. So the sentiments in tweets are usually concise, straightforward and polarity consistent.

Emoticons. Users tend to utilize emoticons to emphasize their sentiment feelings. According to the statistics, about 8.1% tweets contain at least one emoticon (Yang and Leskovec, 2011). Since the tweets have the length limitation, the sentiments expressed in these short texts are usually consistent with the embedded emoticons, such as the word *fun* and *headache* in above examples.

In addition to the above advantages, there are also some disadvantages for measuring sentiment similarity using Twitter data. The spam tweets that caused by advertisements may add noise and bias during the similarity measurement. The short length may also bring in lower co-occurrence probability of words. Some words may not co-occur with each other when the corpus is small. These disadvantages set obstacles for measuring sentiment similarity by using Twitter data as corpus. In the experiment section, we will see if we can overcome these drawbacks and get benefit from the advantages of Twitter data.

3 The Corpus-based Sentiment Similarity Measurements

The intuition behind the corpus-based semantic similarity measuring method is that the words with similar meanings tend to co-occur in the corpus. Given the word w_i, w_j, we use the notation $P(w_i)$ to denote the occurrence counts of word w_i in the corpus $\mathcal{C}$. $P(w_i, w_j)$ denotes the co-occurrence counts of word w_i and w_j in $\mathcal{C}$. In this paper we employ the corpus-based version of the three well-known similarity measurements: Jaccard, Dice and PMI.

$$CorpusJaccard(w_i, w_j) = \frac{P(w_i,w_j)}{P(w_i)+P(w_j)-P(w_i,w_j)} \quad (2)$$

$$CorpusDice(w_i, w_j) = \frac{2 \times P(w_i, w_j)}{P(w_i) + P(w_j)} \quad (3)$$

$$CorpusPMI(w_i, w_j) = \log_2\left(\frac{\frac{P(w_i,w_j)}{N}}{\frac{P(w_i)}{N}\frac{P(w_j)}{N}}\right) \quad (4)$$

In Formula (4), N is the number of documents in the corpus $\mathcal{C}$. The above similarity measurements may have their own strengths and weaknesses. In this paper, we utilize these classical measurements to evaluate the quality of the corpus in polarity classification task.

Google is the world's largest search engine, which has indexed a huge number of Web pages. Using the extreme large indexed Web pages as corpus, Cilibrasi and Vitanyi (2007) presented a method for measuring similarity between words and phrases based on information distance and Kolmogorov complexity. The search result page counts of Google were utilized to estimate the occurrence frequencies of the words in the corpus. Suppose w_i, w_j represent the candidate words, the Normalized Google

Distance is defined as:

$$NGD(w_i, w_j) = \frac{\max\{\log P(w_i), \log P(w_j)\} - \log P(w_i, w_j)}{\log N - \min\{\log P(w_i), \log P(w_j)\}} \quad (5)$$

where $P(w_i)$ denotes page counts returned by Google using w_i as keyword; $P(w_i, w_j)$ denotes the page counts by using w_i and w_j as joint keywords; N is the number of Web pages indexed by Google. Cilibrasi and Vitanyi have validated the effectiveness of Google distance in measuring the semantic similarity between concept words.

Based on the above formulas, we compare the Twitter data with the Web and Wikipedia data as the similarity measurement corpus. Given a candidate word w, we firstly measure its sentiment similarity with a positive seed word and a negative seed word respectively in Formula (1), and the difference of sim is used to further detect the polarity of w. The above four similarity measurements serve as sim with Web, Wikipedia and Twitter data as corpus. Turney (2002) chose *excellent* and *poor* as seed words. However, using isolated seed words may cause the bias problem. Therefore, we further select two groups of seed words that are lack of sensitivity to context and form a positive seed set PS and a negative seed set NS (Turney, 2003). The Formula (1) can be rewritten as:

$$\overline{SO}(w) = \sum_{se_p \in PS} sim(w, se_p) - \sum_{se_n \in NS} sim(w, se_n) \quad (6)$$

Based on the Formula(6) and the sentiment seed words, we can measure the sentiment polarity of the given candidate words.

4 Experiment

4.1 Experiment Setup

Corpus Preparing. The Twitter corpus corresponds to the *476 million Twitter tweets* (Yang and Leskovec, 2011), which includes over 476 million Twitter posts from 20 million users, covering a 7 month period from June 1, 2009 to December 31, 2009. We filter out the non-English tweets and the spam tweets that have only few words with URLs. The tweets that contain three or more trending topics are also removed. Finally, we construct the Twitter corpus that consists of 266.8 million English tweets. For calculating page counts in Web data, the candidate words were launched to Google from February 2013 to April 2013. We also conduct the experiments on the Google Web 1T data that consists of Google n-gram counts (frequency of occurrence of each n-gram) for $1 \leq n \leq 5$ (Brants and Franz, 2006). The Web 1T data provides a nice approximation to the word co-occurrence statistics in Web pages in a predefined window size ($1 \leq n \leq 5$). For example, the 5 gram Web1T data means the co-occurrence window size is 5. The English Wikipedia dump [1] we used was extracted at the end of March 2013, which contained more than 13 million articles. We extracted the plain texts of the Wikipedia data as the training corpus for the Formula (6).

Evaluation Method. Two well-know sentiment lexicons are utilized as gold standard for polarity classification task. The statistics of Liu's sentiment lexicon (Liu et al., 2005) and MPQA subjectivity lexicon (Wilson et al., 2005) are shown in Table 1. For each word w in the lexicons, we employ the Formula (6) to calculate the word's polarity using different corpora. If $\overline{SO}(w) > 0$, the word w is classified into the positive category. Otherwise if $\overline{SO}(w) < 0$, it is classified into the negative category. The accuracy of the classification result is used to measure the quality of the corpus.

	Positive#	Negative#
Liu	2,006	4,783
MPQA	2,304	4,153

Table 1: Lexicon size

4.2 Experiment Results

Firstly, we chose the seed words *excellent* and *poor* as Turney's (2002) settings. The polarity classification accuracies are shown in Table 2.

In Table 2, Google, Web1T, Wikipedia, Twitter represent the corpora that used in the experiment; CJ, CD, CP, GD represent the Formula (2) to Formula (5) respectively. We can see from the Table 2 that the Twitter based method can achieve the best performance. The rich sentiment information and

[1] http://en.wikipedia.org/

Lexicon	Corpus	CJ	CD	CP	GD
Liu	Google	0.5116	0.5117	0.5064	0.5076
	Web1T-5gram	0.3903	0.3903	0.3897	0.3864
	Web1T-4gram	0.3771	0.3771	0.3772	0.3227
	Wikipedia	0.5280	0.5280	0.5350	0.5412
	Twitter	**0.5567**	**0.5567**	**0.5635**	**0.5635**
MPQA	Google	0.4897	0.4890	0.4891	0.4864
	Web1T-5gram	0.3843	0.3843	0.3837	0.3783
	Web1T-4gram	0.3729	0.3729	0.3714	0.3225
	Wikipedia	0.5181	0.5181	0.5380	0.5344
	Twitter	**0.5421**	**0.5421**	**0.5493**	**0.5494**

Table 2: Polarity classification accuracies using *excellent* and *poor* as seed words

Lexicon	Corpus	CJ	CD	CP	GD
Liu	Google	0.4859	0.4936	0.4884	0.5060
	Web1T-5gram	0.5785	0.5785	0.3963	0.5782
	Web1T-4gram	0.5766	0.5766	0.3872	0.5775
	Wikipedia	0.6226	0.6225	0.5957	0.6145
	Twitter	0.6678	0.6678	0.6917	0.6457
	Twitter$^+$	**0.6921**	**0.6921**	**0.7273**	**0.6599**
MPQA	Google	0.5108	0.5225	0.5735	0.5763
	Web1T-5gram	0.5737	0.5737	0.4225	0.5718
	Web1T-4gram	0.5749	0.5749	0.3329	0.4797
	Wikipedia	0.6086	0.6085	0.5773	0.5985
	Twitter	0.6431	0.6431	0.6671	0.6253
	Twitter$^+$	**0.6665**	**0.6665**	**0.7001**	**0.6383**

Table 3: Polarity classification accuracies using the seed word groups

natural window size (140 characters) have a positive impact on determining the word's polarity. The Google based method gets a lower accuracy, this may be due to the length of Web documents which can not usually guarantee the semantic consistency in the returned data. Even though two words appear in one page (returned by Google), they might not be semantically related. Furthermore, the Google based method is time-consuming, because we have to periodically send queries in order to avoid being blocked by Google. The Web1T based method gets a much worse accuracy. After detailed analysis, we find that although the small window size (4 or 5) can guarantee the semantic consistency, the short length also brings in lower co-occurrence probability. Statistics show that about 38% $\overline{SO}$ values are zero when using Web1T corpus. Due to the short length, the Twitter data also suffers from the low co-occurrence problem.

To tackle the low co-occurrence problem, the seed word sets are selected as Turney's (2003) settings. The positive word set *PS*={*good, nice, excellent, positive, fortunate, correct, superior*} and negative word set *NS* = {*bad, nasty, poor, negative, unfortunate, wrong, inferior*} for the Formula (6). These seed words have been verified to be effective in Turney's paper for polarity classification. The experiment results are shown in Table 3.

Table 3 shows that the performance of Twitter corpus is much improved since the multiple seed words alleviate the problem of low co-occurrence probability in tweets. Generally, when using the seed word groups the Twitter can achieve a much better performance than all the other corpora. The improvements are statistically significant (p-value < 0.05). We further add the emoticons ':)' and ':(' into the seed word groups, denoted by Twitter$^+$ in Table 3. The emoticons are natural sentiment labels. We can see that the performances are further improved by considering emoticons as seed words. The above experiment results have validated the effectiveness of Twitter data as a better corpus for measuring the sentiment similarity. The results also reveal the potential usefulness of Twitter corpus in semantic similarity measurement.

5 Related Work

Detecting the polarity of words is the fundamental problem for most of sentiment analysis tasks (Hatzivassiloglou and McKeown, 1997; Pang and Lee, 2007; Feldman, 2013).

Many methods have been proposed to measure the words' or short texts similarity based on large corpus (Sahami and Heilman, 2006; Yih and Meek, 2007; Gabrilovich and Markovitch, 2007). Bollegala *et al*. (2011) submitted the word to the search engine, and the related result pages were employed to represent the meaning of the original word. Mihalcea *et al*. (2006) proposed a method to measure the semantic similarity of words or short texts, considering both corpus-based and knowledge-based information. Although the previous algorithms have achieved promising results, there are no work done on evaluating the quality of different corpora.

Mohtarami *et al*. (2012; 2013a; 2013b) introduced the concept of sentiment similarity, which was considered as different from the traditional semantic similarity, and more focused on revealing the underlying sentiment relations between words. Mo-

htarami *et al*. (2013b) proposed a hidden emotional model to calculating the sentiment similarity of word pairs. However, the impact of the different corpora is not considered for this task.

Mohammad *et al*. (2013) generated word-sentiment association lexicons from Tweets with the help of hashtags and emoticons. Pak and Paroubek (2010) collected tweets with happy and sad emoticons as training corpus, and built sentiment classifier based on traditional machine learning methods. Brody and Diakopoulos (2011) showed that lengthening was strongly associated with subjectivity and sentiment in tweets. Davidov *et al*. (2010) treated 50 Twitter tags and 15 smileys as sentiment labels and a supervised sentiment classification framework was proposed to classify the tweets. The previous literatures have showed that the emoticons can be treated as natural sentiment labels of the tweets.

6 Conclusion and Future Work

The quality of corpus may affect the performance of sentiment similarity measurement. In this paper, we compare the Twitter data with the Google, Web1T and Wikipedia data in polarity classification task. The experiment results validate that when using the seed word groups the Twitter can achieve a much better performance than the other corpora and adding emoticons as seed words can further improve the performance. It is observed that the twitter corpus is a potential good source for measuring sentiment similarity between words. In future work, we intend to design new similarity measurements that can make best of the advantages of Twitter data.

Acknowledgments

This research is partially supported by General Research Fund of Hong Kong (No. 417112) and Shenzhen Fundamental Research Program (JCYJ20130401172046450). This research is also supported by the State Key Development Program for Basic Research of China (Grant No. 2011CB302200-G), State Key Program of National Natural Science of China (Grant No. 61033007), National Natural Science Foundation of China (Grant No. 61370074, 61100026), and the Fundamental Research Funds for the Central Universities (N120404007).

References

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. *Linguistic Data Consortium*, ISBN: 1-58563-397-6, Philadelphia.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 562–570, Edinburgh, UK, ACL.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2011. A Web Search Engine-Based Approach to Measure Semantic Similarity between Words. *IEEE Transactions on Knowledge and Data Engineering*, 23(7): 977–990.

Rudi Cilibrasi and Paul Vitanyi. 2007. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3): 370–383.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 241–249, Beijing, China, ACL.

Ronen Feldman. 2013. Techniques and Applications for Sentiment Analysis. *Communications of the ACM*, 56(4):82–89.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611, Hyderabad, India.

Vasileios Hatzivassiloglou and Kathleen McKeown. 1997. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 174–181, Madrid, Spain, ACL.

Akshay Java, Xiaodan Song, Tim Finin, and Belle L. Tseng. 2007. Why We Twitter: An Analysis of a Microblogging Community. In *Proceedings of the 9th International Workshop on Knowledge Discovery on the Web and 1st International Workshop on Social Networks Analysis*, pages 118–138, San Jose, CA, USA, Springer.

Nobuhiro Kaji and Masaru Kitsuregawa. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1075–1083, Prague, Czech Republic, ACL.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully Automatic Lexicon Expansion for Domain-oriented

Sentiment Analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355–363, Sydney, Australia, ACL.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue B. Moon. 2010. What is Twitter, a Social Network or a News Media? In *Proceedings of the 19th International Conference on World Wide Web*, pages 591–600, Raleigh, North Carolina, USA, ACM.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 768–774, Montreal, Quebec, Canada, ACL.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion Observer: Analyzing and Comparing Opinions on the Web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351, Chiba, Japan, ACM.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*, pages 775–780, Boston, Massachusetts, USA, AAAI Press.

Mitra Mohtarami, Hadi Amiri, Man Lan, Thanh Phu Tran, and Chew Lim Tan. 2012. Sense Sentiment Similarity: An Analysis. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1706–1712, Toronto, Ontario, Canada, AAAI Press.

Mitra Mohtarami, Man Lan, and Chew Lim Tan. 2013a. From Semantic to Emotional Space in Probabilistic Sense Sentiment Analysis. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 711–717, Bellevue, Washington, USA, AAAI Press.

Mitra Mohtarami, Man Lan, and Chew Lim Tan. 2013b. Probabilistic Sense Sentiment Similarity through Hidden Emotions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 983–992, Sofia, Bulgaria, ACL.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises*, Atlanta, Georgia, USA, ACL.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *Proceedings of the 2010 International Conference on Language Resources and Evaluation*, pages 1320–1326, Valletta, Malta, ELRA.

Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information based Measure and Its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 11:95–130.

Mehran Sahami and Timothy D. Heilman. 2006. A Web-based Kernel Function for Measuring the Similarity of Short Text Snippets. In *Proceedings of the 15th international conference on World Wide Web*, pages 377–386, Edinburgh, Scotland, UK, ACM.

Bo Pang and Lillian Lee. 2007. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Peter D. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, PA, USA, ACL.

Peter Turney and Michael Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transaction Information System*, 21(4): 315–346.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan T. McDonald. The Viability of Web-derived Polarity Lexicons. In *Proceedings of the 2010 North American Chapter of the Association of Computational Linguistics*, pp. 777–785, Los Angeles, California, USA, ACL.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of the 2005 Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada, ACL.

Jaewon Yang and Jure Leskovec. 2011. Patterns of Temporal Variation in Online Media. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining*, pages 177–186, Hong Kong, China, ACM.

Wen-tau Yih and Christopher Meek. 2007. Improving Similarity Measures for Short Segments of Text. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 1489–1494, Vancouver, British Columbia, Canada, AAAI Press.

Implicit Feature Detection via a Constrained Topic Model and SVM

Wei Wang [*], **Hua Xu** [*] and **Xiaoqiu Huang** [†]
[*]State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, China
[†]Beijing University of Posts and Telecommunications, Beijing 100876, China
ww880412@gmail.com, xuhua@tsinghua.edu.cn, alexalexhxqhxq@gmail.com

Abstract

Implicit feature detection, also known as implicit feature identification, is an essential aspect of feature-specific opinion mining but previous works have often ignored it. We think, based on the explicit sentences, several Support Vector Machine (SVM) classifiers can be established to do this task. Nevertheless, we believe it is possible to do better by using a constrained topic model instead of traditional attribute selection methods. Experiments show that this method outperforms the traditional attribute selection methods by a large margin and the detection task can be completed better.

1 Introduction

Feature-specific opinion mining has been well defined by Ding and Liu(2008). Example 1 is a cell phone review in which two features are mentioned.

Example 1 *This cell phone is fashion in appearance, and it is also very cheap.*

If a feature appears in a review directly, it is called an *explicit feature*. If a feature is only implied, it is called an *implicit feature*. In Example 1, *appearance* is an explicit feature while *price* is an implicit feature, which is implied by *cheap*. Furthermore, an *explicit sentence* is defined as a sentence containing at least one explicit feature, and an *implicit sentence* is the sentence only containing implicit features. Thus, the first sentence is an explicit sentence, while the second is an implicit one.

This paper proposes an approach for implicit feature detection based on SVM and Topic Model(TM). The Topic Model, which incorporated into constraints based on the pre-defined product feature, is established to extract the training attributes for SVM. In the end, several SVM classifiers are constructed to train the selected attributes and utilized to detect the implicit features.

2 Related Work

The definition of implicit feature comes from Liu et al. (2005)'s work. Su et al. (2006) used Pointwise Mutual Information (PMI) based semantic association analysis to identify implicit features, but no quantitative experimental results were provided. Hai et al. (2011) used co-occurrence association rule mining to identify implicit features. However, they only dealt with opinion words and neglected the facts. Therefore, in this paper, both the opinions and facts will be taken into account.

Blei et al. (2003) proposed the original LDA using EM estimation. Griffiths and Steyvers (2004) applied Gibbs sampling to estimate LDA's parameters. Since the inception of these works, many variations have been proposed. For example, LDA has previously been used to construct attributes for classification; it often acts to reduce data dimension(Blei and Jordan, 2003; Fei-Fei and Perona, 2005; Quelhas et al., 2005). Here, we modify LDA and adopt it to select the training attributes for SVM.

3 Model Design

3.1 Introduction to LDA

We briefly introduce LDA, following the notation of Griffiths(Griffiths and Steyvers, 2004). Given D

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 903–907,
Seattle, Washington, USA, 18-21 October 2013.

documents expressed over *W* unique words and *T* topics, LDA outputs the document-topic distribution θ and topic-word distribution φ, both of which can be obtained with Gibbs Sampling. For this scheme, the core process is the topic updating for each word in each document according to Equation 1.

$$P(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \alpha, \beta) = \\ (\frac{n_{-i,j}^{(w_i)} + \beta}{\sum_{w'}^{W} n_{-i,j}^{(w')} + W\beta})(\frac{n_{-i,j}^{(d_i)} + \alpha}{\sum_{j}^{T} n_{-i,j}^{(d_i)} + T\alpha}) \quad (1)$$

where $z_i = j$ represents the assignment of the i^{th} word in a document to topic j, z_{-i} represents all the topic assignments excluding the i^{th} word. $n_j^{(w')}$ is the number of instances of word w' assigned to topic j and $n_j^{(d_i)}$ is the number of words from document d_i assigned to topic j, the $-i$ notation signifies that the counts are taken omitting the value of z_i. Furthermore, α and β are hyper-parameters for the document-topic and topic-word Dirichlet distributions, respectively. After N iterations of Gibbs sampling for all words in all documents, the distribution θ and φ are finally estimated using Equations 2 and 3.

$$\phi_j^{(w_i)} = \frac{n_j^{(w_i)} + \beta}{\sum_{w'}^{W} n_j^{(w')} + W\beta} \quad (2)$$

$$\theta_j^{(d_i)} = \frac{n_j^{(d_i)} + \alpha}{\sum_{j}^{T} n_j^{(d_i)} + T\alpha} \quad (3)$$

3.2 Framework

Algorithm 1 summarizes the main steps. When a specific product and the reviews are provided, the explicit sentences and corresponding features are extracted(Line 1) by word segmentation, part-of-speech(POS) tagging and synonyms feature clustering. Then the prior knowledge are drawn from the explicit sentences automatically and integrated into the constrained topic model((Line 3 - Line 5). The word clusters are chosen as the training attributes(Line 6). Finally, several SVM classifiers are generated and applied to detect implicit features(Line 7 - Line 12).

Algorithm 1 Implicit Feature Detection

1: $ES \leftarrow$ extract explicit sentence set
2: $NES \leftarrow$ non-explicit sentence set
3: $CS \leftarrow$ constraint set from ES
4: $CPK \leftarrow$ correlation prior knowledge from ES
5: $ETM \leftarrow$ ConstrainedTopicModel(T,ES,CS,CPK)
6: $TA \leftarrow$ select training attributes from ETM
7: **for** each f_i in feature clusters **do**
8: $TD_i \leftarrow$ GenerateTrainingData(TA_i,ES)
9: $C_i \leftarrow$ BuildClassificationModelBySVM(TD_i)
10: $PR_i \leftarrow$ positive result of Classify(C_i,NES)
11: the feature of sentence in $PR_i \leftarrow f_i$
12: **end for**

3.3 Prior Knowledge Extraction and Incorporation

It is obvious that the pre-existing knowledge can assist to produce better and more significant clusters. In our work, we use a constrained topic model to select attributes for each product features. Each topic is first pre-defined a product feature. Then two types of prior knowledge, which are derived from the pre-defined product features, are extracted automatically and incorporated: must-link/cannot-link and correlation prior knowledge.

3.3.1 Must-link and Cannot-link

Must-link: It specifies that two data instances must be in the same cluster. Here is the must-link from an observation: as "*cheap*" to "*price*", some words must be associated with a feature. In order to mine these words, we compute the co-occurrence degree by *frequency*PMI(f,w)*, whose formula is as following: $P_{f\&w} * \log_2 \frac{P_{f\&w}}{P_f P_w}$, where P is the probability of subscript occurrence in explicit sentences, f is the feature, w is the word, and $f\&w$ means the co-occurrence of f and w. A higher value of *frequency*PMI* signifies that w often indicates f. For a feature f_i, the top five words and f_i constitute must-links. For example, the co-occurrence of "*price*" and "*cheap*" is very high, then the must-link between "*price*" and "*cheap*" can be identified.

Cannot-link: It specifies that two data instances cannot be in the same cluster. If a word and a feature never co-occur in our corpus, we assume them to form a cannot-link. For example, the word *low-cost* has never co-occurred with the product feature *screen*, so they constitute a cannot-link in our cor-

pus.

In this paper, the pre-defined process, must-link, and cannot-link are derived from Andrzejewski and Zhu (2009)'s work, all must-links and cannot-links are incorporated our constrained topic model. We multiply an indicator function $\delta(w_i, z_j)$, which represents a hard constraint, to the Equation 1 as the final probability for topic updating (see Equation 4).

$$P(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \alpha, \beta) = \delta(w_i, z_j)(\frac{n_{-i,j}^{(w_i)} + \beta}{\sum_{w'}^{W} n_{-i,j}^{(w')} + W\beta})(\frac{n_{-i,j}^{(d_i)} + \alpha}{\sum_{j}^{T} n_{-i,j}^{(d_i)} + T\alpha}) \quad (4)$$

As illustrated by Equations 1 and 4, $\delta(w_i, z_j)$, which represents intervention or help from pre-existing knowledge of must-links and cannot-links, plays a key role in this study. In the topic updating for each word in each document, we assume that the current word is w_i and its linked feature topic set is $Z^{(w_i)}$, then for the current topic z_j, $\delta(w_i, z_j)$ is calculated as follows:

1. If w_i is constrained by must-links and the linked feature belongs to $Z^{(w_i)}$, $\delta(w_i, z_j|z_j \in Z^{(w_i)}) = 1$ and $\delta(w_i, z_j|z_j \notin Z^{(w_i)}) = 0$.

2. If w_i is constrained by cannot-links and the linked feature belongs to $Z^{(w_i)}$, $\delta(w_i, z_j|z_j \in Z^{(w_i)}) = 0$ and $\delta(w_i, z_j|z_j \notin Z^{(w_i)}) = 1$.

3. In other cases, $\delta(w_i, z_j|j = 1, \ldots, T) = 1$.

3.3.2 Correlation Prior Knowledge

In view of the explicit product feature of each topic, the association of the word and the feature to topic-word distribution should be taken into account. Therefore, Equation 2 is revised as the following:

$$\phi_j^{(w_i)} = \frac{(1 + C_{w_i,j})(n_j^{(w_i)}) + \beta}{\sum_{w'}^{W}(1 + C_{w',j})(n_j^{(w')}) + W\beta} \quad (5)$$

where $C_{w',j}$ reflects the correlation of w' with the topic j, which is centered on the product feature f_{z_j}. The basic idea is to determine the association of w' and f_{z_j}, if they have the high relevance, $C_{w',j}$ should be set as a positive number. Otherwise, if we can determine w' and f_{z_j} are irrelevant, $C_{w',j}$ should be set as a positive number. In this paper, we attempt to using PMI or dependency relation to judge the relevance. For word w' and feature f_{z_j}:

1. Dependency relation judgement: If w' as parent node in the syntax tree mainly co-occurs with f_{z_j}, $C_{w',j}$ will be set positive. If w' mainly co-occurs with several features including f_{z_j}, $C_{w',j}$ will be set negative. Otherwise, $C_{w',j}$ will be set 0.

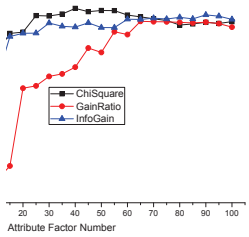
2. PMI judgement: If w' mainly co-occurs with f_{z_j} and $PMI(w', f_{z_j})$ is greater than the given value, $C_{w',j}$ will be set positive. Otherwise, $C_{w',j}$ will be set negative.

3.4 Attribute Selection

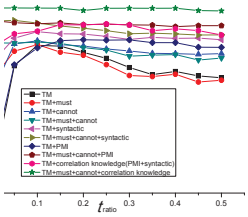
Some words, such as "*good*", can modify several product features and should be removed. In the result of run once, if a word appears in the topics which relates to different features, it is defined as a **conflicting word**. If a term is thought to describe several features or indicate no features, it is defined as a **noise word** .

When each topic has been pre-allocated, we run the explicit topic model 100 times. If a word turns into a conflicting word T_{cw} times(T_{cw} is set to 20), we assume that it is a noise word. Then the noise word collection is obtained and applied to filter the explicit sentences. Actually, here 100 is just an estimated number. And for T_{cw}, when it is between 15 and 25, the result is same, and when it exceeds 25, the result does not change a lot. The most important part to filter noise words is the correlation computation. So the experiment can work well with only estimated parameters.

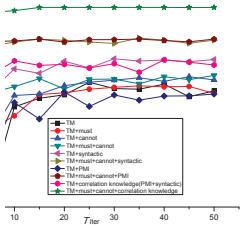
Next, By integrating pre-existing knowledge, the explicit topic model, which runs T_{iter} times, severs as attribute selection for SVM. In every result for each topic cluster, we remove the least four probable of word groups and merge the results by the pre-defined product feature. For a feature, if a word appears in its topic words more than $T_{iter} * t_{ratio}$ times, it is selected as one of the training attributes for the feature. In the end, if an attribute associates with different features, it is deleted.



(a) SVM based on traditional attribute selection method



(b) our constrained topic model by different t_{ratio} ($T_{iter} = 20$)



(c) our constrained topic model by different T_{iter} ($t_{ratio} = 0.1$)

Figure 1: Performance of different cases

3.5 Implicit Feature Detection via SVM

After completing attribute selection, vector space model(VSM) is applied to the selected attributes on the explicit sentences. For each feature f_i, a SVM classifier C_i is adopted. In train-set, the positive cases are the explicit sentences of f_i, and the negative cases are the other explicit sentences. For a non-explicit sentence, if the classification result of C_i is positive, it is an implicit sentence which implies f_i.

4 Evaluation of Experimental Results

4.1 Data Sets

There has no standard data set yet, we crawled the experiment data, which included reviews about a cellphone, from a famous Chinese shopping website[1]. The data contains 14218 sentences. The feature of each sentence was manually annotated by two research assistants. A handful of sentences which were annotated inconsistently were deleted. Table 1 depicts the data set which is evaluated. Other features were ignored because of their rare appearance.

Here are some explanations: (1)The sentences containing several explicit features were not added to the train-set. (2) A tiny number of sentences contain both explicit and implicit features, and they can only be regarded as explicit sentences. (3) The training set contains 3140 explicit sentences, the test set contains 7043 non-explicit sentences and more than 5500 sentences have no feature. (4) According to the ratio among the explicit sentences(6:1:2:3:1:2), it is reasonable that the most suitable number of topics should be 14. For example, the ratio of the product feature *screen* is 6, so we can assign the feature to topic 0,1,2,3,4,5. In our experiment, the performance of algorithm 1 is evaluated using F-measure. (5) Although the size of dataset is limited, out proposed is based on the constraint-based topic model, which has been widely used in different NLP fields. So, our approach can generalize well in different datasets. Of course, more high quality data will be collected to do the experiment in the future.

[1] http://www.360buy.com/

Table 1: Experiment data

Features	Explicit	Implicit	Total
screen	1165	244	1409
quality	199	83	282
battery	456	205	661
price	627	561	1188
appearance	224	167	391
software	469	129	598

4.2 Experimental Results

Figure 1a depicts the performance of using traditional attribute selection methods on SVM. Using χ^2 *test* on SVM can achieve the best performance, which is about 66.7%. In our constrained topic model, we use different T_{iter} and t_{ratio}. We conducted experiments by incorporating different types prior knowledge. From Figure 1b and 1c, we conclude that: (1)All these methods perform much better than the traditional feature selection methods, the improvements are more than 6%. (2)The reason for the little improvement of must-links is that the topic clusters have already obtained these linked word-

s. (3)All the pre-existing knowledge performs best and shows 3% improvement over non prior knowledge. (4)Different types of prior knowledge have different impact on the stabilities of different parameters. (5)As we have expected, by combing all prior knowledge, the best performance can reach 77.78%. Furthermore, as t_{ratio} or T_{iter} changes, our constrained topic model incorporating all prior knowledge look like very stable.

5 Conclusions

In this paper, we adopt a constrained topic model incorporating prior knowledge to select attribute for SVM classifiers to detect implicit features. Experiments show this method outperforms the attribute feature selection methods and detect implicit features better.

6 Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No: 61175110) and National Basic Research Program of China (973 Program, Grant No: 2012CB316305).

References

David Andrzejewski and Xiaojin Zhu. 2009. Latent dirichlet allocation with topic-in-set knowledge. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 43–48. Association for Computational Linguistics.

D.M. Blei and M.I. Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 127–134. ACM.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 231–240, New York, NY, USA. ACM.

L. Fei-Fei and P. Perona. 2005. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE.

T.L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Z. Hai, K. Chang, and J. Kim. 2011. Implicit feature identification via co-occurrence association rule mining. *Computational Linguistics and Intelligent Text Processing*, pages 393–404.

B. Liu, M. Hu, and J. Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM.

P. Quelhas, F. Monay, J.M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. Van Gool. 2005. Modeling scenes with local descriptors and latent aspects. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 883–890. IEEE.

Q. Su, K. Xiang, H. Wang, B. Sun, and S. Yu. 2006. Using pointwise mutual information to identify implicit features in customer reviews. *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead*, pages 22–30.

Online Learning for Inexact Hypergraph Search

Hao Zhang
Google
haozhang@google.com

Liang Huang **Kai Zhao**
City University of New York
{lhuang@cs.qc,kzhao@gc}.cuny.edu

Ryan McDonald
Google
ryanmcd@google.com

Abstract

Online learning algorithms like the perceptron are widely used for structured prediction tasks. For sequential search problems, like left-to-right tagging and parsing, beam search has been successfully combined with perceptron variants that accommodate search errors (Collins and Roark, 2004; Huang et al., 2012). However, perceptron training with inexact search is less studied for bottom-up parsing and, more generally, inference over hypergraphs. In this paper, we generalize the violation-fixing perceptron of Huang et al. (2012) to hypergraphs and apply it to the cube-pruning parser of Zhang and McDonald (2012). This results in the highest reported scores on WSJ evaluation set (UAS 93.50% and LAS 92.41% respectively) without the aid of additional resources.

1 Introduction

Structured prediction problems generally deal with exponentially many outputs, often making exact search infeasible. For sequential search problems, such as tagging and incremental parsing, beam search coupled with perceptron algorithms that account for potential search errors have been shown to be a powerful combination (Collins and Roark, 2004; Daumé and Marcu, 2005; Zhang and Clark, 2008; Huang et al., 2012). However, sequential search algorithms, and in particular left-to-right beam search (Collins and Roark, 2004; Zhang and Clark, 2008), squeeze inference into a very narrow space. To address this, Huang (2008) formulated constituency parsing as approximate bottom-up inference in order to compactly represent an exponential number of outputs while scoring features of arbitrary scope. This idea was adapted to graph-based dependency parsers by Zhang and McDonald (2012) and shown to outperform left-to-right beam search.

Both these examples, bottom-up approximate dependency and constituency parsing, can be viewed as specific instances of inexact hypergraph search. Typically, the approximation is accomplished by cube-pruning throughout the hypergraph (Chiang, 2007). Unfortunately, as the scope of features at each node increases, the inexactness of search and its negative impact on learning can potentially be exacerbated. Unlike sequential search, the impact on learning of approximate hypergraph search – as well as methods to mitigate any ill effects – has not been studied. Motivated by this, we develop online learning algorithms for inexact hypergraph search by generalizing the violation-fixing percepron of Huang et al. (2012). We empirically validate the benefit of this approach within the cube-pruning dependency parser of Zhang and McDonald (2012).

2 Structured Perceptron for Inexact Hypergraph Search

The structured perceptron algorithm (Collins, 2002) is a general learning algorithm. Given training instances $(x, \hat{y})$, the algorithm first solves the decoding problem $y' = \mathbf{argmax}_{y \in \mathcal{Y}(x)} \mathbf{w} \cdot \mathbf{f}(x, y)$ given the weight vector $\mathbf{w}$ for the high-dimensional feature representation $\mathbf{f}$ of the mapping (x, y), where y' is the prediction under the current model, $\hat{y}$ is the gold output and $\mathcal{Y}(x)$ is the space of all valid outputs for input x. The perceptron update rule is simply: $\mathbf{w}' = \mathbf{w} + \mathbf{f}(x, \hat{y}) - \mathbf{f}(x, y')$.

The convergence of original perceptron algorithm relies on the $\mathbf{argmax}$ function being exact so that the condition $\mathbf{w} \cdot \mathbf{f}(x, y') > \mathbf{w} \cdot \mathbf{f}(x, \hat{y})$ (modulo ties) always holds. This condition is called a *violation* because the prediction y' scores higher than the correct label $\hat{y}$. Each perceptron update moves weights

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 908–913,
Seattle, Washington, USA, 18-21 October 2013.

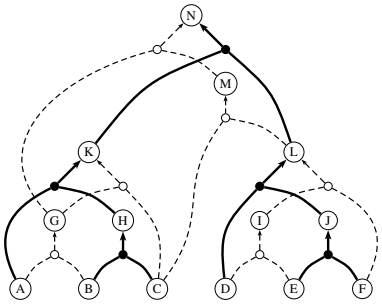


Figure 1: A hypergraph showing the union of the gold and Viterbi subtrees. The hyperedges in bold and dashed are from the gold and Viterbi trees, respectively.

away from y' and towards $\hat{y}$ to fix such violations. But when search is inexact, y' could be suboptimal so that sometimes $\mathbf{w} \cdot \mathbf{f}(x, y') < \mathbf{w} \cdot \mathbf{f}(x, \hat{y})$. Huang et al. (2012) named such instances *non-violations* and showed that perceptron model updates for non-violations nullify guarantees of convergence. To account for this, they generalized the original update rule to select an output y' within the pruned search space that scores higher than $\hat{y}$, but is not necessarily the highest among all possibilities, which represents a true violation of the model on that training instance. This *violation fixing perceptron* thus relaxes the $\mathbf{argmax}$ function to accommodate inexact search and becomes provably convergent as a result.

In the sequential cases where $\hat{y}$ has a linear structure such as tagging and incremental parsing, the violation fixing perceptron boils down to finding and updating along a certain prefix of $\hat{y}$. Collins and Roark (2004) locate the earliest position in a chain structure where $\hat{y}_{\text{pref}}$ is worse than y'_{pref} by a margin large enough to cause $\hat{y}$ to be dropped from the beam. Huang et al. (2012) locate the position where the violation is largest among all prefixes of $\hat{y}$, where size of a violation is defined as $\mathbf{w} \cdot \mathbf{f}(x, y'_{\text{pref}}) - \mathbf{w} \cdot \mathbf{f}(x, \hat{y}_{\text{pref}})$.

For hypergraphs, the notion of prefix must be generalized to subtrees. Figure 1 shows the packed-forest representation of the union of gold subtrees and highest-scoring (Viterbi) subtrees at every gold node for an input. At each gold node, there are two incoming hyperedges: one for the gold subtree and the other for the Viterbi subtree. After bottom-up parsing, we can compute the scores for the gold subtrees as well as extract the corresponding Viterbi subtrees by following backpointers. These Viterbi subtrees need not necessarily to belong to the full Viterbi path (i.e., the Viterbi tree rooted at node (N)). An update strategy must choose a subtree or a set of subtrees at gold nodes. This is to ensure that the model is updating its weights relative to the intersection of the search space and the gold path.

Our first update strategy is called *single-node max-violation (s-max)*. Given a gold tree $\hat{y}$, it traverses the gold tree and finds the node n on which the violation between the Viterbi subtree and the gold subtree is the largest over all gold nodes. The violation is guaranteed to be greater than or equal to zero because the lower bound for the max-violation on any hypergraph is 0 which happens at the leaf nodes. Then we choose the subtree pair $(\hat{y}_n, y'_n)$ and do the update similar to the prefix update for the sequential case. For example, in Figure 1, suppose the max-violation happens at node (K), which covers the left half of the input x, then the perceptron update would move parameters to the subtree represented by nodes (B), (C), (H) and (K) and away from (A), (B), (G) and (K).

Our second update strategy is called *parallel max-violation (p-max)*. It is based on the observation that violations on non-overlapping nodes can be fixed in parallel. We define a set of *frontiers* as a set of nodes that are non-overlapping and the union of which covers the entire input string x. The frontier set can include up to $|x|$ nodes, in the case where the frontier is equivalent to the set of leaves. We traverse $\hat{y}$ bottom-up to compute the set of frontiers such that each has the max-violation in the span it covers. Concretely, for each node n, the max-violation frontier set can be defined recursively,

$$\text{ft}(n) = \begin{cases} n, & \text{if } n = \text{maxv}(n) \\ \bigcup_{n_i \in \text{children}(n)} \text{ft}(n_i), & \text{otherwise} \end{cases}$$

where $\text{maxv}(n)$ is the function that returns the node with the absolute maximum violation in the subtree rooted at n and can easily be computed recursively over the hypergraph. To make a perceptron update, we generate the max-violation frontier set for the entire hypergraph and use it to choose subtree pairs $\bigcup_{n \in \text{ft}(\text{root}(x))} (\hat{y}_n, y'_n)$, where $\text{root}(x)$ is the root of the hypergraph for input x. For example, in Figure 1, if the union of (K) and (L) satisfies the definition of ft, then the perceptron update would move feature

weights away from the union of the two Viterbi subtrees and towards their gold counterparts.

In our experiments, we compare the performance of the two violation-fixing update strategies against two baselines. The first baseline is the *standard* update, where updates always happen at the root node of a gold tree, even if the Viterbi tree at the root node leads to a non-violation update. The second baseline is the *skip* update, which also always updates at the root nodes but skips any non-violations. This is the strategy used by Zhang and McDonald (2012).

3 Experiments

We ran a number of experiments on the cube-pruning dependency parser of Zhang and McDonald (2012), whose search space can be represented as a hypergraph in which the nodes are the complete and incomplete states and the hyperedges are the instantiations of the two parsing rules in the Eisner algorithm (Eisner, 1996).

The feature templates we used are a superset of Zhang and McDonald (2012). These features include first-, second-, and third-order features and their labeled counterparts, as well as valency features. In addition, we also included a feature template from Bohnet and Kuhn (2012). This template examines the leftmost child and the rightmost child of a modifier simultaneously. All other high-order features of Zhang and McDonald (2012) only look at arcs on the same side of their head. We trained the parser with hamming-loss-augmented MIRA (Crammer et al., 2006), following Martins et al. (2010). Based on results on the English validation data, in all the experiments, we trained MIRA with 8 epochs and used a beam of size 6 per node.

To speed up the parser, we used an unlabeled first-order model to prune unlikely dependency arcs at both training and testing time (Koo and Collins, 2010; Martins et al., 2013). We followed Rush and Petrov (2012) to train the first-order model to minimize filter loss with respect to max-marginal filtering. On the English validation corpus, the filtering model pruned 80% of arcs while keeping the oracle unlabeled attachment score above 99.50%. During training only, we insert the gold tree into the hypergraph if it was mistakenly pruned. This ensures that the gold nodes are always available, which is required for model updates.

3.1 English and Chinese Results

We report dependency parsing results on the Penn WSJ Treebank and the Chinese CTB-5 Treebank. Both treebanks are constituency treebanks. We generated two versions of dependency treebanks by applying commonly-used conversion procedures. For the first English version (PTB-YM), we used the Penn2Malt[1] software to apply the head rules of Yamada and Matsumoto and the Malt label set. For the second English version (PTB-S), we used the Stanford dependency framework (De Marneffe et al., 2006) by applying version 2.0.5 of the Stanford parser. We split the data in the standard way: sections 2-21 for training; section 22 for validation; and section 23 for evaluation. We utilized a linear chain CRF tagger which has an accuracy of 96.9% on the validation data and 97.3% on the evaluation data[2]. For Chinese, we use the Chinese Penn Treebank converted to dependencies and split into train/-validation/evaluation according to Zhang and Nivre (2011). We report both unlabeled attachment scores (UAS) and labeled attachment scores (LAS), ignoring punctuations (Buchholz and Marsi, 2006).

Table 1 displays the results. Our improved cube-pruned parser represents a significant improvement over the feature-rich transition-based parser of Zhang and Nivre (2011) with a large beam size. It also improves over the baseline cube-pruning parser without max-violation update strategies (Zhang and McDonald, 2012), showing the importance of update strategies in inexact hypergraph search. The UAS score on Penn-YM is slightly higher than the best result known in the literature which was reported by the fourth-order unlabeled dependency parser of Ma and Zhao (2012), although we did not utilize fourth-order features. The LAS score on Penn-YM is on par with the best reported by Bohnet and Kuhn (2012). On Penn-S, there are not many existing results to compare with, due to the tradition of reporting results on Penn-YM in the past. Nevertheless, our result is higher than the second best by a large margin. Our Chinese parsing scores are the highest reported results.

[1] http://stp.lingfil.uu.se//∼nivre/research/Penn2Malt.html

[2] The data was prepared by André F. T. Martins as was done in Martins et al. (2013).

		Penn-YM			Penn-S			CTB-5		
	Parser	*UAS*	*LAS*	*Toks/Sec*	*UAS*	*LAS*	*Toks/Sec*	*UAS*	*LAS*	*Toks/Sec*
	Zhang and Nivre (2011)	92.9-	91.8-	†680	-	-	-	86.0-	84.4-	-
	Zhang and Nivre (reimpl.) (beam=64)	93.00	91.98	800	92.96	90.74	500	85.93	84.42	700
	Zhang and Nivre (reimpl.) (beam=128)	92.94	91.91	400	93.11	90.84	250	86.05	84.50	360
	Koo and Collins (2010)	93.04	-	-	-	-	-	-	-	-
	Zhang and McDonald (2012)	93.06	91.86	220	-	-	-	86.87	85.19	-
	Rush and Petrov (2012)	-	-	-	92.7-	-	4460	-	-	-
	Martins et al. (2013)	93.07	-	740	92.82	-	600	-	-	-
	Qian and Liu (2013)	93.17	-	180	-	-	-	87.25	-	100
	Bohnet and Kuhn (2012)	93.39	92.38	†120	-	-	-	87.5-	85.9-	-
	Ma and Zhao (2012)	93.4-	-	-	-	-	-	87.4-	-	-
cube-pruning	w/ skip	93.21	92.07	300	92.92	90.35	200	86.95	85.23	200
	w/ s-max	**93.50**	**92.41**	300	93.59	91.17	200	87.78	86.13	200
	w/ p-max	93.44	92.33	300	**93.64**	**91.28**	200	**87.87**	**86.24**	200

Table 1: Parsing results on test sets of the Penn Treebank and CTB-5. UAS and LAS are measured on all tokens except punctuations. We also include the tokens per second numbers for different parsers whenever available, although the numbers from other papers were obtained on different machines. Speed numbers marked with † were converted from sentences per second.

The speed of our parser is around 200-300 tokens per second for English. This is faster than the parser of Bohnet and Kuhn (2012) which has roughly the same level of accuracy, but is slower than the parser of Martins et al. (2013) and Rush and Petrov (2012), both of which only do unlabeled dependency parsing and are less accurate. Given that predicting labels on arcs can slow down a parser by a constant factor proportional to the size of the label set, the speed of our parser is competitive. We also tried to prune away arc labels based on observed labels for each POS tag pair in the training data. By doing so, we could speed up our parser to 500-600 tokens per second with less than a 0.2% drop in both UAS and LAS.

3.2 Importance of Update Strategies

The lower portion of Table 1 compares cube-pruning parsing with different online update strategies in order to show the importance of choosing an update strategy that accommodates search errors. The max-violation update strategies (s-max and p-max) improved results on both versions of the Penn Treebank as well as the CTB-5 Chinese treebank. It made a larger difference on Penn-S relative to Penn-YM, improving as much as 0.93% in LAS against the skip update strategy. Additionally, we measured the percentage of non-violation updates at root nodes. In the last epoch of training, on Penn-YM, there was 24% non-violations if we used the skip update strategy; on Penn-S, there was 36% non-violations. The portion of non-violations indicates the inexactness of the underlying search. Search is harder on Penn-S due to the larger label set. Thus, as expected, max-violation update strategies improve most where the search is the hardest and least exact.

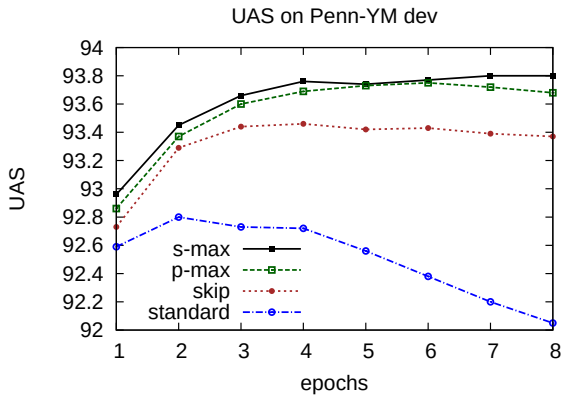


Figure 2: Constrast of different update strategies on the validation data set of Penn-YM. The x-axis is the number of training epochs. The y-axis is the UAS score. s-max stands for single-node max-violation. p-max stands for parallel max-violation.

Figure 2 shows accuracy per training epoch on the validation data. It can be seen that bad update strategies are not simply slow learners. More iterations of training cannot close the gap between strategies. Forcing invalid updates on non-violations (standard update) or simply ignoring them (skip update) produces less accurate models overall.

	ZN 2011 (reimpl.)		skip		s-max		p-max		Best Published†	
Language	*UAS*	*LAS*	*UAS*	*LAS*	*UAS*	*LAS*	*UAS*	*LAS*	*UAS*	*LAS*
SPANISH	86.76	83.81	87.34	84.15	**87.96**	**84.95**	87.68	84.75	*87.48*	*84.05*
CATALAN	94.00	88.65	94.54	89.14	94.58	89.05	**94.98**	**89.56**	*94.07*	*89.09*
JAPANESE	93.10	91.57	**93.40**	91.65	93.26	**91.67**	93.20	91.49	*93.72*	*91.7-*
BULGARIAN	93.08	89.23	93.52	89.25	**94.02**	**89.87**	93.80	89.65	*93.50*	*88.23*
ITALIAN	87.31	82.88	87.75	83.41	87.57	83.22	**87.79**	**83.59**	*87.47*	*83.50*
SWEDISH	90.98	**85.66**	90.64	83.89	**91.62**	85.08	91.62	85.00	*91.44*	*85.42*
ARABIC	78.26	67.09	80.42	69.46	80.48	69.68	**80.60**	**70.12**	*81.12*	*66.9-*
TURKISH	76.62	66.00	76.18	65.90	**76.94**	**66.80**	76.86	66.56	*77.55*	*65.7-*
DANISH	90.84	86.65	91.40	86.59	91.88	86.95	**92.00**	**87.07**	*91.86*	*84.8-*
PORTUGUESE	91.18	87.66	91.69	88.04	92.07	88.30	**92.19**	**88.40**	*93.03*	*87.70*
GREEK	85.63	78.41	86.37	78.29	86.14	78.20	**86.46**	**78.55**	*86.05*	*77.87*
SLOVENE	84.63	76.06	85.01	75.92	**86.01**	**77.14**	85.77	76.62	*86.95*	*73.4-*
CZECH	87.78	82.38	86.92	80.36	88.36	82.16	**88.48**	**82.38**	*90.32*	*80.2-*
BASQUE	**79.65**	71.03	79.57	71.43	79.59	71.52	79.61	**71.65**	*80.23*	*73.18*
HUNGARIAN	84.71	80.16	85.67	80.84	85.85	81.02	**86.49**	**81.67**	*86.81*	*81.86*
GERMAN	91.57	**89.48**	91.23	88.34	**92.03**	89.44	91.79	89.28	*92.41*	*88.42*
DUTCH	82.49	79.71	83.01	79.79	**83.57**	**80.29**	83.35	80.09	*86.19*	*79.2-*
AVG	86.98	81.55	87.33	81.56	87.76	82.08	87.80	82.14		

Table 2: Parsing Results for languages from CoNLL 2006/2007 shared tasks. When a language is in both years, we use the 2006 data set. The best results with † are the maximum in the following papers: Buchholz and Marsi (2006), Nivre et al. (2007), Zhang and McDonald (2012), Bohnet and Kuhn (2012), and Martins et al. (2013), For consistency, we scored the CoNLL 2007 best systems with the CoNLL 2006 evaluation script. ZN 2011 (reimpl.) is our reimplementation of Zhang and Nivre (2011), with a beam of 64. Results in bold are the best among ZN 2011 reimplementation and different update strategies from this paper.

3.3 CoNLL Results

We also report parsing results for 17 languages from the CoNLL 2006/2007 shared-task (Buchholz and Marsi, 2006; Nivre et al., 2007). The parser in our experiments can only produce projective dependency trees as it uses an Eisner algorithm backbone to generate the hypergraph (Eisner, 1996). So, at training time, we convert non-projective trees – of which there are many in the CoNLL data – to projective ones through flattening, i.e., attaching words to the lowest ancestor that results in projective trees. At testing time, our parser can only predict projective trees, though we evaluate on the true non-projective trees.

Table 2 shows the full results. We sort the languages according to the percentage of non-projective trees in increasing order. The Spanish treebank is 98% projective, while the Dutch treebank is only 64% projective. With respect to the Zhang and Nivre (2011) baseline, we improved UAS in 16 languages and LAS in 15 languages. The improvements are stronger for the projective languages in the top rows. We achieved the best published UAS results for 7 languages: Spanish, Catalan, Bulgarain, Italian, Swedish, Danish, and Greek. As these languages are typically from the more projective data sets, we speculate that extending the parser used in this study to handle non-projectivity will lead to state-of-the-art models for the majority of languages.

4 Conclusions

We proposed perceptron update strategies for inexact hypergraph search and experimented with a cube-pruning dependency parser. Both single-node max-violation and parallel max-violation update strategies signficantly improved parsing results over the strategy that ignores any invalid udpates caused by inexactness of search. The update strategies are applicable to any bottom-up parsing problems such as constituent parsing (Huang, 2008) and syntax-based machine translation with online learning (Chiang et al., 2008).

Acknowledgments: We thank André F. T. Martins for the dependency converted Penn Treebank with automatic POS tags from his experiments; the reviewers for their useful suggestions; the NLP team at Google for numerous discussions and comments; Liang Huang and Kai Zhao are supported in part by DARPA FA8750-13-2-0041 (DEFT), PSC-CUNY, and a Google Faculty Research Award.

References

B. Bohnet and J. Kuhn. 2012. The best of bothworlds - a graph-based completion model for transition-based parsers. In *Proc. of EACL*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*.

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

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of ACL*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*.

H. Daumé and D. Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. of ICML*.

M. De Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proc. of COLING*.

L. Huang, S. Fayong, and G. Yang. 2012. Structured perceptron with inexact search. In *Proc. of NAACL*.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

X. Ma and H. Zhao. 2012. Fourth-order dependency parsing. In *Proc. of COLING*.

A. F. T. Martins, N. Smith, E. P. Xing, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

A. F. T. Martins, M. B. Almeida, and N. A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of EMNLP-CoNLL*.

X. Qian and Y. Liu. 2013. Branch and bound algorithm for dependency parsing with non-local features. *TACL*, Vol 1.

A. Rush and S. Petrov. 2012. Efficient multi-pass dependency pruning with vine parsing. In *Proc. of NAACL*.

Y. Zhang and S. Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proc. of EMNLP*.

H. Zhang and R. McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proc. of EMNLP*.

Y. Zhang and J. Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proc. of ACL-HLT*, volume 2.

Predicting the Presence of Discourse Connectives

Gary Patterson and **Andrew Kehler**
Department of Linguistics
UC San Diego
9500 Gilman Drive #0108
La Jolla, CA 92093
{gpatterson,akehler}@ucsd.edu

Abstract

We present a classification model that predicts the presence or omission of a lexical connective between two clauses, based upon linguistic features of the clauses and the type of discourse relation holding between them. The model is trained on a set of high frequency relations extracted from the Penn Discourse Treebank and achieves an accuracy of 86.6%. Analysis of the results reveals that the most informative features relate to the discourse dependencies between sequences of coherence relations in the text. We also present results of an experiment that provides insight into the nature and difficulty of the task.

1 Introduction

A central goal of natural language generation and summarization systems is to produce interpretable, coherent text that rivals material a human would produce. Doing so requires that systems not only have the ability to generate clauses that are grammatical and easy for people to process, but also the ability to employ the appropriate discourse structuring devices needed to yield fluid transitions between these clauses. This is a tricky issue in that it requires that a balance be achieved between the opposing goals of communicative expressiveness and economy. On the one hand, insufficient cueing of inter-clausal relationships can lead to a discourse that is at best difficult to process, and at worst misunderstood. On the other hand, too much explicit marking can result in a clunky and even redundant sounding discourse.

Here we consider the question of when to explicitly mark the COHERENCE RELATIONS in discourse, that is, the inter-clausal relationships that the language producer intends the interpreter to infer between the meanings of clauses (Hobbs, 1979; Mann and Thompson, 1988; Kehler, 2002; Asher and Lascarides, 2003). Consider, for example, the EXPLANATION coherence relation that holds in (1), in which the second clause provides a cause or reason for the eventuality described in the first:

(1) a. Max will visit Australia this summer **because** his father is turning 65.
    b. Max will visit Australia this summer. His father is turning 65.

As example (1) shows, coherence relations can be marked explicitly—by using lexical connectives such as coordinating or subordinating conjunctions (e.g., *because* in 1a) or certain types of prepositional or adverbial phrases—or left implicit as in (1b). Either way, establishing the relation itself requires the reader to go through a complex inferential process necessitating that a variety of assumptions be made, typically supported by context and/or world knowledge, that are not explicitly asserted by the actual linguistic material. In (1), for instance, such inferences would include that Max intends to see his father when he travels to Australia, that his father resides in that country, and that the birthday will taking place during the time of the visit. Importantly, the role of the connective in (1a) is therefore not to *establish* that an EXPLANATION relation holds. Instead, connectives serve the function of directing the addressee's inference processes toward a smaller set of coherence relations than might otherwise be available, among other possible roles.

The fact that both (1a) and (1b) are felicitous may lead us to believe that the choice to insert a connective between clauses is simply optional. This is not

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 914–923,
Seattle, Washington, USA, 18-21 October 2013.

always the case, however. Sometimes the use of a connective is required, since omitting it would likely result in incorrect inferences being drawn by the addressee. For example, the use of *when* in (2a) implies a backward temporal ordering of events, which is reversed if the connective is left out, as in (2b).

(2) a. Maggie fell over in shock **when** Saul offered to help her.
    b. Maggie fell over in shock. Saul offered to help her.

On the other hand, a connective can seem unnecessary if the relation between the two clauses is sufficiently implied by other cues in the text. For instance, since the act of throwing a vase against a concrete wall would normally be expected to cause the vase to break, the adverbial phrase *as a result* in (3a), while felicitous, seems overly verbose and perhaps even redundant.

(3) a. Susan threw the fragile vase against the concrete wall. **As a result**, it broke.
    b. Susan threw the fragile vase against the concrete wall. It broke.

The foregoing examples suggest that the appropriateness of including an explicit connective is inherently gradient, and is in fact correlated with ease of inference: the more difficult recovering the correct relation would be without a connective, the more necessary it is to include one. This characterization in turn suggests that predicting whether a connective should be included might be a difficult problem for an NLP system to address, since current-day systems lack the requisite world knowledge and capacity for inference that would be necessary to evaluate the ease with which coherence relations can be established on arbitrary examples. However, it is also possible that the decision to include a connective depends in part on stylistic and other types of factors as well, such that there might be predictive information in the kinds of shallow linguistic and textual features that systems *do* have access to. This is the question taken up in this work: Given two adjacent clauses in a text, the type of coherence relation holding between them and a candidate connective that could be used to signal the relation, we ask how well a system can predict whether or not that connective was used by the author of the text. This capability would be useful to generation systems as a post-cursor to discourse-level message planning and sentence realization processes, as well as summarization systems that take existing sentences and have to reconsider connective placement upon reassembling them.

To our knowledge, there is no work in the literature that addresses this issue directly. There is a growing body of research (Sporleder and Lascarides, 2008; Pitler et al., 2009; Lin et al., 2009; Zhou et al., 2010) that focuses on building supervised models for classifying implicit relations using a variety of contextual features, such as the polarity of clauses, the semantic class and tense/aspect of verbs, and information from syntactic parses. With respect to explicit relations, Elhadad and McKeown (1990) sketch a procedure to select an appropriate connective to link two propositions as part of a larger text generation system, using linguistic features derived from the sentences. The procedure selects the best connective from a given set of candidates, but does not allow for the option of leaving the relation implicit. More recently, Asr and Demberg (2012a) look at both explicit and implicit relations, and make the observation that certain relation types are more likely to be realized explicitly than others. Relatedly, Asr and Demberg (2012b) discuss which connectives are the strongest predictors of which relation types. However, there is no work of which we are aware that specifically predicts whether connectives should be used or omitted.

2 Classification Model

Our model is a binary classifier trained on data extracted from the Penn Discourse Treebank (PDTB; Prasad et al. (2008)), a large-scale corpus of annotated discourse coherence relations covering the one-million-word Wall Street Journal corpus of the Penn Treebank (Marcus et al., 1993).

2.1 Data

For every relation in the PDTB, the following components are annotated: (i) the connective used to signal the relation; (ii) the textual spans of the two clausal arguments that constitute the relation; (iii) the semantic sense of the relation, according to a hierarchical tagset of senses; and (iv) the attribution of the assertions and beliefs expressed in the text to the

relevant individuals. Crucially for our purposes, for the implicit relations the corpus indicates the most suitable connective if the relation were instead signaled explicitly. For example, the annotators decided that the best connective to signal the REASON relation in (4) would be *because*, rather than other plausible candidates, such as *as* or *since*.

(4) It's a shame their meeting never took place. [IMPLICIT=*because*] Mr. Katzenstein certainly would have learned something. (WSJ0037)

In total, there are 18,459 explicit and 16,053 implicit relations annotated in the PDTB. We excluded a subset of these cases in training our model based on two criteria. First, whereas explicit relations in the PDTB can hold between spans of text that either are or are not adjacent, we excluded the non-adjacent cases. This was done to ensure consistency in discourse structure between the relations considered in the model, since only the implicit relations between adjacent clauses were annotated. Second, we excluded relations that have lower frequency semantic senses or use low frequency connectives. As a result, the model considers only the eight most common semantic senses of relations, which in total account for just less than 90% of the relations in the corpus.[1] Further, for each relation, we only consider the connectives that account for more than 5% of the instances of that relation. After applying these filters, the resulting corpus comprised 10,039 explicit and 11,690 implicit relations.

Table 1 shows the eight relations that were modeled. The majority of these relations exist at the middle layer of the three-level hierarchy of semantic senses annotated in the PDTB.[2] Relations at the highest level – representing the four major semantic categories COMPARISON, CONTINGENCY, EXPANSION and TEMPORAL – were deemed too broad to be of practical use in a generation system, whereas the lowest-level senses were considered either unnecessarily fine-grained or have too few tokens in the corpus to allow for meaningful statistical modeling. Two exceptions were made for REASON and RESULT relations, which do appear at the lowest level in the PDTB hierarchy (beneath the CAUSE category). These were included because they are both attested frequently in the corpus and are undeniably contrastive: with REASON, the second clause provides an explanation for the proposition expressed in the first clause, whereas with RESULT, the second clause describes a consequence of the first. It is reasonable to want these relations to be modeled separately.

Sections 2-22 of the corpus were used as the training set, and sections 23 and 24 were used as the test set. Sections 0 and 1 of the corpus were set aside as a development set for feature design and parameter optimization. The training set comprised 18,218 tokens, distributed as shown in Table 1.

Relation Type	**Explicit**		**Implicit**	
Asynchronous	1,120	(70%)	469	(30%)
Conjunction	2,940	(61%)	1,906	(39%)
Contrast	2,044	(66%)	1,054	(34%)
Instantiation	203	(16%)	1,093	(84%)
Reason	771	(28%)	1,938	(72%)
Restatement	76	(4%)	2,081	(96%)
Result	354	(21%)	1,295	(79%)
Synchronous	748	(86%)	126	(14%)
Total	**8,256**	(45%)	**9,962**	(55%)

Table 1: Distribution of Training Set

As Table 1 shows, the preference for an overt connective varies significantly according to the type of relation. The ASYNCHRONOUS, CONJUNCTION, CONTRAST and SYNCHRONOUS relations are realized explicitly the majority of the time, whereas INSTANTIATION, REASON, RESTATEMENT and RESULT relations are more often left implicit. We can also see that some relation types (such as RESTATEMENT, INSTANTIATION and SYNCHRONOUS) exhibit a strong preference to be realized in a particular form, whereas other types show more variability in whether they are realized explicitly or implicitly.

The distribution of tokens in Table 1 can be used to determine a baseline accuracy against which the performance of our model is evaluated. A naive model that uses the semantic type of the coherence relation as the sole predictive feature makes a binary classification based simply on the majority category for that relation type. A baseline model using this methodology results in classification accuracy of 77.0% over the held-out test set.

[1] The next most common relation type, CONDITION, was excluded because it is always marked explicitly in the corpus.

[2] For more details of the PDTB sense hierarchy, see Prasad et al. (2008).

2.2 Model

We built a composite model containing binary logistic regression classifiers for each coherence relation, trained on a set of linguistic features extracted from each token in the training set. Logistic regression was chosen because it produces a model with high performance and results that are easily interpretable. The features included in the model fall into the following three broad classes: relation-level, argument-level, and discourse-level.

Relation-level features

In addition to the semantic type of the relation, we include as a feature the connective used to signal the relation in the text (or, for the implicit relations, the connective indicated by the annotators as most appropriate). This feature (*Connect*) is included based upon the observation that connectives vary as to their rates of being realized explicitly—even for connectives that signal relations with the same semantic sense. Consequently, given a relation of a particular semantic type, an indication of the best fitting connective may be a consistent predictor of whether or not this relation is realized explicitly.

We also include a feature reflecting the attribution of the relation. As mentioned above, the PDTB is annotated to describe the attribution of the propositions expressed within a relation to individuals or entities in the text. For example, in the relation shown in (5), the first clause contains a direct quotation, clearly attributing the proposition expressed to the individual *Rep. Stark*. However, the second clause contains no such indication of attribution to an entity in the text, and so the proposition is instead assumed to be asserted by the writer of the article.

(5) "No magic bullet will be discovered next year, an election year," says Rep. Stark. **But** 1991 could be a window for action. (WSJ0314)

Inspection of the corpus data suggests that when one argument of a relation contains a proposition that is attributable to an individual in the text (either by direct or indirect quotation) but the other is assumed by default to be attributed to the author, this relation is more likely to be realized explicitly. This may well have an explanation based on sentence processing: the intervening attribution phrase *'says Rep. Stark'* may serve as a distraction, with the result that the intended coherence relation is harder to infer without a connective. Consequently, we include a factor (*AttMismatch*) indicating if the two arguments are not attributed to the perspective of the same individual.

Finally, in any particular genre there may be formulaic prose whose systematic features can be exploited by a system tasked with generating text within that same genre. In this case, the genre represented by the corpus data comprises copy-edited articles from the Wall Street Journal, many of which refer to company earnings reports or other financial events, and are written in a highly prescribed style. Accordingly, we may suspect that there is a greater prevalence of implicit relations in these cases, since the reader is assumed to be habituated to the way in which the information in this type of article is presented. Consequently, for the domain at hand we include a binary feature (*Financial*) indicating whether the relation pertains to financial information. This feature takes the value 1 if the textual spans of both arguments in the relation contain percentage amounts or dollar figures.

Argument-level features

For each relation, the model includes features capturing the size or complexity of each of its two arguments. The arguments were identified by the annotators according to a principle of minimality, whereby the annotations indicate the shortest text spans necessary for the appropriate coherence relation to be interpreted. However, the annotators also indicated other text that is in some way relevant to the interpretation of the arguments. This supplementary material can include unrestricted relative clauses, appositives, or other parenthetical information. Our observation of the data indicates that relations which have supplementary material annotated alongside one or both of their arguments are more often than not realized explicitly with connectives. As a result, we include binary features (*Supp1*, *Supp2*) indicating whether the first and second arguments of the relation include such supplementary information. We also include features (*Length1*, *Length2*) reflecting a simple measure of the length of each argument, calculated as the log transformed count of the number of words in the arguments' minimal text spans.

One measure of the complexity of an argument

is the number of clauses it contains. It might be thought that the greater the syntactic complexity of an argument, the more likely it is that the relation containing it is marked explicitly, so as to give the reader more help in drawing the correct intended inference between the arguments. As a proxy for the number of clauses in each argument, we include features (*NPSbj1*, *NPSbj2*) equal to the total number of main, subordinate, or complement clause subjects included within the textual spans of the respective arguments, determined using the syntactic parses available in the corpus data.

We also consider whether the underlying richness of the informational content expressed by the argument may influence the presence or omission of a connective. Considering the way in which readers process text in real time, it would intuitively be more difficult to infer the intended relation between two clauses without the aid of a connective if the arguments themselves had greater processing demands owing to increased lexical retrieval, reference and anaphor resolution requirements, and so forth. Given this intuition, we may expect that arguments with higher density of information are correlated with the increased use of connectives as a means of facilitating the inference of the relation type and thereby easing the overall processing burden. Consequently, our model includes features (*ContDensity1*, *ContDensity2*) calculated as the ratio of the count of words in each argument that are content words (i.e. ignoring articles, prepositions and pronouns), divided by the total number of words, as well as features (*PronDensity1*, *PronDensity2*) calculated as the ratio of pronouns in each argument to the number of noun phrases.

Finally, the accessibility of the subject of the second argument in a relation may play a role in determining whether the relation is explicitly marked. Specifically, informal observation of the data suggests that there is a tendency for the second argument of an implicit relation to begin with a longer, contentful noun phrase, rather than a pronoun. Consequently, our model includes a binary feature (*FirstA2Pron*) indicating whether the first word in the second argument is a pronoun.

Discourse-level features

The final class of features takes account of the way in which a relation fits into the broader discourse structure in the text. In their work on implicit relation classification, Pitler et al. (2008) identified various dependencies between bigram sequences of explicitly- and implicitly-realized relations of different semantic types. These results suggest that the semantic type and the presence of a connective in one relation may be predictive of whether or not the following relation in the text is marked with a connective. Consequently, we include features indicating the semantic type of the relation occurring immediately prior in the text (*PrevSemType*), and whether this relation was marked implicitly or explicitly (*PrevForm*).

The other discourse-level features take account of the dependencies between the relation in question and its neighboring relations in the text. As part of a supervised learning model developed to classify the semantic class of implicit relations in the PDTB, Lin et al. (2009) found features based on the two main types of discourse dependency pattern in the corpus ('shared' and 'fully embedded' arguments) to be highly predictive. We speculatively include similar features in our model to see if they are helpful in predicting the presence of connectives.

The first type of dependency between adjacent relations is one where the second argument of one relation is also the first argument of the following relation, as in Figure 1. Accordingly, we include two binary features indicating whether an argument is shared with the preceding relation (*Arg1isPrevArg2*) or the following relation (*Arg2isNextArg1*) in the corpus.

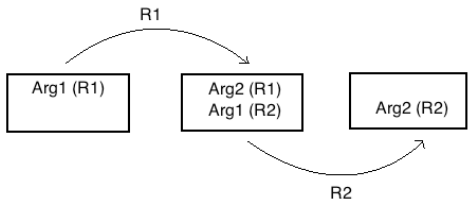


Figure 1: Shared argument

The other main type of discourse dependency, a 'fully embedded' dependency, is one where an entire relation (including both of its arguments) is completely embedded within one argument of an adjacent relation in the text, as in Figure 2. To capture

this type of dependency structure, we include two binary features (*EmbedNext*, *EmbedPrev*) indicating whether the current relation is embedded within either one of its adjacent relations. We also include two binary features (*Arg1Embed*, *Arg2Embed*) to indicate whether either argument of the current relation completely contains an embedded relation.

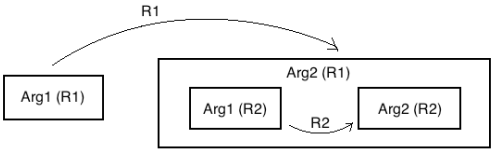


Figure 2: Fully embedded argument

The two relations in (6) exemplify a typical instantiation of this embedded dependency structure.

(6) It is an overwhelming job. [IMPLICIT= *because*] There are so many possible proportions **when** you consider how many things are made out of eggs and butter and milk. (WSJ0261)

In this example, there is an implicit REASON relation holding between the two complete sentences, and an explicit SYNCHRONOUS relation signaled by the connective *when* holding between the two clauses of the second sentence. Since the REASON relation fully embeds another relation within its second argument, the feature *Arg2Embed* for this relation takes the value 1. For the SYNCHRONOUS relation, the feature *EmbedPrev* takes the value 1 since the entire relation is fully contained within the second argument of the preceding relation in the text.

3 Results and Evaluation

3.1 Classification accuracy

The model was evaluated by assessing the accuracy of its predictions against the unseen test set. The model achieved an overall accuracy of 86.6%, an improvement of 9.6% above baseline.[3] Table 2 shows the model accuracy for each relation type, together with the baseline performance based on the majority category for that type.

[3]During the preparation of the final version of this paper, a model was trained with an SVM using the same set of features, which resulted in a modest improvement in performance (87.3%). The ensuing discussion of results, however, will continue to pertain to the regression model.

Relation Type	Accuracy	Baseline
Asynchronous	91.7%	79.7%
Conjunction	84.5%	78.2%
Contrast	81.1%	65.0%
Instantiation	83.3%	82.5%
Reason	88.2%	68.3%
Restatement	95.2%	95.2%
Result	84.4%	76.9%
Synchronous	96.5%	92.9%
Total	**86.6%**	**77.0%**

Table 2: Classification Accuracy by Relation Type

The model achieved an improvement in accuracy across all relation types but one: RESTATEMENT relations, for which the baseline accuracy was already close to 100%. The greatest improvement in accuracy was seen for REASON relations, for which the model accuracy was 19.9% above baseline. We now discuss which of the factors in each of the feature classes turned out to be the most predictive.

3.2 Significant predictors

We trained the model on subsets of the features to investigate the predictive power of the different feature classes. The accuracy assessed against the test set is shown in Table 3.

Feature Class	# Features	Accuracy
Relation Level	4	80.4%
Argument Level	11	77.2%
Discourse Level	8	80.9%
Rel + Arg Levels	15	82.8%
Rel + Disc Levels	12	85.1%
Arg + Disc Levels	19	82.4%
All Features	23	86.6%

Table 3: Classification Accuracy by Feature Class

The classes of Relation-level and Discourse-level features each separately yielded significantly better performance over baseline (one-sided tests of proportion, z=2.50 and z=2.92, respectively; $p<0.01$ for both), whereas the Argument-level features alone performed only marginally better than baseline. However, all three classes of features are needed to attain the highest model performance.

Across all relation types, we found that the features relating to the discourse dependencies between a relation and its neighbors were the strongest and

most consistent predictors of whether that relation is explicit or implicit. A relation that is fully embedded within a single argument of an adjacent relation in the text (indicated by the features *EmbedPrev* and *EmbedNext*) has a much higher likelihood of being signaled explicitly. Conversely, a relation that fully contains another relation within one of its arguments (indicated by *Arg1Embed* and *Arg2Embed*) has a significantly higher likelihood of being implicit. The result is consistent with the embedded discourse dependency shown in (6), in which the implicit REASON relation fully contains an explicit SYNCHRONOUS relation within its second argument.

The model also found that the features which indicate whether a relation has shared arguments with either the preceding or following relations in the text (*Arg1isPrevArg2*, *Arg2isNextArg1*) are both predictors of an implicit outcome. In other words, if a clause in the text serves as the argument for two adjacent relations, then both of these relations are more likely to be realized implicitly.

The next most predictive feature was the connective used to signal the relation (*Connect*). This feature was a significant predictor for every relation type. Eliminating this feature from the final model reduces the overall accuracy by 2.5%. The other features in the model were less significantly predictive, and generally worked in the expected direction. Longer arguments (*Length1, Length2*) and the indication of a financial genre (*Financial*) were generally associated with predicted implicit outcomes, whereas the presence of supplementary material (*Supp1, Supp2*), a mismatch of attribution (*AttMismatch*), more 'content rich' arguments (*ContDensity1, ContDensity2*), and a pronoun appearing as the first word of the second argument (*FirstA2Pron*) all tended to increase the odds in favor of a predicted explicit outcome.

The features indexing syntactic complexity (*NPSbj1* and *NPSbj2*) were found to be marginally predictive of an explicit outcome for most relation types, but the overall effect in the model was relatively small—resulting in only a 0.2% improvement—meaning that the level of performance reported on this task depends very little on the model having access to full syntactic parses. Somewhat unexpectedly, the factors indicating the semantic type of the previous relation in the text (*PrevSemType*) and whether or not this relation was explicitly signaled by a connective (*PrevForm*) were found not to be significant predictors. Our analysis of the training data confirmed the findings of Pitler et al. (2008) in that certain bigrams of coherence relation types are significantly more prevalent than others. However, the differences in the frequencies were evidently not sufficiently correlated with the explicit/implicit distinction as to make the type or form of the previous relation a significant feature in the model.

3.3 Error analysis

We analyzed a sample of cases incorrectly predicted by the model to see if there were any consistent traits. We focus our attention here on the CONTRAST relations, which is the type with the lowest model accuracy. The majority of these errors were cases where the model predicted that the relation would be explicit—the most likely outcome for a CONTRAST relation—whereas in the corpus the intended relation was signaled by linguistic cues other than an overt connective. For instance, the strong syntactic parallelism of the two arguments in (7), and the opposite polarity of the lexical items *delight* and *detriment*, combine to induce a contrastive relationship without the need for a connective.

(7) To the delight of some doctors, the bill dropped a plan passed by the Finance Committee. [IMPLICIT=*but*] To the detriment of many low-income people, efforts to boost Medicaid funding were also stricken. (WSJ2372)

Other ways that the contrast relation is signaled implicitly include contrasting temporal modifiers (*It wasn't so long ago X. Now, Y*), repetition of the predicate in the argument (*. . . it could only happen once. . . . it's happening again*), or even by the use of punctuation such as a semicolon. Previous work (Sporleder and Lascarides, 2008; Lin et al., 2009) has sought to make use of such cues to identify and classify implicit relations in the text. The results of this brief error analysis suggest that such indirect cues could also be useful factors in determining whether to choose to use a connective for a given relation type when generating text.

4 Judgment Study

The system described in the last section outperformed a baseline majority-category classifier on the task of deciding whether a relation should be made explicit or left implicit. This result might be considered surprising, for two reasons that we have previously discussed. First, the system was able to make this improvement using relatively shallow features extracted from the text, without access to the richer types of contextual information and world knowledge required for establishing coherence relations during actual discourse comprehension. Second, the data suggest that the appropriateness of including a connective is not as cut-and-dried as a binary classification task may suggest, but is instead gradient, with many cases for which the inclusion of a connective appears to be optional. Obviously, the PDTB does not avail us of the opportunity to evaluate this gradience directly (or even use a 3-way required/optional/redundant distinction), since the producer of the actual text samples in the corpus had to ultimately decide whether or not to use a connective. The apparent optionality of many examples thus puts limits on how well we can expect a system to perform, since there is no way to reliably predict cases in which the decision is made arbitrarily.

This observation leads us to ask how well humans perform on this same task. Do they make highly accurate predictions, or does optionality limit their performance? In order to shed light on this question, we carried out an experiment to see how consistently humans choose to use lexical connectives to signal intended coherence relations between clauses.

4.1 Methodology

We selected a balanced sample of 100 clause-pair tokens from the test set, reflecting the distribution of the different major relation types (six relations were represented in the sample). This sample comprised 44 explicit and 56 implicit tokens, consistent with the distribution in the overall corpus. The experimental stimulus for each item consisted of two versions of the same clause pair, one including a connective between the clauses, and the other without. For relations that were realized explicitly in the corpus, as in (8a), the alternative implicit stimulus omitted the connective and showed the second argument as a separate sentence, as in (8b).

(8) a. Mr. Nesbit also said the FDA has asked Bolar Pharmaceutical Co. to recall at the retail level its urinary tract antibiotic, **but** so far the company hasn't complied with that request.

b. Mr. Nesbit also said the FDA has asked Bolar Pharmaceutical Co. to recall at the retail level its urinary tract antibiotic. So far the company hasn't complied with that request.

For the implicit relations, the alternative explicit stimulus for the experiment used the connective annotated in the PDTB as the one being most appropriate. For each item, a short passage was created including the preceding and following sentences in the text to serve as context. The relative ordering of the presentation of the explicit and implicit forms was randomized, without regard to the actual corpus outcome for that stimulus.

Using Amazon's Mechanical Turk, judges were presented with the two passages for each item. They were told to assume that the passages had the same intended meaning, and were asked to judge which of the two sounded more natural. We collected 30 responses for each item.[4]

4.2 Results

We classified each experimental item as either explicit or implicit, based on the majority response of the judges. Using this classification, the judges' responses matched the actual outcomes in 68 of the 100 cases.[5] The distribution of correctly-judged items across relation types is shown in Table 4. The judgments for REASON relations most closely matched the corpus outcomes, with 9 out of the 12 explicit tokens and all 6 implicit tokens in the cor-

[4]The data from a small number of judges were discarded due to an unreasonably fast response time or because their judgments showed a unanimous preference across every experimental item. This left a total of 2,925 judgments over the 100 experimental items, from 113 different judges.

[5]Using the majority response of judges for each item to measure classification accuracy is consistent with the statistical model, whereby probabilities are rounded up or down to arrive at a binary classification. If accuracy is instead calculated in terms of average correctness over the individual responses, performance drops to 60.4%.

pus correctly identified by the judges. The lowest scoring relation type was CONTRAST, for which 9 of the 10 explicit tokens were judged correctly but only 4 out of the 11 implicit tokens were correctly identified.

Relation Type	Items	Correct	Accuracy
Conjunction	22	15	68.2%
Contrast	21	13	61.9%
Instantiation	9	6	66.7%
Reason	18	15	83.3%
Restatement	16	9	62.5%
Result	14	10	64.2%
Total	100	68	68.0%

Table 4: Results of Mechanical Turk study

There were hence 32 experimental items for which the majority response by the judges did not match the actual corpus outcome. In two-thirds (21) of these cases, the judges indicated a preference for a connective when the relation in the corpus was implicit. These mismatches occurred across the range of relation types. This suggests that the judges tended to err on the side on inserting a connective, even when it may not have been strictly necessary. While the reason for this is not clear, one possibility is that the texts reflected the genre and the highly-prescribed editing guidelines for the newspaper articles that comprise the corpus, under which unnecessary or redundant words are excised. Without such pressures to edit the copy down to a minimal form, the judges may have preferred to see the relations signaled explicitly in cases in which either decision would result in a felicitous passage.

In the remaining 11 cases, for which the relations in the corpus were explicitly signaled with a connective, the judges on average indicated a preference to leave the relation implicit. Interestingly, all of these cases were either CONJUNCTION or CONTRAST relations, semantic types which are usually signaled explicitly with a connective. We inspected these cases to ascertain why judges may have preferred an outcome opposite to that actually seen in the text. We found that all 7 of the CONTRAST mismatches were instances where the second argument of the relation in the corpus was a sentence beginning with the coordinating conjunction *but*, as in (9).

(9) At those levels stocks are set up to be hammered by index arbitragers. **But** nobody knows at what level the futures and stocks will open today. (WSJ2300)

Similarly, three of the mismatched CONJUNCTION relations had a sentential second argument beginning with the conjunction *and*. The responses of the judges to these cases may simply reflect a dispreference for sentence-initial conjunctions, a practice which is frowned upon in prescriptive grammar books, but apparently allowed by the Wall Street Journal style sheet.

For this sample of 100 relations, the model achieves a classification accuracy of 84%. This may seem at first blush to be an odd result, since it appears that the model is surpassing human performance. As we have suggested, however, this could be the result of our experimental judges having different preferences than the writers and editors at the Wall Street Journal for cases in which connective placement is truly optional. We therefore sought to evaluate the effect of optionality on these results.

If inaccurate predictions are associated with optionality of connective use, we might expect that both human judges and the classification model would be less certain about their categorizations of these examples than for the cases that were correctly classified. This was indeed the case. First, there was a significant difference in the variability of judges' responses between items that were incorrectly classified and those that were correct (66% vs. 73%, respectively; two-sample t test: t=2.60, df=73, $p<0.02$). Thus, as a group the judges were less sure of themselves in those cases in which they incorrectly decided to use or omit the connective, suggesting that either option may have been acceptable. Second, we analyzed the levels of confidence our model had for its judgments on correctly and incorrectly categorized cases, measured in terms of the probability of the predicted outcome assigned by the model. The analysis revealed that the average model confidence for the relations that were incorrectly classified was significantly lower than the average model confidence for the correctly-classified items (71% vs. 88%, respectively; t=5.65, df=25, $p<0.001$). Taken together, these results are consistent with the idea that, at least for a significant portion of the data, the incorrect judgments made by

both the judges and the model may have occurred on passages for which either including or omitting the connective would have been acceptable.

5 Conclusion

We have presented a model that predicts whether the coherence relation holding between two clauses is marked explicitly with a lexical connective or left implicit. Whereas there is reason to think that an author's decision to use a connective is in part influenced by properties of the extra-linguistic context that are inaccessible to NLP systems (such as semantics and world knowledge), we find that relatively simple linguistic features derivable from the clauses and from local discourse dependencies can be exploited to reach a level of performance significantly greater than that achieved by a baseline. The variability in the judgments of native speakers when presented with these data suggests that the use of a connective is in many cases simply optional; in such cases the decision may reflect lower-level stylistic choices on the part of the author. This in turn indicates that there may be an inherent upper bound to the performance of computational systems on this task.

Acknowledgments

We thank Roger Levy for useful discussions about this work and three anonymous reviewers for their helpful feedback.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Fatemeh Torabi Asr and Vera Demberg. 2012a. Implicitness of discourse relations. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 2669–2684.

Fatemeh Torabi Asr and Vera Demberg. 2012b. Measuring the strength of linguistic cues for discourse relations. In *Proceedings of the Workshop on Advances in Discourse Analysis and its Computational Aspects (ADACA)*, pages 33–42.

Michael Elhadad and Kathleen R McKeown. 1990. Generating connectives. In *Proceedings of the 13th Conference on Computational Linguistics-Volume 3*, pages 97–101.

Jerry R Hobbs. 1979. Coherence and coreference. *Cognitive science*, 3(1):67–90.

Andrew Kehler. 2002. *Coherence, reference, and the theory of grammar*. CSLI Publications, Stanford.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 343–351.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 683–691.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14(3):369–416.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1507–1514.

Japanese Zero Reference Resolution Considering Exophora and Author/Reader Mentions

Masatsugu Hangyo **Daisuke Kawahara** **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku
Kyoto, 606-8501, Japan
{hangyo,dk,kuro}@nlp.ist.i.kyoto-u.ac.jp

Abstract

In Japanese, zero references often occur and many of them are categorized into zero exophora, in which a referent is not mentioned in the document. However, previous studies have focused on only zero endophora, in which a referent explicitly appears. We present a zero reference resolution model considering zero exophora and author/reader of a document. To deal with zero exophora, our model adds pseudo entities corresponding to zero exophora to candidate referents of zero pronouns. In addition, we automatically detect mentions that refer to the author and reader of a document by using lexico-syntactic patterns. We represent their particular behavior in a discourse as a feature vector of a machine learning model. The experimental results demonstrate the effectiveness of our model for not only zero exophora but also zero endophora.

1 Introduction

Zero reference resolution is the task of detecting and identifying omitted arguments of a predicate. Since the arguments are often omitted in Japanese, zero reference resolution is essential in a wide range of Japanese natural language processing (NLP) applications such as information retrieval and machine translation.

(1) パスタが 好きで 毎日 (ϕ ガ)
pasta-NOM like everyday (ϕ-NOM)
(ϕ ヲ) 食べます。
(ϕ-ACC) eat
(Liking pasta, (ϕ) eats (ϕ) every day)

For example, in example (1) , the accusative argument of the predicate “食べます” (eat) is omitted .[1] The omitted argument is called a zero pronoun. In this example, the zero pronoun refers to “パスタ” (pasta).

Zero reference resolution is divided into two subtasks: zero pronoun detection and referent identification. Zero pronoun detection is the task that detects omitted zero pronouns from a document. In example (1), this task detects that there are the zero pronouns in the accusative and nominative cases of “食べます” (eat) and there is no zero pronoun in the dative case of “食べます”. Referent identification is the task that identifies the referent of a zero pronoun. In example (1), this task identifies that the referent of the zero pronoun in the accusative case of “食べます” is “パスタ” (pasta). These two subtasks are often resolved simultaneously and our proposed model is a unified model.

Many previous studies (Imamura et al., 2009; Sasano et al., 2008; Sasano and Kurohashi, 2011) have treated only **zero endophora**, which is a phenomenon that a referent is mentioned in a document, such as “パスタ” (pasta) in example (1). However, **zero exophora**, which is a phenomenon that a referent does not appear in a document, often occurs in Japanese when a referent is an author or reader of a document or an indefinite pronoun. For example, in example (1), the referent of the zero pronoun of the nominative case of “食べます” (eat) is the author of

[1] In this paper, we use the following abbreviations: NOM (nominative), ABL(ablative), ACC (accusative), DAT (dative), ALL (allative), GEN (genitive), CMI (comitative), CNJ (conjunction), INS(instrumental) and TOP (topic marker).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 924–934,
Seattle, Washington, USA, 18-21 October 2013.

	Zero pronoun	Referent in the document	Example
Zero endophora	Exist	Exist	僕はカフェが好きで毎日 (カフェ ニ)通っている。 (I like cafes and go (to a cafe) everyday.)
Zero exophora	Exist	Not exist	私がメリットを ([*reader*] ニ) 説明させていただきます。 (I would like to explain the advantage (to [*reader*]).)
No zero reference	Not exist	Not exist	あなたはリラックスタイムが (×ニ)過ごせる。 (You can have a relaxing time.) *There is no dative case.

Table 1: Examples of zero endophora, zero exophora and no zero reference.

the document, but the author is not mentioned explicitly.

(2) 最近は パソコンで 動画を
recently PC-INS movie-ACC
([*unspecified:person*] ガ) 見れる。
([*unspecified:person*]-NOM) can see
(Recently, (people) can see movies by a PC.)

Similarly, in example (2), the referent of the zero pronoun of the nominative case of "見れる" (can see) is an unspecified person.[2]

Most previous studies have neglected zero exophora, as though a zero pronoun does not exist in a sentence. However, such a rough approximation has impeded the zero reference resolution research. In Table 1, in "zero exophora," the dative case of the predicate has the zero pronoun, but in "no zero reference," the dative case of the predicate does not have a zero pronoun. Treating them with no distinction causes a decrease in accuracy of machine learning-based zero pronoun detection due to a gap between the valency of a predicate and observed arguments of the predicate. In this work, to deal with zero exophora explicitly, we provide pseudo entities such as [*author*], [*reader*] and [*unspecified:person*] as candidate referents of zero pronouns.

In the referent identification, selectional preferences of a predicate (Sasano et al., 2008; Sasano and Kurohashi, 2011) and contextual information (Iida et al., 2006) have been widely used. The author and reader (A/R) of a document have not been used for contextual clues because the A/R rarely appear in the discourse in corpora based on newspaper articles, which are main targets of the previous studies. However, in other domain documents such as blog articles and shopping sites, the A/R often appear in the discourse. The A/R tend to be omitted and there are many clues for the referent identification about the A/R such as honorific expressions and modality expressions. Therefore, it is important to deal with the A/R of a document explicitly for the referent identification.

The A/R appear as not only the exophora but also the endophora.

(3) 僕 $_{author}$ は 京都に (僕ガ)
I-TOP Kyoto-DAT (I-NOM)
行こうと 思っています。
will go have thought
(I have thought (I) will go to Kyoto.)

皆さん $_{reader}$ は どこに 行きたいか
you all-TOP where-DAT want to go
(皆さんガ) (僕ニ) 教えてください。
(you all-NOM) (I-DAT) let me know

(Please let (me) know where do you want to go.)

In example (3), "僕" (I), which is explicitly mentioned in the document, is the author of the document and "皆さん" (you all) is the reader. In this paper, we call these expressions, which refer to the author and reader, **author mention** and **reader mention**. We treat them explicitly to improve the performance of zero reference resolution. Since the A/R are mentioned as various expressions besides personal pronouns in Japanese, it is difficult to detect the A/R mentions based merely on lexical information. In this work, we automatically detect the A/R mentions by using a learning-to-rank algorithm(Herbrich et al., 1998; Joachims, 2002) that uses lexico-syntactic patterns as features.

Once the A/R mentions can be detected, their information is useful for the referent identification.

[2] In the following examples, omitted arguments are put in parentheses and exophoric referents are put in square brackets.

The A/R mentions have both a property of the discourse element mentioned in a document and a property of the zero exophoric A/R. In the first sentence of example (3), it can be estimated that the referent of the zero pronoun of the nominative case of “行こう” (will go) from a contextual clue that “僕” (I) is the topic of this sentence and a syntactic clues that “僕” (I) depends on “思っています” (have thought) over the predicate “行こう” (will go).[3] Such contextual clues can be available only for the discourse entities that are mentioned explicitly. On the other hand, in the second sentence, since “教えてください” (let me know) is a request form, it can be assumed that the referent of the zero pronoun of the nominative case is “僕” (I), which is the author, and the one of the dative case is “皆様” (you all), which is the reader. The clues such as request forms, honorific expressions and modality expressions are available for the author and reader. In this work, to represent such aspect of the A/R mentions, both the endophora and exophora features are given to them.

In this paper, we propose a zero reference resolution model considering the zero exophora and the author/reader mentions, which resolves the zero reference as a part of a predicate-argument structure analysis.

2 Related Work

Several approaches to Japanese zero reference resolution have been proposed.

Iida et al. (2006) proposed a zero reference resolution model that uses the syntactic relations between a zero pronoun and a candidate referent as a feature. They deal with zero exophora by judging that a zero pronoun does not have anaphoricity. However, the information of zero pronoun existences is given and thus they did not address zero pronoun detection.

Zero reference resolution has been tackled as a part of predicate-argument structure analysis. Imamura et al. (2009) proposed a predicate-argument structure analysis model based on a log-linear model that simultaneously conducts zero endophora resolution. They assumed a particular candidate referent, NULL, and when the analyzer selected this referent, the analyzer outputs “zero exophora or no zero pronoun,” in which they are treated without distinction. Sasano et al. (2008) proposed a probabilistic predicate-argument structure analysis model including zero endophora resolution by using wide-coverage case frames constructed from a web corpus. Sasano and Kurohashi (2011) extended the Sasano et al. (2008)’s model by focusing on zero endophora. Their model is based on a log-linear model that uses case frame information and the location of a candidate referent as features. In their work, zero exophora is not treated and they assumed that a zero pronoun is absent when there is no referent in a document.

For languages other than Japanese, zero pronoun resolution methods have been proposed for Chinese, Portuguese, Spanish and other languages. In Chinese, Kong and Zhou (2010) proposed tree-kernel based models for three subtasks: zero pronoun detection, anaphoricity decision and referent selection. In Portuguese and Spanish, only a subject word is omitted and zero pronoun resolution has been tackled as a part of coreference resolution. Poesio et al. (2010) and Rello et al. (2012) detected omitted subjects and made a decision whether the omitted subject has anaphoricity or not as preprocessing of coreference resolution systems.

3 Baseline Model

In this section, we describe a baseline zero reference resolution system. In our model, the zero reference resolution is conducted as a part of predicate-argument structure (PAS) analysis. The PAS consists of a case frame and an alignment between case slots and referents. The case frames are constructed for each meaning of a predicate. Each case frame describes surface cases that each predicate has (case slot) and words that can fill each case slot (example). In this study, the case frames are constructed from 6.9 billion Web sentences by using Kawahara and Kurohashi (2006a)’s method.

The baseline model does not treat zero exophora as the previous studies. The baseline model analyzes a document in the following procedure in the same way as the previous study (Sasano and Kurohashi, 2011).[4]

[3] Since “僕” (I) depends on “思っています” (have thought), the relation between “僕” (I) and “行こう” (will go) is the zero reference.

[4] For learning, the previous study used a log-linear model, but we use a learning-to-rank model. In our preliminary exper-

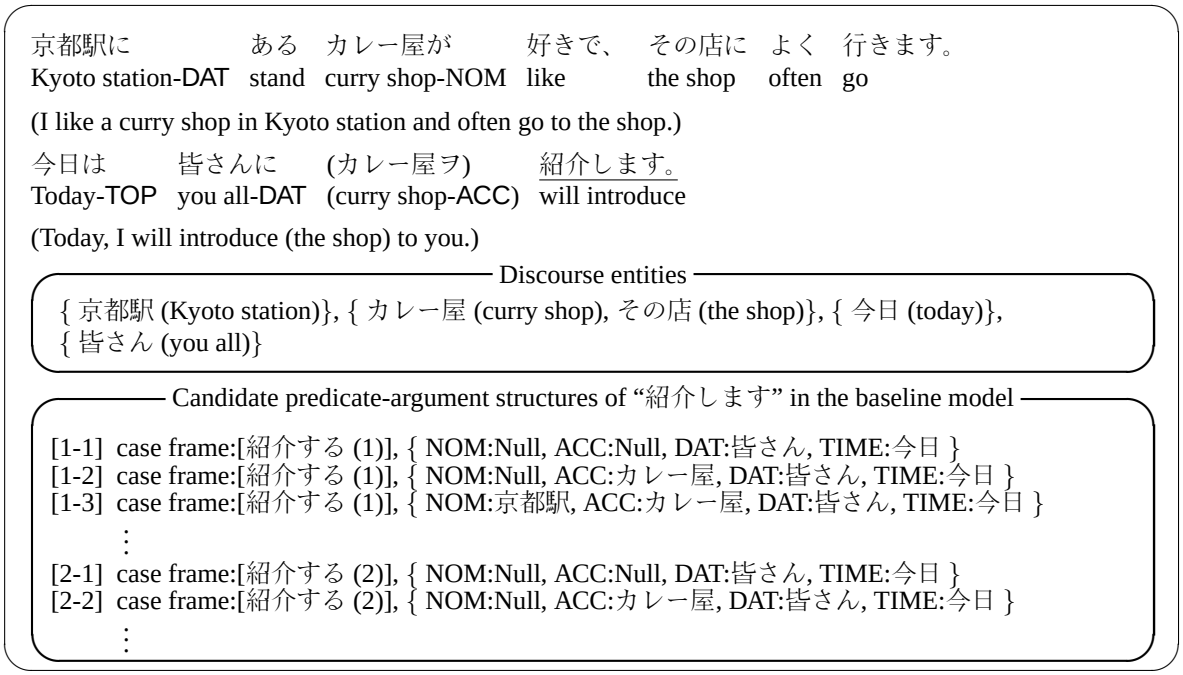


Figure 1: Examples of discourse entities and predicate-argument structures

1. Parse the input document and recognize named entities.
2. Resolve coreferential relations and set discourse entities.
3. Analyze the predicate-argument structure for each predicate using the following steps:
   (a) Generate candidate predicate-argument structures.
   (b) Calculate the score of each predicate-argument structure and select the one with the highest score.

We illustrate the details of the above procedure. First, we describe how to set the discourse entities in step 2. In our model, we treat referents of a zero pronoun using a unit called **discourse entity**, which is what mentions in a coreference chain are bound into. In Figure 1, we treat "カレー屋" (curry shop) and "その店" (the shop), which are in a coreference chain, as one discourse entity. In Figure 1, the discourse entity { カレー屋, その店 } is selected for the referent of the accusative case of the predicate "紹介します" (will introduce).

Next, we illustrate the PAS analysis in step 3. In step 3a, possible combinations of the case frame (cf) and the alignment (a) between case slots and discourse entities are listed. First, one case frame is selected from case frames for the predicate. Next, overt arguments, which have dependency relations with the predicate, are aligned to a case slot of the case frame. Finally, each of zero pronouns of remaining case slots is assigned to a discourse entity or is not assigned to any discourse entities. The case slot whose zero pronoun is not assigned to any discourse entities corresponds to the case that does not have a zero pronoun. In Figure 1, we show the examples of candidate PASs. In these examples, [紹介する (1)] and [紹介する (2)] are case frames corresponding to each meaning of "紹介する". Referents of each case slot are actually selected from discourse entities but are explained as a representative word for illustration. "Null" indicates that a case slot is not assigned to any discourse entities. Since alignments between case slots and discourse entities of the PAS [1-2] and [2-2] are the same but their case frames are different, we deal with them as discrete PASs. In this case, however, the results of zero reference resolution are the same.

We represent each PAS as a feature vector, which is described in section 3.1, and calculate a score of each PAS with the learned weights. Finally, the system outputs the PAS with the highest score.

iment of the baseline model, there is little difference between the results of these methods.

Type	Value	Description
Case frame	Log	Probabilities that {words, categories and named entity types} of e is assigned to c of cf
	Log	Generative probabilities of {words, categories and named entity types} of e
	Log	PMIs between {words, categories and named entity types} of e and c of cf
	Log	Max of PMIs between {words, categories and named entity types} of e and c of cf
	Log	Probability that c of cf is assigned to any words
	Log	Ratio of examples of c to ones of cf
	Binary	c of cf is {adjacent and obligate} case
Predicate	Binary	Modality types of p
	Binary	Honorific expressions of p
	Binary	Tenses of p
	Binary	p is potential form
	Binary	Modifier of p (predicate, noun and end of sentence)
	Binary	p is {dynamic and stative} verb
Context	Binary	Named entity types of e
	Integer	Number of mentions about e in t
	Integer	Number of mentions about e {before and after} p in t
	Binary	e is mentioned with post position "は" in a target sentence
	Binary	Sentence distances between e and p
	Binary	Location categories of e (Sasano and Kurohashi, 2011)
	Binary	e is mentioned at head of a target sentence
	Binary	e is mentioned with post position {"は" and "が" } at head of a target sentence
	Binary	e is mentioned at head of the first sentence
	Binary	e is mentioned with post position "は" at head of the first sentence
	Binary	e is mentioned at end of the first sentence
	Binary	e is mentioned with copula at end of the first sentence
	Binary	e is mentioned with noun phrase stop at end of the first sentence
	Binary	Salience score of e is larger than 1 (Sasano and Kurohashi, 2011)
other	Binary	c is assigned

Table 2: The features of $\phi_{assigned}(cf, c \leftarrow e, p, t)$

3.1 Feature Representation of Predicate-Argument Structure

When text t and target predicate p are given and PAS (cf, a) is chosen, we represent a feature vector of the PAS as $\phi(cf, a, p, t)$. $\phi(cf, a, p, t)$ consists of a feature vector $\phi_{overt\text{-}PAS}(cf, a, p, t)$ and feature vectors $\phi(cf, c/e, p, t)$. Where $\phi_{overt\text{-}PAS}(cf, a, p, t)$ corresponds to alignment between case slots and overt (not omitted) arguments and $\phi(cf, c/e, p, t)$ represents that a case slot c is assigned to a discourse entity e. If a case slot is assigned to an overt entity, $\phi(cf, c/e, p, t)$ is set to a zero vector.

Each feature vector $\phi(cf, c/e, p, t)$ consists of $\phi_A(cf, c/e, p, t)$ and $\phi_{NA}(cf, c/Null, p, t)$. $\phi_A(cf, c/e, p, t)$ becomes active when the case slot c is assigned to the discourse entity e and $\phi_{NA}(cf, c/Null, p, t)$ becomes active when the case slot c is not assigned to any discourse entities. For example, the PAS [1-2] in Figure 1 is represented as:
($\phi_{overt\text{-}PAS}$(紹介する (1), {NOM:Null, ACC:Null, NOM:皆さん, TIME:今日 }), $\mathbf{0}_{\phi_A}$,
ϕ_{NA}(紹介する (1), NOM/$Null$),
ϕ_A(紹介する (1), ACC/カレー屋),
$\mathbf{0}_{\phi_{NA}}, \mathbf{0}_{\phi_A}, \mathbf{0}_{\phi_{NA}}$). [5]
In our feature representation, the second and third terms correspond to the nominative case, the forth and fifth ones correspond to the accusative and the sixth and seventh ones correspond to the dative case.

We present the details of $\phi_{overt\text{-}PAS}(cf, a, p, t)$, $\phi_A(cf, c/e, p, t)$ and $\phi_{NA}(cf, c/Null, p, t)$. We use a score of the probabilistic PAS analysis (Kawahara and Kurohashi, 2006b) to $\phi_{overt\text{-}PAS}(cf, a, p, t)$. We list the features of $\phi_A(cf, c/e, p, t)$ in Table 2 and the features of $\phi_{NA}(cf, c/Null, p, t)$ in Table

[5] In the following example, p and t are sometimes omitted and $\mathbf{0}_\phi$ is 0 vector that has the same dimension as ϕ.

Type	Value	Description
Case frame	Log	Probability that c of cf is not assigned
	Log	Ratio of number of examples of c to ones of cf
	Binary	c of cf is {adjacent and obligate} case

Table 3: The features of $\phi_{NA}(cf, c/Null, p, t)$

3.

3.2 Weight Learning

In the previous section, we defined the feature vector $\phi(cf, a, p, t)$, which represents a PAS. In this section, we illustrate the learning method of the weight vector corresponding to the feature vector. The weight vector is learned by using a learning-to-rank algorithm.

In a corpus, gold-standard alignments a^* are manually annotated but case frames are not annotated. Since the case frames are constructed for each meaning, some of them are unsuitable for a usage of a prdicate in a context. If training data includes PASs (cf, a^*) whose cf is such case a frame as correct instances, these are harmful for training. Hence, we treat a case frame cf^* which is selected by a heuristic method as a correct case frame and remove (cf, a^*) which has other cf.

In particular, we make ranking data for the learning for each target predicate p in the following steps.

1. List possible PASs (cf, a) for predicate p.
2. Calculate a probabilistic zero reference resolution score for each (cf, a^*) and define the one with highest score as (cf^*, a^*).
3. Remove (cf, a^*) except (cf^*, a^*) from the learning instance.
4. Make ranking data that (cf^*, a^*) has a higher rank than other (cf, a).

In the above steps, we make ranking data for each predicate and use the ranking data collected from all target predicates as training data.

4 Corpus

In this work, we use Diverse Document Leads Corpus (DDLC) (Hangyo et al., 2012) for experiments. In DDLC, documents collected from the web are annotated with morpheme, syntax, named entity, coreference, PAS and A/R mention. Morpheme, syntax, named entity, coreference and PAS are annotated on the basis of Kyoto University Text Corpus (Kawahara et al., 2002). The PAS annotation includes zero reference information and the exophora referents are defined as five elements, [*author*], [*reader*], [*US(unspecified):person*], [*US:matter*] and [*US:situation*]. The A/R mentions are annotated to head phrases of compound nouns when the A/R mentions consist of compound nouns. If the A/R is mentioned by multiple expressions, only one of them is annotated with the A/R mention tag and all of these mentions are linked by a coreference chain. In other words, the A/R mentions are annotated to discourse entities. In the web site of an organization such as a company, the site administrator often writes the document on behalf of the organization. In such a case, the organization is annotated as the author.

5 Author/Reader Mention Detection

A/R mentions, which refer to A/R of a document, have different properties from other discourse entities. The A/R are mentioned as very various expressions such as personal pronouns, proper expressions and role expressions.

(4) こんにちは、 企画チームの
Hello project team-GEN
梅辻 $_{author}$ です。
am Umetsuji
(Hello, I'm Umetsuji on the project team.)

(5) 問題が あれば 管理人 $_{author}$ まで
problem-NOM exist to moderator
お知らせください。
let me know
(Please let me know if there are any problems.)

In example (4), the author is mentioned as "梅辻" (Umetsuji), which is the name of the author, and in example (5), the author is mentioned as "管理人" (moderator), which expresses the status of the author. Likewise, the reader is sometimes mentioned as "お客様" (customer) and others. However, since such expressions often refer to someone other than the A/R, whether an expression indicates the A/R of a document depends on the context of the document.

In English and other languages, the A/R mentions can be detected from coreference information because it can be assumed that the expression that has

a coreference relation with first or second personal pronoun is the A/R mention. However, since the A/R tend to be omitted and personal pronouns are rarely used in Japanese, it is difficult to detect the A/R mentions from coreference information. Because of these reasons, it is difficult to detect which discourse entity is the A/R mention from lexical information of the entities. In this study, the A/R mentions are detected from lexico-syntactic (LS) patterns in the document. We use a learning-to-rank algorithm to detect A/R mentions by using the LS patterns as features.

5.1 Author/Reader Detection Model

We use a learning-to-rank method for detecting A/R mentions. This method learns the ranking that entities of the A/R mentions have a higher rank than other discourse entities. Here, it is an important point that there are no A/R mentions in some documents. The documents in which the A/R mentions do not appear are classified into two types. The first type is a document that the A/R do not appear in the discourse of the document such as newspaper articles and novels. The second type is a document that the A/R appear in the discourse but all of their mentions are omitted. For example, in Figure 1, the author appears in the discourse (e.g. the nominative argument of "like") but is not mentioned explicitly. We introduce two pseudo entities corresponding to these types. The first pseudo entity "no A/R mention (discourse)" represents the document that the A/R do not appear in the discourse. It is considered that the document that the A/R do not appear have characteristics of writing style such that honorific expressions and request expressions are rarely used. This pseudo entity is represented as a document vector that consists of LS pattern features of the whole document, which reflect a writing style of a document. The second pseudo entity "no A/R mention (omitted)" represents the document in which all mentions of the A/R are omitted and this pseudo entity is represented as 0 vector. Since a decision score of this pseudo entity is allways 0, discourse entities whose score is lower than the score of this pseudo entity can be treated as a negative example in a binary classification.

When there are A/R mentions in a document, we make ranking data where the discourse entity of the A/R mention has a higher rank than other discourse entities and "no A/R mention" pseudo entities. When the A/R do not appear in the discourse, we make ranking data where "no A/R mention (discourse)" has a higher rank than all discourse entities and "no A/R mention (omitted)". When the A/R appear in the discourse but all mentions are omitted, we make ranking data where "no A/R mention (omitted)" has a higher rank than all discourse entities and "no A/R mention (discourse)". We judge that the A/R appear in the discourse if the A/R appear as a referent of zero reference in gold-standard PASs and this judgment is used only in the training phase. After making the ranking data for each document, all of the ranking data are merged and the merged data is fed into the learning-to-rank model.

For the A/R mention detection, we calculate the score of all discourse entities and the pseudo entities and select the discourse entity with the highest score to the A/R mention. If any "no A/R mention" have the highest score, we decide that there are no A/R mentions in the document.

5.2 Lexico-Syntactic Patterns

For each discourse entity, phrases of the discourse entity, its parent and their dependency relations are used to make LS patterns that represent the discourse entity. When a discourse entity is mentioned multiple times, the phrases of all mentions are used to make the LS patterns. LS patterns of phrases are made by generalizing these phrases on various levels (types). LS patterns of dependencies are made from combining the LS patterns of phrases.

Table 4 lists generalization types. On the *word* type, we make a phrase LS pattern by generalizing each content word and jointing them. For example, a LS pattern of the phrase "ぼくは" generalized on the <representative form> is "僕は". The *word+* type is the same as *word* except all content words are generalized on the <part of speech and conjugation>. For example, a LS pattern of the dependency relation "太郎は → 走った" generalized on the <named entity> is "NE:PERSON+は → verb:past". We also use the LS patterns of generalized individual morphemes. On the *phrase* type, each phrase is generalized according to the information assigned to the phrase and all content words are generalized on the <part of speech and conjugation> if the information

Unit	Type	Example (original phrase)
word	<no generalization>	僕は (僕は)
	<original form>	走った (走る)
	<representative form>	僕は (ぼくは)
	<part of speech and conjugation>	verb:past (走った)
word+	<category>	Category:PERSON+は (僕は)
	<named entity>	NE:PERSON+は (太郎は)
	<first person pronoun>	FirstPersonPronoun+は (僕は)
	<second person pronoun>	SecondPersonPronoun+に (あなたに)
phrase	<modality>	modality:request (お問い合わせください)
	<honorific expression>	honorific:modest (お送りします)
	<attached words>	ください (お問い合わせください)

Table 4: Generalization types of the LS patterns

is not assigned to the phrase.

For "no A/R mention (discourse)" instance, the above features of all mentions, including verbs and adjectives, and their dependencies in the document are gathered and used as the features representing the instance.

6 Zero Reference Resolution Considering Exophora and Author/Reader Mentions

In this section, we describe the zero reference resolution system that considers the zero exophora and the A/R mentions. The proposed model resolves zero reference as a part of the PAS analysis based on the baseline model.

The proposed model analyzes the PASs in the following steps:

1. Parse the input document and recognize named entities.
2. Resolve coreferential relations and set discourse entities.
3. Detect the A/R mentions of the document.
4. Set pseudo entities from the estimated A/R mentions.
5. Analyze the PAS for each predicate using the same procedure as the baseline model.

The differences form baseline model are the estimation of the A/R mentions in step 3 and the setting of pseudo entities in step 4.

6.1 Pseudo Entities and Author/Reader Mentions for Zero Exophora

In the baseline model, referents of zero pronouns are selected form discourse entities. The proposed model adds pseudo entities([*author*], [*reader*], [*US:person*] (unspecified:person) and [*US:others*] (unspecified:others)[6]) to deal with zero exophora.

When the A/R mentions appear in a document, the A/R pseudo entities raise an issue. The zero endophora are given priority to zero exophora. In other words, the A/R mentions are selected to the referents in preference to pseudo entities when there are A/R mentions. Therefore, when the system estimates that A/R mentions appear, the A/R pseudo entities are not created.

In the PAS analysis, referents are selected from discourse entities and the pseudo entities. A zero reference is the zero exophora when a case slot is assigned to pseudo entities. Candidate PASs of "紹介します" in Figure 1 are shown in Figure 2.

6.2 Feature Representation of Predicate Argument Structure

In the same way as the baseline model, the proposed model represents a PAS as a feature vector that consists of the feature vector $\phi_{overt\text{-}PAS}(cf, a, p, t)$ and the feature vectors $\phi(cf, c/e, p, t)$. The difference from the baseline model is a composition of $\phi_A(cf, c/e, p, t)$. In the proposed model, each $\phi_A(cf, c/e)$ is composed of vectors, $\phi_{discourse}(cf, c/e)$, $\phi_{[author]}(cf, c/e)$, $\phi_{[reader]}(cf, c/e)$, $\phi_{[US:person]}(cf, c/e)$, $\phi_{[US:others]}(cf, c/e)$ and $\phi_{max}(cf, c/e)$. Their contents and dimensions are the same and similar to $\phi_A(cf, c/e)$ of the baseline model the except for the

[6]We merge [*US:matter*] and [*US:situation*] because of the small amount of [*US:situation*] in the corpus.

[1-1] case frame:[紹介する (1)], { NOM:[*author*], ACC:Null, DAT:皆さん $_{reader}$, TIME:今日 }
[1-2] case frame:[紹介する (1)], { NOM:[*US:person*], ACC:Null, DAT:皆さん $_{reader}$, TIME:今日 }
[1-3] case frame:[紹介する (1)], { NOM:[*author*], ACC:カレー屋, DAT:皆さん $_{reader}$, TIME:今日 }
[1-4] case frame:[紹介する (1)], { NOM:京都駅, ACC:カレー屋, DAT:皆さん $_{reader}$, TIME:今日 }
[1-5] case frame:[紹介する (1)], { NOM:[*author*], ACC:[*US:others*], DAT:皆さん $_{reader}$, TIME:今日 }
⋮
[2-1] case frame:[紹介する (2)], { NOM:[*author*], ACC:Null, DAT:皆さん $_{reader}$, TIME:今日 }
[2-2] case frame:[紹介する (2)], { NOM:[*US:person*], ACC:Null, DAT:皆さん $_{reader}$, TIME:今日 }
⋮

Figure 2: Candidate predicate-argument structures of "紹介します" in the proposed model

	Expressions	Categories
author	私 (I), 我々 (we), 俺 (I), 僕 (I), 当社 (our company), 弊社 (our company), 当店 (our shop)	PERSON, ORGANIZATION
reader	あなた (you), 客 (customer), 君 (you), 皆様 (you all), 皆さん (you all), 方 (person), 方々 (people)	PERSON
US:person	人 (person), 人々 (people)	PERSON
US:others	もの (thing), 状況 (situation)	all categories except PERSON and ORGANIZATION

Table 5: Expressions and categories for pseudo entities

addition of a few features described in section 6.3.

$\phi_{discourse}$ corresponds to the discourse entities, which are mentioned explicitly and becomes active when e is a discourse entity including the A/R mentions. $\phi_{discourse}$ is the same as ϕ_A of the baseline model and the difference is explained in section 6.3. $\phi_{[author]}$ and $\phi_{[reader]}$ become active when e is [*author*]/[*reader*] or the discourse entity corresponding to the A/R mention. In particular, when e is the discourse entity corresponding to the A/R mention, both $\phi_{discourse}$ and $\phi_{[author]}$/$\phi_{[reader]}$ become active. This representation gives the A/R mentions the properties of the discourse entity and the A/R. $\phi_{[US:person]}$ and $\phi_{[US:others]}$ become active when e is [*US:person*] and [*US:others*].

Because $\phi_{[author]}$, $\phi_{[reader]}$, $\phi_{[US:person]}$ and $\phi_{[US:others]}$ correspond to the pseudo entities, which are not mentioned explicitly, we cannot use word information such as expressions and categories. We assume that the pseudo entities have expressions and categories shown in Table 5 and use these to calculate case frame features. Finally, ϕ_{max} consists of the highest value of correspondent feature of the above feature vectors.

6.3 Author/Reader Mention Score

We add A/R mention score features to the feature vector $\phi_A(cf, c/e, p, t)$ described in Table 2. The A/R mention scores are the discriminant function scores of the A/R mention detection. When e is the A/R mention, we set the A/R mention score to the feature.

7 Experiments

7.1 Experimental Settings

We used 1,000 documents from DDLC and performed 5-fold cross-validation. 1,440 zero endophora and 1,935 zero exophora are annotated in these documents. 258 documens are annotated with author mentions and 105 documens are annotated with reader mentions. We used gold-standard (manually annotated) morphemes, named entities, dependency structures and coreference relations to focus on the A/R detection and the zero reference resolution. We used SVM^{rank}[7] for the learning-to-rank method of the A/R detection and the PAS analysis. The categories of words are given by the morphological analyzer JUMAN[8]. Named entities and predicate features (e.g., honorific expressions, modality)

[7] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[8] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

		System output		
		Exist		None
		Correct	Wrong	
Gold -standard	Exist	140	6	112
	None	-	38	704

Table 6: Result of the author mention detection

		System output		
		Exist		None
		Correct	Wrong	
Gold -standard	Exist	56	2	47
	None	-	23	872

Table 7: Result of the reader mention detection

are given by the syntactic parser KNP.[9]

7.2 Results of Author/Reader Mention Detection

We show the results of the author and reader mention detection in Table 6 and Table 7. In these tables, "exist" indicates numbers of documents in which the A/R mentions are manually annotated or our system estimated that some discourse entities are A/R mentions. From these results, the A/R mentions including "none" can be predicted to accuracies of approximately 80%. On the other hand, the recalls are not particularly high: the recall of author is 140/258 and the recall of reader is 56/105. This is because the documents in which the A/R do not appear are more than the ones in which the A/R appear and the system prefers to output "no author/reader mention" as the result of training.

7.3 Results of Zero Reference Resolution

We show the results of zero reference resolution in Table 8 and Table 9. The difference between the baseline and the proposed model is statistically significant ($p < 0.05$) from the McNemar's test. In Table 8, we evaluate only the zero endophora for comparison to the baseline model, which deals with only the zero endophora. "Proposed model (estimate)" shows the result of the proposed model which estimated the A/R mentions and "Proposed model (gold-standard)" shows the result of the proposed model which is given the A/R mentions of gold-standard from the corpus.

From Table 8, considering the zero exophora and the A/R mentions improves accuracy of zero endophora resolution as well as zero reference resolution including zero exophora.

[9] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP

	Recall	Precision	F1
Baseline	0.269	0.377	0.314
Proposed model (estimate)	0.282	0.448	0.346
Proposed model (gold-standard)	0.388	0.522	0.445

Table 8: Results of zero endophora resolution

	Recall	Precision	F1
Baseline	0.115	0.377	0.176
Proposed model (estimate)	0.317	0.411	0.358
Proposed model (gold-standard)	0.377	0.485	0.424

Table 9: Results of zero reference resolution

From Table 8 and Table 9, the proposed model given the gold-standard A/R mentions achieves extraordinarily high accuracies. This result indicates that improvement of the A/R mention detection improves the accuracy of zero reference resolution in the proposed model.

8 Conclusion

This paper presented a zero reference resolution model considering exophora and author/reader mentions. In the experiments, our proposed model achieves higher accuracy than the baseline model. As future work, we plan to improve the author/reader detection model to improve the zero reference resolution.

References

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544, Bali,Indonesia, November. Faculty of Computer Science, Universitas Indonesia.

Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer. 1998. Learning preference relations for information retrieval. In *ICML-98 Workshop: text categorization and machine learning*, pages 80–84.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora

resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 625–632, Sydney, Australia, July. Association for Computational Linguistics.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85–88, Suntec, Singapore, August. Association for Computational Linguistics.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Daisuke Kawahara and Sadao Kurohashi. 2006a. Case frame compilation from the web using high-performance computing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1344–1347.

Daisuke Kawahara and Sadao Kurohashi. 2006b. A fully-lexicalized probabilistic model for japanese syntactic and case structure analysis. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 176–183, New York City, USA, June. Association for Computational Linguistics.

Daisuke Kawahara, Sadao Kurohashi, and Koiti Hasida. 2002. Construction of a japanese relevance-tagged corpus. In *Proc. of The Third International Conference on Language Resources Evaluation*, May.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882–891, Cambridge, MA, October. Association for Computational Linguistics.

Massimo Poesio, Olga Uryupina, and Yannick Versley. 2010. Creating a coreference resolution system for italian. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Luz Rello, Ricardo Baeza-Yates, and Ruslan Mitkov. 2012. Elliphant: Improved automatic detection of zero subjects and impersonal constructions in spanish. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 706–715. Association for Computational Linguistics.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 758–766, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for japanese zero anaphora resolution. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 769–776, Manchester, UK, August. Coling 2008 Organizing Committee.

A Dataset for Research on Short-Text Conversation *

Hao Wang[§] **Zhengdong Lu**[‡] **Hang Li**[‡] **Enhong Chen**[§]
[§] xdwangh@mail.ustc.edu.cn [‡]lu.zhengdong@huawei.com
[‡]hangli.hl@huawei.com [§]cheneh@ustc.edu.cn
[§]Univ. of Sci & Tech of China, China [‡]Noah's Ark Lab, Huawei Technologies, Hong Kong

Abstract

Natural language conversation is widely regarded as a highly difficult problem, which is usually attacked with either rule-based or learning-based models. In this paper we propose a retrieval-based automatic response model for short-text conversation, to exploit the vast amount of short conversation instances available on social media. For this purpose we introduce a dataset of short-text conversation based on the real-world instances from Sina Weibo (a popular Chinese microblog service), which will be soon released to public. This dataset provides rich collection of instances for the research on finding natural and relevant short responses to a given short text, and useful for both training and testing of conversation models. This dataset consists of both naturally formed conversations, manually labeled data, and a large repository of candidate responses. Our preliminary experiments demonstrate that the simple retrieval-based conversation model performs reasonably well when combined with the rich instances in our dataset.

1 Introduction

Natural language conversation is one of the holy grail of artificial intelligence, and has been taken as the original form of the celebrated Turing test. Previous effort in this direction has largely focused on analyzing the text and modeling the state of the conversation through dialogue models, while in this paper we take one step back and focus on a much easier task of finding the response for a given short text. This task is in clear contrast with previous effort in dialogue modeling in the following two aspects

- we do not consider the context or history of conversations, and assume that the given short text is self-contained;
- we only require the response to be natural, relevant, and human-like, and do not require it to contain particular opinion, content, or to be of particular style.

This task is much simpler than modeling a complete dialogue session (e.g., as proposed in Turing test), and probably not enough for real conversation scenario which requires often several rounds of interactions (e.g., automatic question answering system as in (Litman et al., 2000)). However it can shed important light on understanding the complicated mechanism of the interaction between an utterance and its response. The research in this direction will not only instantly help the applications of short session dialogue such as automatic message replying on mobile phone and the chatbot employed in voice assistant like Siri[1], but also it will eventually benefit the modeling of dialogues in a more general setting.

Previous effort in modeling lengthy dialogues focused either on rule-based or learning-based models (Carpenter, 1997; Litman et al., 2000; Williams and Young, 2007; Schatzmann et al., 2006; Misu et al., 2012). This category of approaches require relatively less data (e.g. reinforcement learning based) for

*The work is done when the first author worked as intern at Noah's Ark Lab, Huawei Techologies.

[1]http://en.wikipedia.org/wiki/Siri

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 935–945,
Seattle, Washington, USA, 18-21 October 2013.

training or no training at all, but much manual effort in designing the rules or the particular learning algorithms. In this paper, we propose to attack this problem using an alternative approach, by leveraging the vast amount of training data available from the social media. Similar ideas have appeared in (Jafarpour and Burges, 2010; Leuski and Traum, 2011) as an initial step for training a chatbot.

With the emergence of social media, especially microblogs such as Twitter, in the past decade, they have become an important form of communication for many people. As the result, it has collected conversation history with volume previously unthinkable, which brings opportunity for attacking the conversation problem from a whole new angle. More specifically, instead of generating a response to an utterance, we pick a massive suitable one from the candidate set. The hope is, with a reasonable retrieval model and a *large enough* candidate set, the system can produce fairly natural and appropriate responses.

This retrieval-based model is somewhat like non-parametric model in machine learning communities, which performs well only when we have abundant data. In our model, it needs only a relatively small *labeled* dataset for training the retrieval model, but requires a rather large *unlabeled* set (e.g., one million instances) for candidate responses. To further promote the research in similar direction, we create a dataset for training and testing the retrieval model, with a candidate responses set of reasonable size. Sina Weibo is the most popular Twitter-like microblog service in China, hosting over 500 million registered users and generating over 100 million messages per day [2]. As almost all microblog services, Sina Weibo allows users to comment on a published post[3], which forms a natural one-round conversation. Due to the great abundance of those (post, response) pairs, it provides an ideal data source and test bed for one-round conversation. We will make this dataset publicly available in the near future.

[2] http://en.wikipedia.org/wiki/Sina_Weibo

[3] Actually it also allows users to comment on other users' comments, but we will not consider that in the dataset.

2 The Dialogues on Sina Weibo

Sina Weibo is a Twitter-like microblog service, on which a user can publish short messages (will be referred to as *post* in the remainder of the paper) visible to public or a group specified by the user. Similar to Twitter, Sina Weibo has the word limit of 140 Chinese characters. Other users can comment on a published post, with the same length limit, as shown in the real example given in Figure 6 (in Chinese). Those comments will be referred to as *responses* in the remainder of the paper.

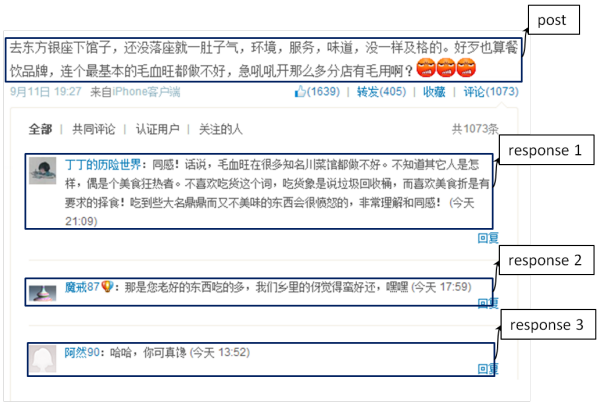


Figure 1: An example of Sina Weibo post and the comments it received.

We argue that the (post, response) pairs on Sina Weibo provide rather valuable resource for studying *one round* dialogue between users. The comments to a post can be of rather flexible forms and diverse topics, as illustrated in the example in Table 1. With a post stating the user's status (traveling to Hawaii), the comments can be of quite different styles and contents, but apparently all appropriate.

In many cases, the (post, response) pair is self-contained, which means one does not need any background and contextual information to get the main point of the conversation (Examples of that include the responses from **B, D, G** and **H**). In some cases, one may need extra knowledge to understand the conversation. For example, the response from user **E** will be fairly elusive if taken out of the context that **A**'s Hawaii trip is for an international conference and he is going to give a talk there. We argue that the number of self-contained (post, response) pairs is vast, and therefore the extracted (post, re-

Post	
User A:	*The first day at Hawaii. Watching sunset at the balcony with a big glass of wine in hand.*
Responses	
User B:	*Enjoy it & don't forget to share your photos!*
User C:	*Please take me with you next time!*
User D:	*How long are you going to stay there?*
User E:	*When will be your talk?*
User F:	*Haha, I am doing the same thing right now. Which hotel are you staying in?*
User G:	*Stop showing-off, buddy. We are still coding crazily right now in the lab.*
User H:	*Lucky you! Our flight to Honolulu is delayed and I am stuck in the airport. Chewing French fries in MacDonald's right now.*

Table 1: A typical example of Sina Weibo post and the comments it received. The original text is in Chinese, and we translated it into English for easy access of readers. We did the same thing for all the examples throughout this paper.

sponse) pairs can serve as a rich resource for exploring rather sophisticated patterns and structures in natural language conversation.

3 Content of the Dataset

The dataset consists of three parts, as illustrated in Figure 2. Part 1 contains the original (post, response) pairs, indicated by the dark-grey section in Figure 2. Part 2, indicated by the light-gray section in Figure 2, consists labeled (post, response) pairs for some Weibo posts, including positive and negative examples. Part 3 collects all the responses, including but not limited to the responses in Part 1 and 2. Some of the basic statistics are summarized in Table 2.

# posts	# responses	vocab.	# labeled pairs
4,6345	1,534,874	105,732	12,427

Table 2: Some statistics of the dataset

Original (Post, Response) Pairs This part of dataset gives (post, response) pairs naturally presented in the microblog service. In other words, we create a (post, response) pair there when the response is actually given to the post in Sina Weibo. The part of data is noisy since the responses given to a Weibo post could still be inappropriate for different reasons, for example, they could be spams or targeting some responses given earlier. We have $628,833$ pairs.

Labeled Pairs This part of data contains the (post, response) pairs that are labeled by human. Note that 1) the labeling is only on a small subset of posts, and 2) for each selected post, the labeled responses are not originally given to it. The labeling is done in an active manner (see Section 4 for more details), so the obtained labels are much more informative than the those on randomly selected pairs (over 98% of which are negative). This part of data can be directly used for training and testing of retrieval-based response models. We have labeled 422 posts and for each of them, about 30 candidate responses.

Responses This part of dataset contains only responses, but they are not necessarily for a certain post. These extra responses are mainly filtered out by our data cleaning strategy (see Section 4.2) for original (post, response) pairs, including those from filtered-out Weibo posts and those addressing other responses. Nevertheless, those responses are still valid candidate for responses. We have about 1.5 million responses in the dataset.

3.1 Using the Dataset for Retrieval-based Response Models

Our data can be used for training and testing of retrieval-based response model, or just as a bank of responses. More specifically, it can be used in at least the following three ways.

Training Low-level Matching Features The rather abundant original (post, response) pairs provide rather rich supervision signal for learning different matching patterns between a post and a response. These matching patterns could be of dif-

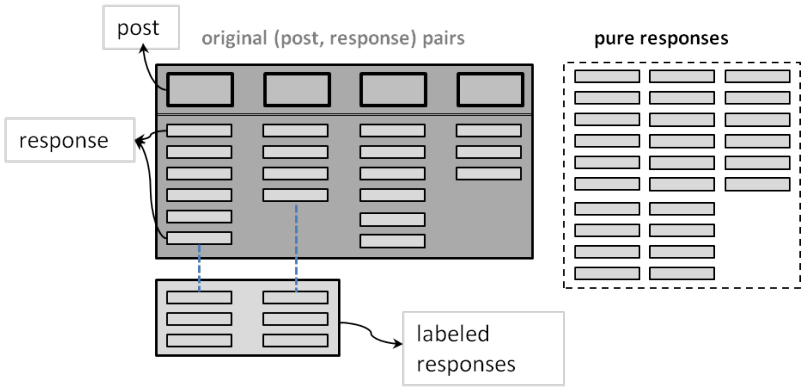


Figure 2: Content of the dataset.

ferent levels. For example, one may discover from the data that when the word "Hawaii" occurs in the post, the response are more likely to contain words like "trip", "flight", or "Honolulu". On a slightly more abstract level, one may learn that when an entity name is mentioned in the post, it tends to be mentioned again in the response. More complicated matching pattern could also be learned. For example, the response to a post asking "how to" is statistically longer than average responses. As a particular case, Ritter et al. (2011) applied translation model (Brown et al., 1993) on similar parallel data extracted from Twitter in order to extract the word-to-word correlation. Please note that with more sophisticated natural language processing, we can go beyond bag-of-words for more complicated correspondence between post and response.

Training Automatic Response Models Although the original (post, response) pairs are rather abundant, they are not enough for discriminative training and testing of retrieval models, for the following reasons. In the labeled pairs, both positive and negative ones are ranked high by some baseline models, and hence more difficult to tell apart. This supervision will naturally tune the model parameters to find the real good responses from the seemingly good ones. Please note that without the labeled negative pairs, we need to generate negative pairs with randomly chosen responses, which in most of the cases are too easy to differentiate by the ranking model and cannot fully tune the model parameters. This intuition has been empirically verified by our experiments.

Testing Automatic Response Models In testing a retrieval-based system, although we can simply use the original responses associated with the query post as positive and treat all the others as negative, this strategy suffers from the problem of spurious negative examples. In other words, with a reasonably good model, the retrieved responses are often good even if they are not the original ones, which brings significant bias to the evaluation. With the labeled pairs, this problem can be solved if we limit the testing only in the small pool of labeled responses.

3.2 Using the Dataset for Other Purposes

Our dataset can also be used for other researches related to short-text conversations, namely anaphora resolution, sentiment analysis, and speech act analysis, based on the large collection of original (post, response) pairs. For example, to determine the sentiment of a response, one needs to consider both the original post as well as the observed interaction between the two. In Figure 3, if we want to understand user's sentiment towards the "invited talk" mentioned in the post, the two responses should be taken as positive, although the sentiment in the mere responses is either negative or neutral.

4 Creation of the Dataset

The (post, comment) pairs are sampled from the Sina Weibo posts published by users in a loosely connected community and the comments they received (may not be from this community). This community is mainly posed of professors, researchers, and students of natural language processing (NLP) and related areas in China, and the users

Query Post:	十点半主楼有XXX博士的关于深度学习的讲座，有兴趣的同学不要错过！ *There is a talk on deep learning given by Dr. XXX on deep learning in the main building. Come if you are interested*
Response1:	太悲催了，我十点钟有节课 *Damn it! I have a class at 10 am*
Response2:	到底主楼哪个房间啊？ *Come on, which room in the main building?*

Figure 3: An example (original Chinese and the English translation) on the difficulty of sentiment analysis on responses.

commonly followed them.

The creation process of the dataset, as illustrated in Figure 4, consists of three consecutive steps: 1) crawling the community of users, 2) crawling their Weibo posts and their responses, 3) cleaning the data, with more details described in the remainder of this section.

4.1 Sampling Strategy

We take the following sampling strategy for collecting the (post, response) pairs to make the topic relatively focused. We first locate 3,200 users from a loosely connected community of Natural Language Processing (NLP) and Machine Learning (ML) in China. This is done through crawling followees[4] of ten manually selected seed users who are NLP researchers active on Sina Weibo (with no less than 2 posts per day on average) and popular enough (with no less than 100 followers).

We crawl the posts and the responses they received (not necessarily from the crawled community) for two months (from April 5th, 2013, to June 5th, 2013). The topics are relatively limited due to our choice of the users, with the most saliently ones being:

- **Research:** discussion on research ideas, papers, books, tutorials, conferences, and researchers in NLP and machine learning, etc;
- **General Arts and Science:** mathematics, physics, biology, music, painting, etc;
- **IT Technology:** Mobile phones, IT companies, jobs opportunities, etc;
- **Life:** traveling (both touring or conference trips), food, photography, etc.

[4]When user A follows user B, A is called B's follower, and B is called A's followee.

4.2 Processing, Filtering, and Data Cleaning

On the crawled posts and responses, we first perform a four-step filtering on the post and responses

- We first remove the Weibo posts and their responses if the length of post is less than 10 Chinese characters or the length of the response is less than 5 characters. The reason for that is two-fold: 1) if the text is too short, it can barely contain information that can be reliably captured, e.g. the following example

P:	*Three down, two to go.*

  and 2) some of the posts or responses are too general to be interesting for other cases, e.g. the response in the example below,

P:	*Nice restaurant. I'd strong recommend it. Everything here is good except the long waiting line*
R:	*wow.*

- In the remained posts, we only keep the first 100 responses in the original (post, response) pairs, since we observe that after the first 100 responses there will be a non-negligible proportion of responses addressing things other than the original Weibo post (e.g., the responses given earlier). We however will still keep the responses in the bank of responses.
- The last step is to filter out the potential advertisements. We will find the long responses that have been posted more than twice on different posts and scrub them out of both original (post, response) pairs and the response repository.

For the remained posts and responses, we remove the punctuation marks and emoticons, and use ICTCLAS (Zhang et al., 2003) for Chinese word segmentation.

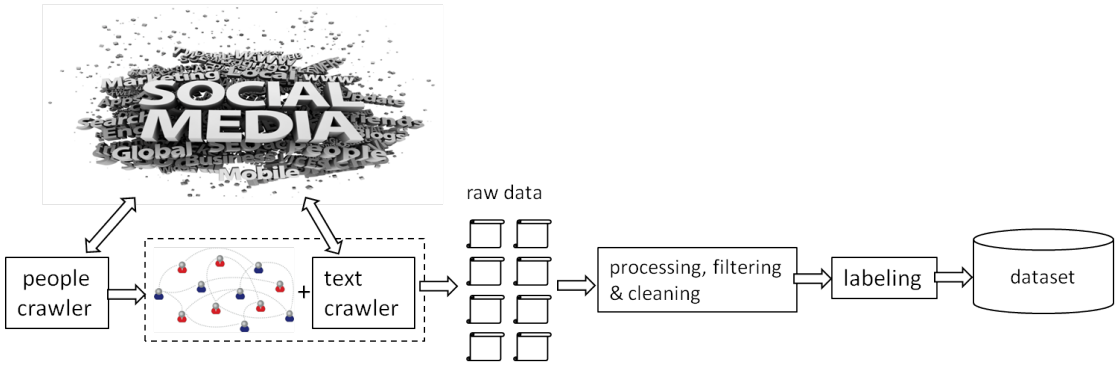


Figure 4: Diagram of the process for creating the dataset.

4.3 Labeling

We employ a pooling strategy widely used in information retrieval for getting the instance to label (Voorhees, 2002). More specifically, for a given post, we use three baseline retrieval models to each select 10 responses (see Section 5 for the description of the baselines), and merge them to form a much reduced candidate set with size ≤ 30. Then we label the reduced candidate set into "suitable" and "unsuitable" categories. Basically we consider a response suitable for a given post if we cannot tell whether it is an original response. More specifically the suitability of a response is judged based on the following three criteria[5]:

Semantic Relevance: This requires the content of the response to be semantically relevant to the post. As shown in the example right below, the post **P** is about soccer, and so is response **R1** (hence semantically relevant), whereas response **R2** is about food (hence semantically irrelevant).

P:	*There are always 8 English players in their own penalty area. Unbelievable!*
R1:	*Haha, it is still 0:0, no goal so far.*
R2:	*The food in England is horrible.*

Another important aspect of semantic relevance is the entity association. This requires the entities in the response to be correctly aligned with those in the post. In other words, if the post is about entity A, while the response is about entity B, they are very likely to be mismatched. As shown in the following example, where the original post is about Paris, and the response **R2** talks about London:

P:	*It is my last day in Paris. So hard to say goodbye.*
R1:	*Enjoy your time in Paris.*
R2:	*Man, I wish I am in London right now.*

This is however not absolute, since a response containing a different entity could still be sound, as demonstrated by the following two responses to the post above

R1:	*Enjoy your time in France.*
R2:	*The fall of London is nice too.*

Logic Consistency: This requires the content of the response to be logically consistent with the post. For example, in the table right below, post **P** states that the Huawei mobile phone "Honor" is already in the market of mainland China. Response **R1** talks about a personal preference over the same phone model (hence logically consistent), whereas **R2** asks the question the answer to which is already clear from **P** (hence logically inconsistent).

P:	*HUAWEI's mobile phone, Honor, sells well in Chinese Mainland.*
R1:	*HUAWEI Honor is my favorite phone*
R2:	*When will HUAWEI Honor get to the market in mainland China?*

Speech Act Alignment: Another important factor in determining the suitability of a response is the

[5]Note that although our criteria in general favor short and general answers like "Well said!" or "Nice", most of these general answers have already been filtered out due to their length (see Section 4.2).

speech act. For example, when a question is posed in the Weibo post, a certain act (e.g., answering or forwarding it) is expected. In the example below, post **P** asks a special question about location. Response **R1** and **R2** either forwards or answers the question, whereas **R3** is a negative sentence and therefore does not align well in speech act.

P:	*Any one knows where KDD will be held the year after next?*
R1:	*co-ask. Hopefully Europe*
R2:	*New York, as I heard*
R3:	*No, it is still in New York City*

5 Retrieval-based Response Model

In a retrieval-based response model, for a given post x we pick from the candidate set the response with the highest ranking score, where the score is the ensemble of several individual matching features

$$\mathsf{score}(x, y) = \sum_{i \in \Omega} w_i \Phi_i(x, y). \quad (1)$$

with y stands for a candidate response.

We perform a two-stage retrieval to handle the scalability associated with the massive candidate set, as illustrated in Figure 5. In **Stage I**, the system employs several fast baseline matching models to retrieve a number of candidate responses for the given post x, forming a much reduced candidate set $\mathcal{C}_{\mathbf{x}}^{(reduced)}$. In **Stage II**, the system uses a ranking function with more and sophisticated features to further evaluate all the responses in $\mathcal{C}_{\mathbf{x}}^{(reduced)}$, returning a matching score for each response. Our response model then decides whether to respond and which candidate response to choose.

In **Stage II**, we use the linear score function defined in Equation 1 with 15 features, trained with RankSVM (Joachims, 2002). The training and testing are both performed on the 422 labeled posts, with about 12,000 labeled (post, response) pairs. We use a 5-fold cross validation with a fixed penalty parameter for slack variable. [6]

5.1 Baseline Matching Models

We use the following matching models as the baseline model for **Stage I** fast retrieval. Moreover, the matching features used in the ranking function in **Stage II** are generated, directly or indirectly, from the those matching models:

POST-RESPONSE SEMANTIC MATCHING: This particular matching function relies on a learned mapping from the original sparse representation for text to a low-dimensional but dense representation for both Weibo posts and responses. The level of matching score between a post and a response can be measured as the inner product between their images in the low-dimensional space

$$\mathsf{SemMatch}(x, y) = \mathbf{x}^\top L_{\mathcal{X}} L_{\mathcal{Y}}^\top \mathbf{y}. \quad (2)$$

where $\mathbf{x}$ and $\mathbf{y}$ are respectively the 1-in-N representations of x and y. This is to capture the semantic matching between a Weibo post and a response, which may not be well captured by a word-by-word matching. More specifically, we find $L_{\mathcal{X}}$ and $L_{\mathcal{Y}}$ through a large margin variant of (Wu et al., 2013)

$$\begin{aligned} \arg\min_{L_{\mathcal{X}}, L_{\mathcal{Y}}} \quad & \sum_i \max(1 - \sum_i \mathbf{x}_i^\top L_{\mathcal{X}} L_{\mathcal{Y}}^\top \mathbf{y}_i, 0) \\ s.t. \quad & \|L_{n,\mathcal{X}}\|_1 \le \mu_1,\ n = 1, 2, \cdots, N_x \\ & \|L_{m,\mathcal{Y}}\|_1 \le \mu_1,\ m = 1, 2, \cdots, N_y \\ & \|L_{n,\mathcal{X}}\|_2 = \mu_2,\ n = 1, 2, \cdots, N_x \\ & \|L_{m,\mathcal{Y}}\|_2 = \mu_2\, m = 1, 2, \cdots, N_y. \end{aligned}$$

where i indices the original (post, response) pairs. Our experiments (Section 6) indicate that this simple linear model can learn meaningful patterns, due to the massive training set. For example, the image of the word "Italy" in the post in the latent space matches well word "Sicily", "Mediterranean sea" and "travel". Once the mapping $L_{\mathcal{X}}$ and $L_{\mathcal{Y}}$ are learned, the semantic matching score $\mathbf{x}^\top L_{\mathcal{X}} L_{\mathcal{Y}}^\top \mathbf{y}$ will be treated as a feature for modeling the overall suitability of y as a response to post x.

POST-RESPONSE SIMILARITY: Here we use a simple vector-space model for measuring the similarity between a post and a response

$$\mathsf{sim}_{PR}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}. \quad (3)$$

Although it is not necessarily true that a good response has many common words as the post, but this measurement is often helpful in finding relevant responses. For example, when the post and response

[6] The performance is fairly insensitive to the choice of the penalty, so we only report the result with a typical choice of it.

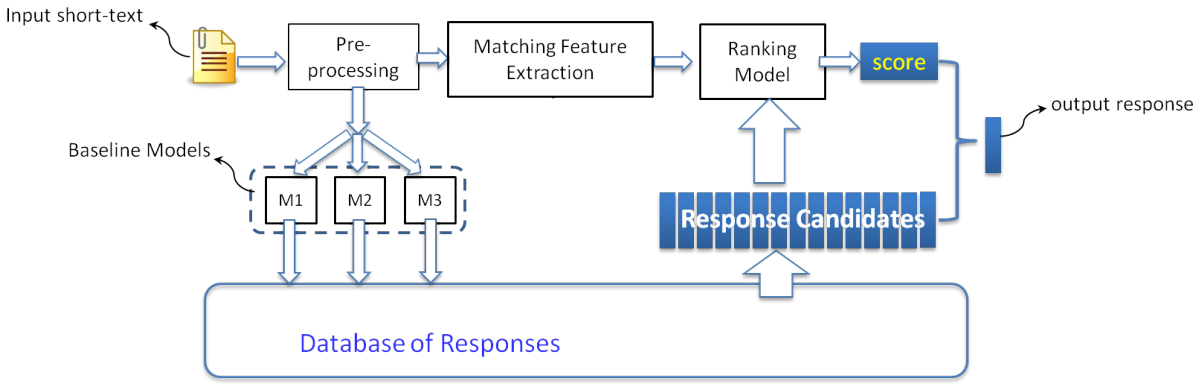


Figure 5: Diagram of the retrieval-based automatic response system.

both have "National Palace Museum in Taipei", it is a strong signal that they are about similar topics. Unlike the semantic matching feature, this simple similarity requires no learning and works on infrequent words. Our empirical results show that it can often capture the Post-Response relation failed with semantic matching feature.

POST-POST SIMILARITY: The basic idea here is to find posts similar to x and use their responses as the candidates. Again we use the vector space model for measuring the post-post similarity

$$\mathsf{sim}_{PP}(x, x') = \frac{\mathbf{x}^\top \mathbf{x}'}{\|\mathbf{x}\|\|\mathbf{x}'\|}. \quad (4)$$

The intuition here is that if a post x' is similar to x its responses might be appropriate for x. It however often fails, especially when a response to x' addresses parts of x not contained by x, which fortunately can be alleviated when combined with other measures.

5.2 Learning to Rank with Labeled Data

With all the matching features, we can learn a ranking model with the labeled (post, response) pairs, e.g., through off-the-shelf ranking algorithms. From the labeled data, we can extract triples (x, y^+, y^-) to ensure that $\mathsf{score}(x, y^+) > \mathsf{score}(x, y^-)$. Apparently y^+ can be selected from labeled positive response of x, while y^- can be sampled either from labeled negative negative or randomly selected ones. Since the manually labeled negative instances are top-ranked candidates according to some individual retrieval model (see Section 5.1) and therefore generally yield slightly better results.

The matching features are mostly constructed by combining the individual matching models, for example the following two

- $\Phi_7(x, y)$: this feature measures the length of the longest common string in the post and the response;
- $\Phi_{12}(x, y)$: this feature considers both semantic matching score between query post x and candidate response $\mathbf{y}$, as well as the similarity between x and y's *original* post x':

$$\Phi_{12}(x, y) = \mathsf{SemMatch}(x, y)\mathsf{sim}_{PP}(x, x').$$

In addition to the matching features, we also have simple features describing responses only, such as the length of it.

6 Experimental Evaluation

We perform experiments on the proposed dataset to test our retrieval-based model as an algorithm for automatically generating response.

6.1 Performance of Models

We evaluate the retrieved models based on the following two metrics:

MAP This one measures the mean average precision (MAP)(Manning et al., 2008) associated with the ranked list on $\mathcal{C}_{\mathbf{x}}^{(reduced)}$.

P@1 This one simply measures the precision of the top one response in the ranked list:

$$P@1 = \frac{\#\text{good top-1 responses}}{\#\text{posts}}$$

We perform a 5-fold cross-validation on the 422 labeled posts, with the results reported in Table 1. As it shows, the semantic matching helps slightly improve the overall performance on P@1.

Model	MAP	P@1
P2R	0.565	0.489
P2R + P2P	0.621	0.567
P2R + MATCH	0.575	0.513
P2R + P2P + MATCH	0.621	0.574

Table 3: Comparison of different choices of features, where P2R stands for the features based on post-response similarity, P2P stands for the features based on post-post similarity, and MATCH stands for the semantic match feature.

To mimic a more realistic scenario on automatic response model on Sina Weibo, we allow the system to choose which post to respond to. Here we simply set the response algorithm to respond only when the highest score of the candidate response passes a certain threshold. Our experiments show that when we choose to respond only to 50% of the posts, the P@1 increases to 0.76, while if the system only respond to 25% of the posts, P@1 keeps increasing to 81%.

6.2 Case Study

Although our preliminary retrieval model does not consider more complicated syntax, it is still able to capture some useful coupling structure between the appropriate (post, response) pairs, as well as the similar (post, post) pairs.

Query Post:	创新工场三年庆, 在 我们的智慧树会议室
	Today is the 3-year anniversary of Innovation Works. We are in the meeting rooms named Tree of Wisdom
Response:	嗯, 中午去的, 新环境不错, 很宽敞
	Yeah, I came in the noon. Nice environment, quite spacious

Figure 6: An actual instance (the original Chinese text and its English translation) of response returned by our retrieval-based system.

Case study shows that our retrieval is fairly effective at capturing the semantic relevance (Section 6.2.1), but relative weak on modeling the logic consistency (Section 6.2.2). Also it is clear that the semantic matching feature (described in Section 5.1) helps find matched responses that do not share any words with the post (Section 6.2.3).

6.2.1 On Semantic Relevance

The features employed in our retrieval model are mostly vector-space based, which are fairly good at capturing the semantic relevance, as illustrated by Example 1 & 2.

EXAMPLE 1:

P:	*It is a small town on an Spanish with 500 population, and guess what, they even have a casino!*
R:	*If you travel to Spain, you need to spend some time there.*

EXAMPLE 2:

P:	*One quote from Benjamin Franklin: "We are all born ignorant, but one must work hard to remain stupid."*
R:	*Benjamin Franklin is a wise man, and one of the founding fathers of USA.*

However our retrieval model also makes bad choice, especially when either the query post or the response is long, as shown in Example 3. Here the response is picked up because 1) the correspondence between the word "IT" in the post and the word "mobile phone" in the candidate, and 2) the Chinese word for "lay off" in the post and the word for "outdated" in the response are the same.

EXAMPLE 3:

P:	*As to the laying-off, I haven't heard anything about it. "Elimination of the least competent" is kind-off conventional in IT, but the ratio is actually quite small.*
R:	*Please don't speak that way, otherwise you can get outdated. Mobile phones are very expensive when they were just out, but now they are fairly cheap. Look forward, or you will be outdated.*

The entity association is only partially addressed with features like post-response cosine similarity, treating entity name just as a word, which is apparently not enough for preventing the following type

of mistakes (see Example 4 & 5) when the post and response match well on other parts

EXAMPLE 4:

P:	*Professor Wang will give a curse on natural language processing, starting next semester.*
R:	*Jealous.. I wish I can attend Prof. Li's course too some time in the future.*

EXAMPLE 5:

P:	*The fine China from Exhibition at the National Palace Museum in Taipei*
R:	*This drawing looks so nice. National Palace Museum in Taipei is full of national treasures*

6.2.2 On Logic Consistency

Our current model does not explicitly maintain the logic consistency between the response and the post, since Logic consistency requires a deeper analysis of the text, and therefore hard to capture with just a vector space model. Below are two examples which are semantically relevant, and correct with respect to speech act, but logically inappropriate.

EXAMPLE 1:

P:	*I checked. Wang Fengyi is not my great grandfather, although they've done similar deeds and both were called "Wang the Well-doer".*
R:	*wow, Wang Fengyi is your great grand-father*

EXAMPLE 2:

P:	*We are looking for summer interns. We provide books and lunch. If you are in Wu Han and interested, drop us an email. Sorry we don't take any students outside Wu Han.*
R:	*Are you looking for summer intern?*

6.2.3 The Effect of Semantic Matching

The experiments also show that we may find interesting and appropriate responses that have no common words as the post, as shown in the example below. Our bi-linear semantic matching model however performs relatively poorly on long posts, where the topics of the sentence cannot be well captured by the sum of the latent vectors associated with each word.

P:	*Eight England players stand in the penalty area.*
R1:	*What a classic match*
R2:	*Haha, it is still 0:0, no goal so far*

7 Summary

In this paper we propose a retrieval-based response model for short-text based conversation, to leverage the massive instances collected from social media. For research in similar directions, we create a dataset based on the posts and comments from Sina Weibo. Our preliminary experiments show that our retrieval-based response model, when combined with a large candidate set, can achieve fairly good performance. This dataset will be valuable for both training and testing automatic response models for short texts.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2).

Rollo Carpenter. 1997. Cleverbot.

Sina Jafarpour and Christopher J. C. Burges. 2010. Filter, rank, and transfer the knowledge: Learning to chat.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA. ACM.

Anton Leuski and David R. Traum. 2011. Npceditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine*, 32(2):42–56.

Diane Litman, Satinder Singh, Michael Kearns, and Marilyn Walker. 2000. Njfun: a reinforcement learning spoken dialogue system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems - Volume 3*, ANLP/NAACL-ConvSyst '00, pages 17–20, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Teruhisa Misu, Kallirroi Georgila, Anton Leuski, and David Traum. 2012. Reinforcement learning of question-answering dialogue policies for virtual museum guides. In *Proceedings of the 13th Annual Meeting*

of the Special Interest Group on Discourse and Dialogue, SIGDIAL '12, pages 84–93.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 583–593, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl. Eng. Rev.*, pages 97–126.

Ellen M Voorhees. 2002. The philosophy of information retrieval evaluation. In *Evaluation of cross-language information retrieval systems*, pages 355–370. Springer.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, 21(2):393–422.

Wei Wu, Zhengdong Lu, and Hang Li. 2013. Learning bilinear model for matching queries and documents. *Journal of Machine Learning Research (2013 to appear)*.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. SIGHAN '03.

Discourse Level Explanatory Relation Extraction from Product Reviews Using First-order Logic

Qi Zhang, Jin Qian, Huan Chen, Jihua Kang, Xuanjing Huang
School of Computer Science
Fudan University
Shanghai, P.R. China
{qz, 12110240030, 12210240054, 12210240059, xjhuang}@fudan.edu.cn

Abstract

Explanatory sentences are employed to clarify reasons, details, facts, and so on. High quality online product reviews usually include not only positive or negative opinions, but also a variety of explanations of why these opinions were given. These explanations can help readers get easily comprehensible information of the discussed products and aspects. Moreover, explanatory relations can also benefit sentiment analysis applications. In this work, we focus on the task of identifying subjective text segments and extracting their corresponding explanations from product reviews in discourse level. We propose a novel joint extraction method using first-order logic to model rich linguistic features and long distance constraints. Experimental results demonstrate the effectiveness of the proposed method.

1 Introduction

Through analyzing product reviews with high helpfulness ratings assigned by readers, we find that a large number of explanatory sentences are used to clarify the causes, details, or consequences of opinions. According to the statistic based on the dataset we crawled from a popular product review website, more than 56.1% opinion expressions are further explained by other sentences. Since most consumers are not experts, these explanations would bring lots of helpful and easy comprehension information for them. Suggestions about writing a product review also advise authors to include not only whether they like or dislike a product, but also why.[1]

For example, let us consider the following snippets extracted from online reviews:

> **Example 1**: *TVs with lower refresh rates may suffer from motion blur. If you're watching a fast-paced football game, for example, you may notice a bit of blurring as the players run around the field.*

> **Example 2**: *The LED screen is highly reflective. The reflection of my own face makes it very hard to see the subject I am trying to shoot.*

The first sentence of example 1 expresses negative opinion about refresh rate, which is one of the most important attributes of TV. The second sentence describes the consequence of it through an example. In example 2, detail descriptions are used to explain the reflection problem of the camera screen.

Although, explanations provide valuable information, to the best of our knowledge, there is no existing work that deals with explanation extraction for opinions in discourse level. We think that if explanatory relations can be automatically identified from reviews, sentiment analysis applications may benefit from it. Existing opinion mining approaches mainly focus on subjective text. They try to determine the subjectivity and polarity of fragments of documents (e.g. a paragraph, a sentence, a phrase and a word) (Pang et al., 2002; Riloff et al., 2003; Takamura et al., 2005; Mihalcea et al., 2007; Dasgupta and Ng, ; Hassan and Radev, 2010; Meng et al., 2012; Dragut et al., 2012). Fine-grained methods were also introduced to extract opinion holder, opinion expression, opinion target, and other opinion elements (Kobayashi et al., 2007; Wu et al.,

[1] http://www.reviewpips.com/
http://www.amazon.com/gp/community-help/customer-reviews-guidelines

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 946–957,
Seattle, Washington, USA, 18-21 October 2013.

2011; Xu et al., 2013; Yang and Cardie, 2013). Major research directions and challenges of sentiment analysis can also be found in surveys (Pang and Lee, 2008; Liu, 2012).

In this work, we aim to identify subjective text segments and extract their corresponding explanations from product reviews in discourse level. We propose to use Markov Logic Networks (MLN) (Richardson and Domingos, 2006) to learn the joint model for subjective classification and explanatory relation extraction. MLN has been applied in several natural language processing tasks (Singla and Domingos, 2006; Poon and Domingos, 2008; Yoshikawa et al., 2009; Andrzejewski et al., 2011; Song et al., 2012) and demonstrated its advantages. It can easily incorporate rich linguistic features and global constraints by designing various logic formulas, which can also be viewed as templates or rules. Logic formulas are combined in a probabilistic framework to model soft constraints. Hence, the proposed approach can benefit a lot from this framework.

To evaluate the proposed method, we crawled a large number of product reviews and constructed a labeled corpus through Amazon's Mechanical Turk. Two tasks were deployed for labeling the corpus. We compared the proposed method with state-of-the-art methods on the dataset. Experimental results demonstrate that the proposed approach can achieve better performance than state-of-the-art methods.

The remaining part of this paper is organized as follows: In Section 2, we define the problem and give some examples to show the challenges of this task. Section 3 describes the proposed MLN based method. Dataset construction, experimental results and analyses are given in Section 4. In Section 5, we present the related work and Section 6 concludes the paper.

2 Problem Statement

Motivated by the argument structure of discourse relations used in Penn Discourse Treebank (Rashmi Prasad and Webber, 2008), in this work, we adopt the clause unit-based definition. It means that clauses are treated as the basic units of opinion expressions and explanations. Let $d = \{c_1, c_2, ...c_n\}$ be the clauses of document d. Directed graph $G = (V, E)$ is used to represent the subjectivity of clauses and explanatory relationships between them. In the graph, vertices represent clauses, whose categories are specified by the vertex attributes. Directed edges describe the explanatory relationships between them, of which the heads are explanatory clauses. If clause c_a describes a set of facts which clarify the causes, context, situation, or consequences of another clause c_b, $c_a \longrightarrow c_b$ is used to indicate that clause c_a *explains* c_b.

Adopting clause unit-based definition is based on the following reasons: 1) clause is normally considered as the smallest grammatical unit which can express a complete proposition (Kroeger, 2005); 2) from analyzing online reviews, we observe that a clause can express a complete opinion about one aspect in most of cases; 3) in Penn Discourse Treebank, the basic unit of discourse relations (with a few exceptions) is also taken to be a clause (Rashmi Prasad and Webber, 2008).

Figure 1(a) illustrates a sample document. Figure 1(b) is the corresponding output of the given document. In the graph, vertices whose color are black stand for subjective clauses. The other clauses are represented by white vertices. Edges describe the explanatory relationships between them, of which the heads are explanatory clauses.

Although the explanatory relation extraction task has been studied from the view of linguistic and discourse representation by existing works (Carston, 1993; Lascarides and Asher, 1993), the automatic extraction task is still an open question. Consider the following examples extracting from online reviews:

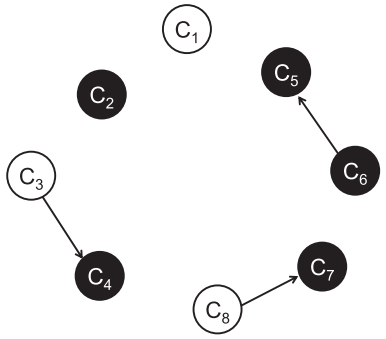
Example 3: *It takes great pictures. Color renditions, skin tones, exposure levels are all first rate.* From the example, we can observe that the second sentence explains the first one. However, the second sentence itself also expresses opinion on various opinion targets. In other words, both subjective and objective sentences can be used as explanations.

Example 4: *When we called their service center they made us wait for them the whole day and no one turned up. This level of service is simply not acceptable.* The first sentence in example 4 explains the second one. Hence, the feature of relative location between two sentences does not always work well in all cases.

Example 5: *This backpack is great! its very big*

(c1) I have both the Panasonic LX3 and the Canon S90. ***(c2)*** Both cameras are quite different but truly excellent. ***(c3)*** The S90 is a true pocket camera. ***(c4)*** It is very compact. ***(c5)*** The build quality is also top notch. ***(c6)*** It feels solid and it is easy to grip. ***(c7)*** It is so small and convenient, ***(c8)*** you will find that you will always carry it with you.

(a) Example Review



(b) Directed Graph Representation

Figure 1: Directed graph representation of a sample document.

and fits more than enough stuff. Many sentences, which express explanatory relation, do not contain any connectives (e.g. "*because*", "*the reason is*", and so on). Lin et al.(2009) generalized four challenges (include ambiguity, inference, context, and world knowledge) to automated implicit discourse relation recognition. In this task, we also need to address those challenges.

From the these examples, we can observe that extracting explanatory relations from product reviews is a challenging task. Both linguistic and global constraints should be carefully studied.

3 The Proposed Approach

In this section, we present our method for jointly classifying the subjectivity of text segments and extracting explanatory relations. Firstly, we briefly describe the framework of Markov Logic Networks. Then, we introduce the clause extraction method based on the definition described in the Section 2. Finally, we present the first-order logic formulas including local formulas and global formulas used for joint modeling in this work.

3.1 Markov Logic Networks

A MLN consists of a set of logic formulas that describe first-order knowledge base. Each formula consists of a set of first-order predicates, logical connectors and variables. Different with first-order logic, these hard logic formulas are softened and can be violated with some penalty (the weight of formula) in MLN.

We use $\mathcal{M}$ to represent a MLN and $\{(\phi_i, w_i)\}$ to represent formula ϕ_i and its weight w_i. These weighted formulas define a probability distribution over sets of possible worlds. Let y denote a possible world, the $p(y)$ is defined as follows (Richardson and Domingos, 2006):

$$p(y) = \frac{1}{Z} exp\left(\sum_{(\phi_i, w_i)\in\mathcal{M}} w_i \sum_{c\in C^{n_{\phi_i}}} f_c^{\phi_i}(y)\right),$$

where each c is a binding of free variable in ϕ_i to constraints; $f_c^{\phi_i}(y)$ is a binary feature function that returns 1 if the true value is obtained in the ground formula we get by replacing the free variables in ϕ_i with the constants in c under the given possible world y, and 0 otherwise; $C^{n_{\phi_i}}$ is all possible bindings of variables to constants, and Z is a normalization constant.

Many methods have been proposed to learn the weights of MLN using both generative and discriminative approaches (Richardson and Domingos, 2006; Singla and Domingos, 2006). There are also several MLN learning packages available online such as thebeast[2], Tuffy[3], PyMLNs[4], Alchemy[5], and so on.

[2]http://code.google.com/p/thebeast
[3]http://hazy.cs.wisc.edu/hazy/tuffy/
[4]http://www9-old.in.tum.de/people/jain/mlns/
[5]http://alchemy.cs.washington.edu/

Describing the attributes of words	
$subjLexicon(w)$	The word w belongs to the subjective lexicon (Baccianella et al., 2010).
$relationLexicon(w)$	The word w belongs to the lexicon of explanation relation connectives (Pitler and Nenkova, 2009).
Describing the attributes of the clause c_i	
$word(i, w)$	The clause c_i has word w.
$firstWord(i, w)$	The first word of clause c_i is word w.
$pos(i, w, t)$	The POS tag of word w is t in clause c_i.
$dep(i, h, m)$	Word m and h are governor and dependent of a dependency relation in clause c_i.
Describing the attributes of relations between clause c_i and clause c_j	
$clauseDistance(i, j, m)$	Distance between clause c_i and clause c_j in clauses is m.
$sentenceDistance(i, j, n)$	Distance between clause c_i and clause c_j in sentences is n.

Table 1: Descriptions of observed predicates.

3.2 Clause Identification

We model the clause boundary identification problem through sequence labeling and use Conditional Random Fields (CRFs) to identify clause boundaries. Words and part-of-speech (POS) tags are used as feature sets. Since we do not allow embedded segments, the performance of our method is promising, which achieves the F1 score of 92.8%. The result is comparable with the best results obtained during the CoNLL-2001 campaign (Tjong et al., 2001).

3.3 Formulas

In this work, we propose to use predicate $subj(i)$ to indicate that the ith clause is subjective and $explain(i, j)$ to indicate that the jth clause explains the ith clause. Both $subj$ and $explain$ are hidden predicates and jointly modeled by MLN. We use local and global formulas to model rich linguistic features and long distance constraints.

3.3.1 Local Formulas

The local formulas relate one or more observed predicates to exactly one hidden predicate. In this work, we define a list of observed predicates to describe the properties of individual clauses and attributes of relations between two clauses. The observed predicates and descriptions are shown in Table 1. The observed predicates can be categorized into 3 groups: words, clauses, and relations between clauses. We use two lexicons to capture background knowledge of words. Lexical, part-of-speech tag, and dependency relation are used to describe a single clause. We also propose two predicates to model distance between clauses.

Table 2 lists the local formulas used in this work. The "+" notation in the formulas indicates that each constant of the logic variable should be weighted separately. For subjective classification and relation extraction, we construct a number of formulas respectively.

For subjective classification, the first two formulas model the influence of lexical and POS tag. It is similar as the bag-of-words model, which is a simplifying representation and has been successfully used for various natural language processing tasks. Since words which provide positive or negative opinions may provide important information for subjectivity classification, we combine predicates of words and lexicon of opinion words. Bigrams are also proved to be useful for textual classification in several NLP tasks. Hence, we also combine predicates about individual word and POS tag to capture this kind of information. Word-level relations are explicitly presented at the dependency trees, we

Formulas for subjective classification
word(i,w+) $\Rightarrow$ subj(i)
pos(i,w+,t+) $\Rightarrow$ subj(i)
word(i,w+) $\wedge$ subjLexicon(w) $\Rightarrow$ subj(i)
pos(i,w+,t+) $\wedge$ subjLexicon(w) $\Rightarrow$ subj(i)
word(i,w_1+) $\wedge$ word(i,w_2+) $\Rightarrow$ subj(i)
pos(i,w_1+,t+) $\wedge$ pos(i,w_2+,t+) $\Rightarrow$ subj(i)
word(i,w_1+) $\wedge$ word(i,w_2+) $\wedge$ subjLexicon(w_1) $\Rightarrow$ subj(i)
word(i,w_1+) $\wedge$ word(i,w_2+) $\wedge$ subjLexicon(w_2) $\Rightarrow$ subj(i)
dep(i,w_1+,w_2+) $\Rightarrow$ subj(i)
dep(i,w_1+,w_2+) $\wedge$ subjLexicon(w_1) $\Rightarrow$ subj(i)
dep(i,w_1+,w_2+) $\wedge$ subjLexicon(w_2) $\Rightarrow$ subj(i)
Formulas for explanatory relation extraction
word(i,w_1+) $\wedge$ word(j,w_2+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
pos(i,w_1+,t+) $\wedge$ pos(j,w_2+,t+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
dep(i,h_1+,m_1+) $\wedge$ dep(j,h_2+,m_2+)$\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
word(i,w_1+) $\wedge$ word(j,w_2+) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
pos(i,w_1+,t+) $\wedge$ pos(j,w_2+,t+) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
dep(i,h_1+,m_1+) $\wedge$ dep(j,h_2+,m_2+) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
word(i,w_1+) $\wedge$ word(j,w_2+) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
pos(i,w_1+,t+) $\wedge$ pos(j,w_2+,t+) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
dep(i,h_1+,m_1+) $\wedge$ dep(j,h_2+,m_2+) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
word(i,w_1+) $\wedge$ word(j,w_2+) $\wedge$ firstWord(j,w+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
pos(i,w_1+,t+) $\wedge$ pos(j,w_2+,t+) $\wedge$ firstWord(j,w+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
dep(i,h_1+,m_1+) $\wedge$ dep(j,h_2+,m_2+) $\wedge$ firstWord(j,w+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
word(i,w_1+) $\wedge$ word(j,w_2+) $\wedge$ subjLexicon(w_1) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
pos(i,w_1+,t+) $\wedge$ pos(j,w_2+,t+) $\wedge$ subjLexicon(w_1) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
dep(i,h_1+,m_1+) $\wedge$ dep(j,h_2+,m_2+) $\wedge$ subjLexicon(m_1) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ pos(j,w,t+) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ pos(j,w,t+) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ word(i,w_1+) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ word(i,w_1+) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ word(j,w_1+) $\wedge$ clauseDistance(i,j,m+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)
firstWord(j,w+) $\wedge$ relationLexicon(w) $\wedge$ word(j,w_1+) $\wedge$ sentenceDistance(i,j,n+) $\wedge$ j$\neq$i $\Rightarrow$ explain(i,j)

Table 2: Descriptions of local formulas.

also construct local formulas based on predicates extracted from dependency trees of clauses.

For explanatory relation extraction, we firstly use formulas to capture lexical and syntactic information from both of the clauses. Since distances between clauses are helpful in determining the relation, we incorporate two kinds of distance features with lexical and syntactic predicates. Connective words such as *for example*, *since*, explicitly signal the presence of the explanation relation. Although some connective words are ambiguous in terms of relation they mark (Pitler and Nenkova, 2009), they may still be useful for explanation relation extraction. Hence, we construct local formulas with relation lexicon and other predicates.

3.3.2 Global Formulas

Local formulas are designed to deal with subjective classification of a single clause or relation determination of a single pair of clauses. Global formulas are designed to handle global constraints of multiple clauses. From the definition of explanatory relation and corpus statistics, we observe the following properties:

Property 1: One clause can only serve as the explanation of one subjective clause.

Property 2: Explanatory clauses occur immediately before or after their corresponding subjective clauses.

Property 3: The positions of explanatory clauses are consecutive. In other words, if clause c_k and c_{k+2} explain clause c_j, the clause c_{k+1} would also be explanatory clause of c_j.

For property 1, we use the following global formula to make sure that one clause only explains at most one another clause.

$$explain(i,j) \Rightarrow \neg explain(k,j) \quad \forall k \neq i,j \quad (1)$$

Based on the property 2 and 3, explanatory clauses are consecutive and immediately before or after their corresponding subjective clauses. We use the following formulas to guarantee the property:

$$explain(i,i+k) \Rightarrow explain(i,i+m), \quad 1 \leq m \leq k-1 \quad (2)$$

$$explain(i,i-k) \Rightarrow explain(i,i-m), \quad 1 \leq m \leq k-1 \quad (3)$$

Since our aim is to extract explanatory for subjective clauses, we also use the following formulas to make sure that the clauses which are explained are subjective ones.

$$explain(i,j) \Rightarrow subj(i) \quad (4)$$

4 Experiments

4.1 Data Set

We crawled a number of reviews about digital cameras from Buzzillions[6], which is a product review site and contains more than 16 million reviews. We randomly select 100 reviews whose usefulness ratings are 5 on a 5-point scale. They contain 1137 sentences, which are composed by 1665 clauses. Amazon's Mechanical Turk is used to deploy two tasks for labeling the corpus. 694 clauses are labeled subjective and 478 clauses explain other ones. More than 56.1% opinion expressions are explained by their corresponding explanatory sentences.

The two projects we deployed on Amazon's Mechanical Turk are: 1) Determine whether a clause contains opinion expressions or not; 2) Determine whether a clause clarifies causes, reasons, or consequences of another given clause. In order to control the labeling quality, we configured parameters of the project to make sure that all the tasks should be judged by at least 20 annotators. Most of the annotators can complete a task within 25 seconds. Figure 2 shows the screenshots of the two projects.

Over all, 127 workers participated in the project. About 72% of them submitted more than 5 tasks. Although we listed several examples on the project descriptions, different people may have their own understanding and criteria for those tasks. In order to measure the quality of the labeling task, we use perplexity to evaluate each task. If the perplexity of a task is below 0.51, which means that more than 80% of the workers submitted the same decision, the result of the task will be used as training or testing data. From the statistic of the corpus, we observe that only 6.2% of the clauses' subjectiveness and 15.6% of explanation relations can not be certainly decided. For the first project, we treated those clauses as objective one. And, those clause pairs in the second project were not considered as explanation relations.

[6] www.buzzillions.com

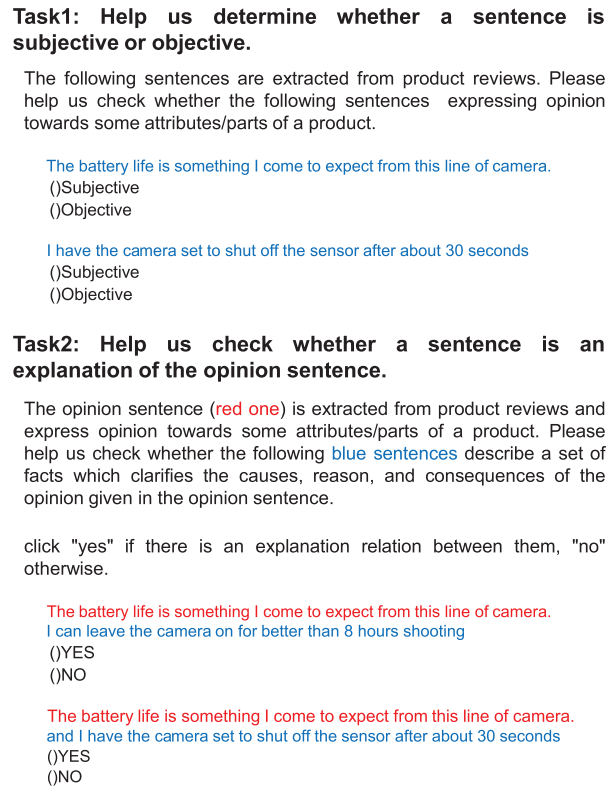


Figure 2: Screenshots of the two tasks on Amazon Mechanical Turk.

4.2 Experiments Configurations

Stanford parser (Klein and Manning, 2003) is used for extracting features from dependency parse trees. For resolving Markov logic network, we use the toolkit *thebeast* [7]. The detailed setting of thebeast engine is as follows: The inference algorithm is the MAP inference with a cutting plane approach. For parameter learning, the weights for formulas are updated by an online learning algorithm with MIRA update rule. All the initial weights are set to zeros. The number of iterations is set to 10 epochs.

Evaluation metrics used for subjectivity classification and relation extraction throughout the experiments include: Precision, Recall, and F1-score. We randomly select 80% reviews as training set and the others as testing set.

Since the dataset is newly created for this task, to compare the performance of the proposed method to other models, we also reimplemented several state-of-the-art methods for comparison.

- **CRF-Subj**: We follow the method proposed by Zhao et al. (2008), which regard the subjectivity of all clauses throughout a paragraph as a sequential flow of sentiments and use CRFs to model it. The feature sets are similar as the local formulas for MLN including words, POS tags, dependency relations, and opinion lexicon.

- **RAE-Subj**: Socher et al. (2011) proposed to use recursive autoencoders for sentence-level predication of sentiment label distributions. To compare with it, we also reimplement their method without any hand designed lexicon.

- **PDTB-Rel**: For discourse relation extraction, we use "PDTB-Styled End-to-End Discourse Parser" (Lin et al., 2010) to extract discourse level relations as baseline. Since it is a general discourse relations identification algorithms, "Cause", "Pragmatic Cause", "Instantiation", and "Restatement" relation types are treated as explanatory relation in this work.

- **SVM-Rel**: We also use LibSVM (Chang and Lin, 2011) to classify the relations between clauses. Following the configurations reported by Feng and Hirst (2012), we use linear kernel and probability estimation to model it.

4.3 Results

Table 3 shows the comparisons of the proposed method with the state-of-the-art systems on subjectivity classification and explanatory relation extraction. From the results, we can observe that recursive autoencoders based subjectivity classification method achieves slightly better performance than our method and conditional random fields based method. The performances of the proposed method are similar as CRFs'. We think that the main reason is that only lexical features are used in MLN models for subjective classification. However, conditional random fields consider not only lexical information but also inference of the contexts of sentences. RAE method learns vector space representations for multi-word phrases and uses compositional semantics to understand sentiment.

[7] http://code.google.com/p/thebeast

Methods	Subjective Classification		
	P	**R**	**F_1**
CRF-Subj	83.5%	76.9%	80.1%
RAE-Subj	**85.3**%	79.1%	**82.1**%
MLN	79.2%	**80.6**%	79.9%
Methods	Relation Extraction		
	P	**R**	**F_1**
RAE-Subj + PDTB-Rel	28.5%	38.6%	32.8%
RAE-Subj + SVM-Rel	32.4%	**89.7**%	47.6%
MLN	**56.2**%	72.9%	**63.5**%

Table 3: Performance comparisons between the proposed method and state-of-the-art methods. "MLN" represents the method proposed in this work.

For evaluating the performance of relation extraction, we combine the results of RAE with PDTB-Rel and SVM-Rel. For all the subjective clauses identified by RAE, PDTB-Rel and SVM-Rel are used to extract corresponding explanatory clauses. The results are shown in the last three rows in the Table 3. From the results, we can observe that the proposed joint model achieves best F1 score and precision among all methods. Although the proposed method achieve slightly worse result in processing subjectivity classification. We think that the error propagation is the main reason for worse results of cascaded methods. The relative improvement of MLN over SVM-Rel is more than 33.4%.

To show the effectiveness of different observed predicates, we evaluate the performances of the proposed method with different predicate sets. We subtract one observed predicate and its corresponding local formulas from the original sets at a time. The results of both subjectivity classification and relation extraction are shown in Table 4. The first row shows the result of the MLN based method with all observed predicates and local formulas. From the results we can observe that the observed predicates which are not used in the local formulas for subjectivity classification also impact the performance of subjectivity classification. We think that the performance is effected by the global formulas, which combine the procedure of subjectivity classification and relation extraction. Among all predicates, we observe that words and dependency relations play the most important roles. Without word predicate, the F1 score of subjectivity classification and relation extraction significantly drop to 51.2% and 42.9% respectively. For subjectivity classification, subjective lexicon contributes a lot for recall. For relation extraction, the impacts of clause distance and sentence distance are not as significant as the other features.

5 Related Work

Our work relates to three research areas: sentiment analysis/opinion mining, discourse-level relation extraction, and Markov logic networks. Along with the increasing requirement, subjectivity classification has recently received considerable attention from both the industry and researchers. A variety of approaches and methods have been proposed for this task from different aspects. Among them, a number of approaches focus on classifying sentiments of text in different levels (e.g. words (Kim and Hovy, 2004), phrases (Wilson et al., 2005), sentences (Zhao et al., 2008), documents (Pang et al., 2002) and so on.), and detecting the overall polarity of them.

Another research direction tries to convert the sentiment analysis task into entity identification and relation extraction. Hu and Liu (2004) proposed to use a set of methods to produce feature-based summary of a large number of customer reviews. Kobayashi et al. (2007) assumed that evaluative opinions could be structured as a frame which is composed by opinion holder, subject, aspect, and evaluation. They converted the task to two kinds of relation extraction tasks and proposed a machine learning-based method which used both contextual and statistical clues.

Analysis of some special types of sentences were also introduced in recent years. Jindal and Liu (2006) studied the problem of identifying comparative sentences. They analyzed different types of comparative sentences and proposed learning approaches to identify them. Conditional sentences were studied by Narayanan et al (2009). They analyzed the conditional sentences in both linguistic and computitional perspectives and used learning

	Subjective Classification			Relation Extraction		
	P	R	F1	P	R	F1
MLN	79.2%	80.6%	79.9%	56.2%	72.9%	63.5%
$-subjLexicon(w)$	76.6%	70.4%	73.4 %	52.3%	68.6%	59.4%
$-relationLexicon(w)$	78.2%	79.4%	78.8%	53.6%	70.8%	61.0%
$-word(i,w)$	52.8%	49.6%	51.2 %	36.4%	52.1%	42.9%
$-firstWord(i,w)$	76.3%	80.1%	78.2%	56.9%	69.8%	62.7%
$-pos(i,w,t)$	72.6%	76.8%	74.6 %	52.4%	60.2%	56.0%
$-dep(i,h,m)$	57.6%	70.6%	63.4%	41.2%	56.8%	47.8%
$-clauseDistance(i,j,m)$	78.9%	80.2%	79.5%	52.6%	70.6%	60.3%
$-sentenceDistance(i,j,n)$	78.6%	80.3%	79.4%	52.4%	70.8%	60.2%

Table 4: Performance comparisons of different observed predicates

method to do it. They followed the *feature-based sentiment analysis* model (Hu and Liu, 2004), which also use flat frames to represent evaluations.

Since the cross sentences relations are considered in this work, the discourse-level relation extraction methods are also related to ours. Marcu and Echihabi (2002) proposed to use an unsupervised approach to recognizing discourse relations. Lin et al.(2009) analyzed the impacts of features extracted from contextual information, constituent parse trees, dependency parse trees, and word pairs. Asher et al.(2009) studied discourse segments containing opinion expressions from the perspective of linguistics. Chen et al. (2010) introduced a multi-label model to detect emotion causes. They developed two sets of linguistic features for this task base on linguistic cues. Zirn et al. (2011) proposed to use MLN framework to capture the context information in analysing (sub-)sentences.

The most similar work to ours was proposed by Somasundaran et al.(2009). They proposed to use iterative classification algorithm to capture discourse-level associations. However different to us, they focused on pairwise relationships between opinion expressions. In this paper, we used MLN framework to capture another different discourse-level relation, which exists between subject clauses or subject clause and objective clause.

Richardson and Domingos (2006) proposed Markov Logic Networks, which combines first-order logic and probabilistic graphical models. In recent years, MLN has been adopted for several natural language processing tasks and achieved a certain level of success (Singla and Domingos, 2006; Riedel and Meza-Ruiz, 2008; Yoshikawa et al., 2009; Andrzejewski et al., 2011; Jiang et al., 2012; Huang et al., 2012). Singla and Domingos (2006) modeled the entity resolution problem with MLN. They demonstrated the capability of MLN to seamlessly combine a number of previous approaches. Poon and Domingos (2008) proposed to use MLN for joint unsupervised coreference resolution. Yoshikawa et al. (2009) proposed to use Markov logic to incorporate both local features and global constraints that hold between temporal relations. Andrzejewski et al. (2011) introduced a framework for incorporating general domain knowledge, which is represented by First-Order Logic (FOL) rules, into LDA inference to produce topics shaped by both the data and the rules.

6 Conclusions

In this paper, we propose to use Markov logic networks to identify subjective text segments and extract their corresponding explanations in discourse level. We use MLN to jointly model subjectivity classification and explanatory relation extraction. Rich linguistic features and global constraints are incorporated by various logic formulas and global formulas. To evaluate the proposed method, we collected a large number of product reviews and

constructed a labeled corpus through Amazon's Mechanical Turk. Experimental results demonstrate that the proposed approach achieve better performance than state-of-the-art methods.

7 Acknowledgement

The authors wish to thank the anonymous reviewers for their helpful comments and Kang Han for preparing the corpus. This work was partially funded by National Natural Science Foundation of China (61003092, 61073069), Key Projects in the National Science & Technology Pillar Program(2012BAH18B01), National Major Science and Technology Special Project of China (2014ZX03006005), Shanghai Municipal Science and Technology Commission (12511504502) and "Chen Guang" project supported by Shanghai Municipal Education Commission and Shanghai Education Development Foundation(11CG05).

References

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1171–1177. AAAI Press.

Nicholas Asher, Farah Benamara, and Yannick Mathieu. 2009. Appraisal of opinion expressions in discourse. *Lingvisticæ Investigationes*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

R. Carston. 1993. Conjunction, explanation and relevance. *Lingua 90*, pages 27–48.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *Proceedings ofColing 2010*.

Sajib Dasgupta and Vincent Ng. Mine the easy, classify the hard: A semi-supervised approach to automatic sentiment classification. In *Proceedings of ACL-IJCNLP 2009*.

Eduard Dragut, Hong Wang, Clement Yu, Prasad Sistla, and Weiyi Meng. 2012. Polarity consistency checking for sentiment dictionaries. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 997–1005, Jeju Island, Korea, July. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 60–68, Jeju Island, Korea, July. Association for Computational Linguistics.

Ahmed Hassan and Dragomir R. Radev. 2010. Identifying text polarity using random walks. In *Proceedings of ACL 2010*, Uppsala, Sweden, July.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD 2004*.

Minlie Huang, Xing Shi, Feng Jin, and Xiaoyan Zhu. 2012. Using first-order logic to compress sentences. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Shangpu Jiang, D. Lowd, and Dejing Dou. 2012. Learning to refine an automatically extracted knowledge base using markov logic. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 912–917.

Nitin Jindal and Bing Liu. 2006. Identifying comparative sentences in text documents. In *Proceedings of SIGIR 2006*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING 2004*.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Proceedings of NIPS 2003*.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of EMNLP-CoNLL 2007*.

Paul Kroeger. 2005. *Analyzing Grammar: An Introduction*. Cambridge.

Alex Lascarides and Nicholas Asher. 1993. Temporal interpretation, discourse relations, and common sense entailment. *Linguistics and Philosophy*, 16(5):437–493.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of EMNLP 2009*.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A pdtb-styled end-to-end discourse parser. *CoRR*, abs/1011.0835.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 368–375.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 572–581, Jeju Island, Korea, July. Association for Computational Linguistics.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of ACL 2007*.

Ramanathan Narayanan, Bing Liu, and Alok Choudhary. 2009. Sentiment analysis of conditional sentences. In *Proceedings of EMNLP 2009*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135, January.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP 2002*.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009*.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 650–659, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alan Lee Eleni Miltsakaki Livio Robaldo Aravind Joshi Rashmi Prasad, Nikhil Dinesh and Bonnie Webber. 2008. The penn discourse treebank 2.0. In Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of L-REC'08*, Marrakech, Morocco, may. http://www.lrec-conf.org/proceedings/lrec2008/.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Sebastian Riedel and Ivan Meza-Ruiz. 2008. Collective semantic role labelling with markov logic. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 193–197, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of HLT-NAACL 2003*.

P. Singla and P. Domingos. 2006. Entity resolution with markov logic. In *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 572–582.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Swapna Somasundaran, Galileo Namata, Lise Getoor, and Janyce Wiebe. 2009. Opinion graphs for polarity and discourse classification. In *Proceedings of TextGraphs-4*.

Yang Song, Jing Jiang, Wayne Xin Zhao, Sujian Li, and Houfeng Wang. 2012. Joint learning for coreference resolution with markov logic. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1245–1254, Jeju Island, Korea, July. Association for Computational Linguistics.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of ACL 2005*.

Erik F. Tjong, Kim Sang, and Hervé Déjean. 2001. Introduction to the conll-2001 shared task: clause identification. In *Proceedings of the 2001 workshop on Computational Natural Language Learning - Volume 7*, ConLL '01, Stroudsburg, PA, USA. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP 2005*.

Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. 2011. Structural opinion mining for graph-based sentiment representation. In *Proceedings of EMNLP 2011*.

Liheng Xu, Kang Liu, Siwei Lai, Yubo Chen, and Jun Zhao. 2013. Walk and learn: a two-stage approach for opinion words and opinion targets co-extraction. In *Proceedings of the 22nd international conference on World Wide Web companion*, WWW '13 Companion, pages 95–96, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of ACL 2013*.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 405–413, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jun Zhao, Kang Liu, and Gen Wang. 2008. Adding redundant features for crfs-based sentence sentiment classification. In *Proceedings of EMNLP 2008*.

Cäcilia Zirn, Mathias Niepert, Heiner Stuckenschmidt, and Michael Strube. 2011. Fine-grained sentiment analysis with structural features. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 336–344, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Building Event Threads out of Multiple News Articles

Xavier Tannier
LIMSI-CNRS
Univ. Paris-Sud
Orsay, France
xavier.tannier@limsi.fr

Véronique Moriceau
LIMSI-CNRS
Univ. Paris-Sud
Orsay, France
moriceau@limsi.fr

Abstract

We present an approach for building multi-document event threads from a large corpus of newswire articles. An event thread is basically a succession of events belonging to the same story. It helps the reader to contextualize the information contained in a single article, by navigating backward or forward in the thread from this article. A specific effort is also made on the detection of reactions to a particular event.

In order to build these event threads, we use a cascade of classifiers and other modules, taking advantage of the redundancy of information in the newswire corpus.

We also share interesting comments concerning our manual annotation procedure for building a training and testing set[1].

1 Introduction

In this paper, we explore a new way of dealing with temporal relations between events. Our task is somewhat between multidocument summarization and classification of temporal relations between events. We work with a large collection of English newswire articles, where each article relates an event: the main topic of the article is a specific event, and other older events are mentioned in order to put it into perspective. Thus, we consider that an event is associated with an article and that defining temporal relations between articles is a way to define temporal relations between events.

[1]This work has been partially funded by French National Research Agency (ANR) under project Chronolines (ANR-10-CORD-010). We would like to thank the French News Agency (AFP) for providing us with the corpus.

The task is to build a temporal graph of articles, linked between each other by the following relations:

- *Same event*, when two documents relate the same event, or when a document is an update of another one.
- *Continuation*, when an event is the continuation or the consequence of a previous one.

We also define a subset of *continuation*, called *reaction*, concerning a document relating the reaction of someone to another event.

Some examples of these three classes will be given in Section 3.

These relations can be represented by a directed graph where documents are vertices and relations are edges (as illustrated in all figures of this article). Figure 1 shows an example of such a graph.

Press articles, and especially newswire articles, are characterized by an important redundancy of related events. An important event[2] is likely to be treated by several successive articles, which will give more and more details and update some numbers (mainly, tragedy casualty updates, as shown in Figure 2). On the one hand, this redundancy is an issue since a system must not show duplicate information to the user; on the other hand, we show in this article that it can also be of great help in the process of extracting temporal graphs.

In what follows, we first review some of the related work in Section 2. Section 3 presents the annotation procedure and the resulting annotated corpus

[2]Note that we do not focus intentionally on "important" events. However, the fact is that minor events do hardly lead to dense temporal graphs.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 958–967,
Seattle, Washington, USA, 18-21 October 2013.

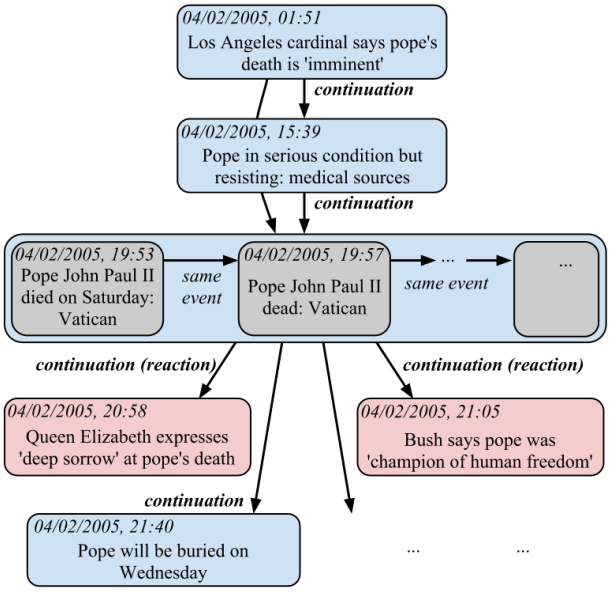


Figure 1: Example of "temporal graph": around the Pope's death. The associated text is the title of each article. Relations that can be obtained by transitivity have been hidden for clarity's sake.

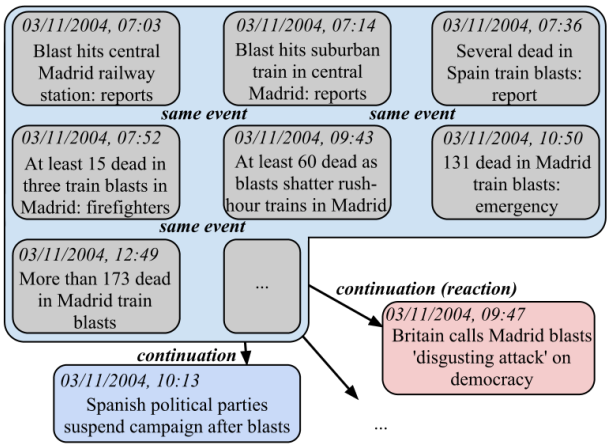


Figure 2: Example of "temporal graph": Madrid attacks, with many updates of the initial information. Note that articles gathered in this main pool of articles can be posterior to the continuations and reactions to the described event.

used for developing, learning and evaluating the system. The simple modules used to predict the *same event*, *continuation* and, possibly, *reaction* relations are described in Section 4, and results are given in Section 5.

We also propose an end-user application to this work. When a user reads an article, the system will then be able to provide her with a thread of events having occurred before or after, helping her to contextualize the information she is reading. This application is described in Section 6.

2 Related work

The identification of temporal relations between events in texts has been the focus of increasing attention because of its importance in NLP applications such as information extraction, question-answering or summarization. The evaluation campaigns TempEval 2007 (Verhagen et al., 2007) and TempEval 2010 (Verhagen et al., 2010) focused on temporal relation identification, mainly on temporal relations between events and times in the same sentence or in consecutive sentences and between events and the creation time of documents. In this context, the goal is to identify the type of a temporal relation which is known to be present. Systems having the best results (accuracy about 0.6) use statistical learning based on temporal features (modality, tense, aspect, etc.) (Mani et al., 2006; Chambers et al., 2007). More recently, Mirroshandel and Ghassem-Sani (2012) proposed a new method for temporal relation extraction by using a bootstrapping method on annotated data and have a better accuracy than state-of-the-art systems. Their method is based on the assumption that similar event pairs in topically related documents are likely to have the same temporal relations. For this work, the authors had already some collections of topically related documents and did not need to identify them.

In the 2012 i2b2 challenge (i2b, 2012), the problem was not only to identify the type of temporal relations, but also to decide whether a temporal relation existed or not between two elements, either clinical concepts or temporal expressions. But, as in TempEval, the temporal analysis were only to be performed within a single document.

Other works focus on event ordering. For example, Fujiki et al. (2003) and Talukdar et al. (2012) proposed methods for automatic acquisition of event sequences from texts. They did not use temporal information present in texts and extracted sequences of events (*e.g.* *arrest/escape*) from sentences which were already arranged in chronologi-

cal order. Chambers and Jurafsky (2008) proposed a method to learn narrative chains of events related to a protagonist in a single document. The first step consists in detecting narrative relations between events sharing coreferring arguments. Then, a temporal classifier orders partially the connected events with the *before* relation.

Concerning the identification of the *reaction* relation, to our knowledge, there is no work on the detection of reaction between several documents. Pouliquen et al. (2007), Krestel et al. (2008) and Balahur et al. (2009) focused on the identification of reported speech or opinions in quotations in a document, but not on the identification of an event which is the source of a reaction and which can possibly be in another document.

As we can see, all these approaches, as well as traditional information extraction approaches, lean on information contained by a single document, and consider an event as a word or a phrase. However, Ahmed et al. (2011) proposed a framework to group temporally and tocipally related news articles into *same story* clusters in order to reveal the temporal evolution of stories. But in these topically related clusters of documents, no temporal relation is detected between articles or events except chronological order. On this point of view, our task is closer to what is done in multidocument summarization, where a system has to detect redundant excerpts from various texts on the same topic and present results in a relevant chronological order. For example, Barzilay et al. (2002) propose a system for multidocument summarization from newswire articles describing the same event. First, similar text units from different documents are identified using statistical techniques and shallow text analysis and grouped into thematic clusters. Then, in each theme, sentences which are selected as part of the summary are ordered using the publication date of the first occurrence of events to order sentences.

3 Resources

We built an annotated collection of English articles, taken from newswire texts provided by the French news agency (AFP), spreading over the period 2004-2012. The entire collection contains about 1.5 million articles. Each document is an XML file containing a title, a creation time (DCT), a set of keywords and textual content split into paragraphs.

3.1 Selection of Article Pairs

Pairs of documents were automatically selected according to the following constraints:

- The article describes an event. Articles such as timelines, fact files, agendas or summaries were discarded (all these kinds of articles were tagged by specific keywords, making the filtering easy).
- The distance between the two DCTs does not exceed 7 days.
- There are at least 2 words in common in the set of keywords and/or 2 proper nouns in common in the first paragraph of each article.

These last two restrictions are important, but necessary, in order to give annotators a chance to find some related articles. Pure random selection of pairs over a collection of 1.5 million articles would be impractical.

We assume that the title and the first paragraph describe the event associated with the document. This is a realistic hypothesis, since the basic rules of journalism impose that the first sentence should summarize the event by informing on the "5 Ws" (*What*, *Who*, *When*, *Where*, *Why*). However, reading more than the first paragraph is sometimes necessary to determine whether a relation exists between two events.

3.2 Relation Annotation

Two annotators were asked to attribute the following relations between each pair of articles presented by the annotation interface system.

In a first annotation round, 7 types of relations were annotated:

- Three relations concerning cases where the two articles relate the same event or an update:
  - *number update*, when a document is an update of numerical data (see top of Figure 5),
  - *form update*, when the second document brings only minor corrections,

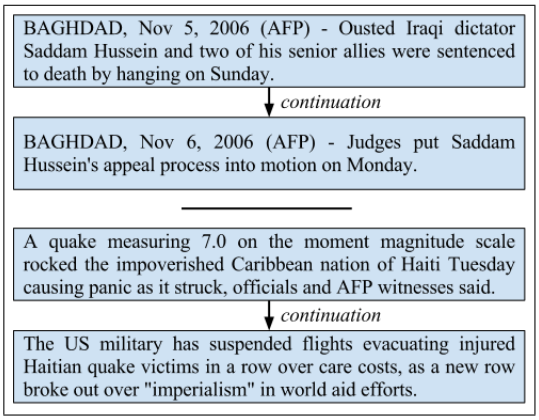


Figure 3: Examples of relation *continuation* between two documents.

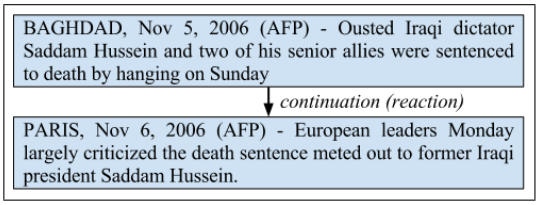


Figure 4: Examples of relation *continuation-reaction* between two documents.

  - *details*, when the second document gives more details about the events (see bottom of Figure 5).

- *development of same story*, when the two documents relate two events which are included into a third one;

- *continuation*, when an event is the continuation or the consequence of a previous one. Figure 3 shows two examples of such a relation. It is important to make clear that a *continuation* relation is more than a simple thematic relation, it implies a natural prolongation between two events. For example, two sport events of the same Olympic Games, or two different attacks in Iraq, shall not be linked together unless a direct link between both is specified in the articles.

- *reaction*, a subset of *continuation*, when a document relates the reaction of someone to another event, as illustrated by the example in Figure 4.

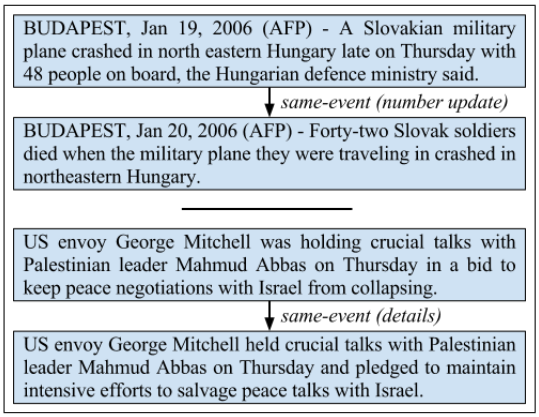


Figure 5: Example of relations *same-event* between two documents: update on casualties (top) or details (bottom).

- *nil*, when no relation can be identified between the two documents.

The inter-annotator agreement was calculated with Cohen's Kappa measure (Cohen, 1960) across 150 pairs: $\kappa = 0.68$. The agreement was low for the first 4 types of relations mostly because the difference between relations was not clear enough. We therefore aggregated the *number update*, *form update* and *details* relations into a more generic and consensual *same-event* relation (see Figure 5). We also discarded the *development of same story* relation, leaving only *same-event*, *continuation* and *reaction*.

Annotation guidelines were modified and a second annotation round was carried out: only the *same-event*, *continuation*, *reaction* and *nil* relations were annotated. Inter-annotator agreement across 150 pairs was then $\kappa = 0.83$, which is a good agreement.

3.3 Relation Set Extension

This manual annotation would have led to very sparse temporal graphs without the two following additional processes:

- When the annotator attributed a "non-nil" relation to a pair of documents, the annotation system suggested other pairs to annotate around the concerned articles.

- *Same-event* and *continuation* relations are transitive: if A *same-event* B and B *same-event* C, then A *same-event* C (and respectively for

	Pair number	Learning	Evaluation
Same event	762	458	304
Continuation	1134	748	386
Reaction	182	123	59
Nil	918	614	304
TOTAL	**2996**	1943	1053

Table 1: Characteristics of the corpus.

continuation). Then, when the annotation was done, a transitive closure was performed on the entire graph, in order to get more relations with low effort (and to detect and correct some inconsistencies in the annotations).

Finally, almost 3,000 relations were annotated. $^2/_3$ of the annotated pairs were used for development and learning phases, while $^1/_3$ were kept for evaluation purpose (cf. Table 1).

4 Building Temporal Graphs

As we explained in the introduction, the main purpose of this paper is to show that it is possible to extract temporal graphs of events from multiple documents in a news corpus. This is achieved with the help of redundancy of information in this corpus. Therefore, we will use a cascade of classifiers and other modules, each of them using the relations deduced by the previous one. All modules predict a relation between two documents (*i.e.*, two events).

We did not focus on complex algorithms or classifiers for tuning our results, and most of our features are very simple. The idea here is to show that good results can be obtained in this original and useful task. The process can be separated into 3 main stages, illustrated in Figure 6:

A. Filtering out pairs that have no relation at all, *i.e.* classifying between *nil* and *non-nil* relations;

B. Classifying between *same-event* and *continuation* relations;

C. Extracting *reactions* from the set of *continuation* relations.

All described classifiers use SMO (Platt, 1998), the SVM optimization implemented into Weka (Hall et al., 2009), with logistic models fitting (option "-M"). With this option, the confidence score of each prediction can be used, while SMO alone provides a constant probability for all instances.

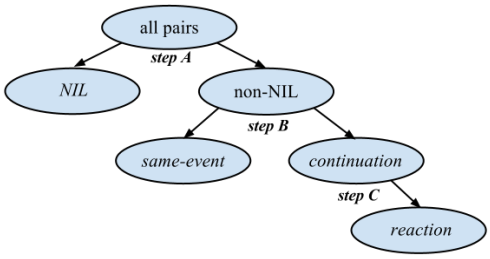


Figure 6: A 3-step classification.

From now on, when considering a pair of documents, we will refer to the older document as document 1, and to the more recent one as document 2. The relations found between documents will be represented by a directed graph where documents are vertices and relations are edges.

4.1 A. Nils *versus* non-nils

We first aim at separating *nil* relations (no relation between two events) from other relations. This step is achieved by two successive classifiers: the first one (A.1) uses mainly similarity measures between documents, while the second one (A.2) uses the relations obtained by the first one.

4.1.1 Step A.1: Nil classifier, level 1

Features provided to the SMO classifier at this first step are based on 3 different similarity measures applied to pairs of titles, pairs of first sentences, and pairs of entire documents: cosine similarity (as implemented by Lucene search engine[3]), inclusion similarity (rate of words from element 1 present in element 2) and overlap similarity (number of words present in both elements). This classifier is therefore based on only 9 features.

4.1.2 Step A.2: Nil classifier, level 2

Finding relations on a document implies that the described event is important enough to be addressed by several articles (*same-event*) or to have consequences (*continuation*). Consequently, if we find such relations concerning a document, we are more likely to find more of them, because this means that

[3] `http://lucene.apache.org`

the document has some importance. A typical example is shown in Figure 7, where an event described by several documents (on the left) has many continuations. For this reason, we build a second classifier A.2 using additional features related to the relations found at step A.1:

- Number of non-nil edges, incoming to or outgoing from document 1 (2 features); the sum of both numbers (1 extra feature);
- Number of non-nil edges, incoming to or outgoing from document 2 (2 features); the sum of both numbers (1 extra feature);
- Number of non-nil edges found involving one of the two documents (*i.e.*, the sum of all edges described above – 1 feature).

These figures have been computed on training set for training, and on result of step A.1 classifier for testing. This new information will basically help the classifier to be more optimistic toward non-nil relations for documents having already non-nil relations.

4.2 B. Same-event *versus* Continuation

We are now working only with *non-nil* relations (even if some relations may switch between *nil* and *non-nil* during the transitive closure).

4.2.1 Step B.1: Relation classifier, level 1

Distinction between *same-event* and *continuation* is made by the following sets of features:

- Date features:
  - Difference between the two document creation times (DCTs): difference in days, in hours, in minutes (3 features);
  - Whether the creation time of doc. 1 is mentioned in doc. 2. For this purpose, we use the date normalization system described in Kessler et al. (2012).
  - Cosine similarity between the first sentence of doc. 1 and sentences of doc. 2 containing the DCT of doc. 1.
  - Cosine similarity between the first sentence of doc. 1 and the most similar sentence of doc. 2.

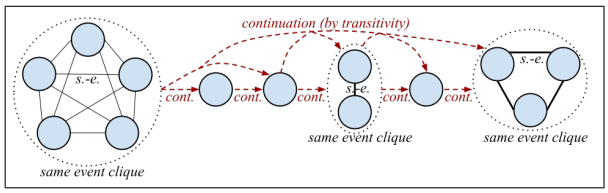


Figure 7: An example of highly-connected subgraph, corresponding to the development of an important story. Same events are grouped by cliques (see Section 4.2.3) and some redundant relations are not shown for clarity's sake.

These last three features come from the idea that a *continuation* relation can be made explicit in text by mentioning the first event in the second document.

- Temporal features: whether words introducing temporal relations occur in document 1 or document 2. These manually-collected words can be prepositions (*after, before*, etc.) or verbs (*follow, confirm, prove*, etc.).
- Reaction features: whether verbs introducing reactions occur in document 1 or document 2 (25 manually-collected verbs as *approve, accept, welcome, vow*, etc.).
- Opinion features: whether opinion words occur in document 1 or document 2. The list of opinion words comes from the MPQA subjectivity lexicon (Wilson et al., 2005).

Only *same-event* relations classified with more than 90% confidence by the classifier are kept, in order to ensure a high precision (recall will be improved at next step). This threshold has been set up on development set.

4.2.2 Step B.2: Relation classifier, level 2

As for step A.2, a second classifier is implemented, using the results of step B.1 with the same manner as A.2 uses A.1 (collecting numbers of *same-event* and *continuation* relations that have been found by the previous classifier).

4.2.3 Steps B.3 and B.4: Transitive closure by vote

As already stated, *same-event* and *continuation* relations are transitive. *Same-event* is also symmetric (A *same-event* B $\Rightarrow$ B *same-event* A). In the

graph formed by documents (vertices) and relations (edges), it is then possible to find all cliques, *i.e.* subsets of vertices such that every two vertices in the subset are connected by a *same-event* relation, as illustrated by Figure 7.

This step does not involve any learning phase. Starting from the result of last step, we find all *same-event* cliques in the graph by using the Bron and Kerbosch (1973) algorithm. The transitive closure process is then illustrated by Figure 8. If the classifier proposed a relation between some documents of a clique and some other documents (as D1, D2 and D3), then a vote is necessary:

- If the document is linked to half or more of the clique, then all missing links are created (Figure 8.a);
- Otherwise, the document is entirely disconnected from the clique (Figure 8.b).

This vote is done for *same-event* and *continuation* relations (resp. steps B.3 and B.4). Only cliques containing at least 3 nodes are used. A drawback of this voting procedure is that the final result may not be independent of the voting order, in some cases. However, it is assured that the result is consistent, *i.e.* that no document will sit in two different cliques, or that two documents from the same clique will not have two different relations toward a third document.

Note that this vote leads to improvements only if the precision of the initial classifier is sufficiently good. As we will see in Section 5.2, this is the case in our situation, but one must keep in mind that a vote leaning on too imprecise material would lead to even worse results. Some experiments on the development set show us that at least 70% precision was necessary. Another way to ensure robustness of the vote would be to apply the transitive closure only on bigger cliques (*e.g.*, containing more than 3 or 4 nodes).

4.3 C. Continuation *versus* Reaction

The approach for reaction extraction is different. We first try to determine which documents describe reactions, regardless of which event it is a reaction to. In the training set, all documents having at least one incoming *reaction* edge are considered as reaction

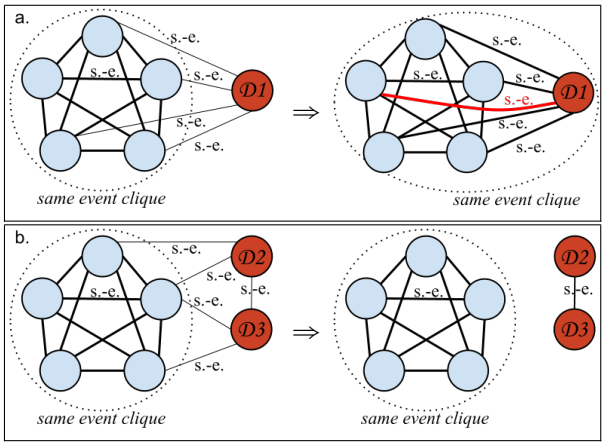


Figure 8: Vote for *same-event* transitive closure. At the top (a.), four nodes from the 5-node clique are linked to document D_1, which is enough to add D_1 to the clique. At the bottom (b.), only two nodes from the clique are linked to documents D_2 and D_3, which is not enough to add them into the clique. All edges from the clique to D_2 and D_3 are then deleted.

documents, all others are not. This distinction is then learned with the same model and features as for step B.1 (Section 4.2.1).

Once reaction documents have been selected, the question is how to decide to which other document(s) it must be linked. For example, in Figure 1, *"Queen Elizabeth expresses deep sorrow"* is a reaction to pope's death, not to other documents in the temporal thread (for example, not to other reactions or to *"Pope in serious condition"*). We did not manage to build any classifier leading to satisfying results at this point. We then proposed the two following basic heuristics, applied on all *continuation* relations found after step B:

- A reaction reacts to only one event.
- A reaction reacts to an important event. Then, among all *continuation* edges incoming to the reaction document, we choose the biggest *same-event* clique and create *reaction* edges instead of *continuations*. If there is no clique (only single nodes) or several same-size cliques, all of them are tagged as *reactions*.

This module is called step C.1. Finally, a transitive closure is performed for reactions (C.2).

Relation	Precision	Recall	F1
NIL	0.754	0.821	0.786
same-event	0.832	0.812	0.822
continuation	0.736	0.696	0.715
↳ *reaction*	0.273	0.077	0.120

Table 2: Results obtained by the baseline system. *Continuation* scores do not consider reactions, only the last row makes the distinction.

5 Results

5.1 Baseline

As a baseline, we propose a single classifier determining all classes at once, based on the same SMO classifier with the exact same parameters and all similarity-based features (on titles, first sentences and entire documents) described in Section 4.1.1.

Table 2 shows results for this baseline. Unsurprisingly, *same-event* relations are quite well classified by this baseline, since similarity is the major clue for this class. *Continuation* is much lower and only 3 reactions are well detected.

5.2 System Evaluation

Results for all successive steps described in previous section are shown in Figure 3. The final result of the entire system is the last one. The first observation is that redundancy-based steps improve performance in a significant manner:

- Classifiers A.2 and B.2, using the number of incoming or outgoing edges found at previous steps, lead to very significant improvement.
- Among transitivity closure algorithms (B.3, B.4, C.2), only *same-event* transitivity B.3 leads to significant improvement. Furthermore, as we already noticed, these algorithms must be used only when a good precision is guaranteed at previous step. Otherwise, there is a risk of inferring mostly bad relations. This is why we biased classifier at step B.1 towards precision. Finally, if this condition on precision is true, transitivity closure is a robust way to get new relations for free.

Results also tell that classification of relations *same-event* and *continuation* is encouraging. *Reaction* level gets a fair precision but a bad recall. This is not catastrophic since most of the missed reactions are tagged as *continuation*, which is still true (only 10% of the *reaction* relations are mistagged as *same-event*). However, there is big room for improvement on this point.

Step	**Relation**	**Precision**	**Recall**	**F1**
A. *NIL* vs *non-NIL classifier*				
A.1	NIL	0.764	0.815	0.788
	non-NIL	0.921	0.896	0.910
A.2 ***	NIL	0.907	0.811	0.857
	non-NIL	0.925	0.966	0.945
B. *Same-event* vs *continuation classifier*				
B.1	NIL	0.907	0.811	0.857
	same-event	0.870	0.553	0.676
	continuation	0.664	0.867	0.752
B.2 ***	NIL	0.947	0.831	0.885
	same-event	0.894	0.724	0.800
	continuation	0.744	0.911	0.819
B.3 **	NIL	0.884	0.831	0.857
	same-event	0.943	0.819	0.877
	continuation	0.797	0.906	0.848
B.4 *	NIL	0.890	0.831	0.860
	same-event	0.943	0.819	0.877
	continuation	0.798	0.911	0.851
C. *Reaction* vs *continuation*				
C.1	NIL	**0.890**	**0.831**	**0.860**
C.2	same-event	**0.943**	**0.819**	**0.877**
	continuation	**0.798**	**0.911**	**0.851**
	↳ reaction	**0.778**	**0.359**	**0.491**

Table 3: Results obtained at each step of the classification process. The significance of the improvement wrt previous step (when relevant) is indicated by the Student t-test (*: non significant; **: $p < 0.05$ (significant); ***: $p < 0.01$ (highly significant)). Steps C.1 and C.2 are aggregated, since their results are exactly the same.

6 Application

As we showed in previous section, results for classification of *same-event* and *continuation* relations between documents are good enough to use this system in an application that builds "event threads" around an input document. The use case is the following:

- The reader reads an article (let's say, about the death of John Paul II, article published on Feb. 4th, 2005 (UT) – see Figure 1).
- A link in the page suggests the user to visualize the event thread around this article.

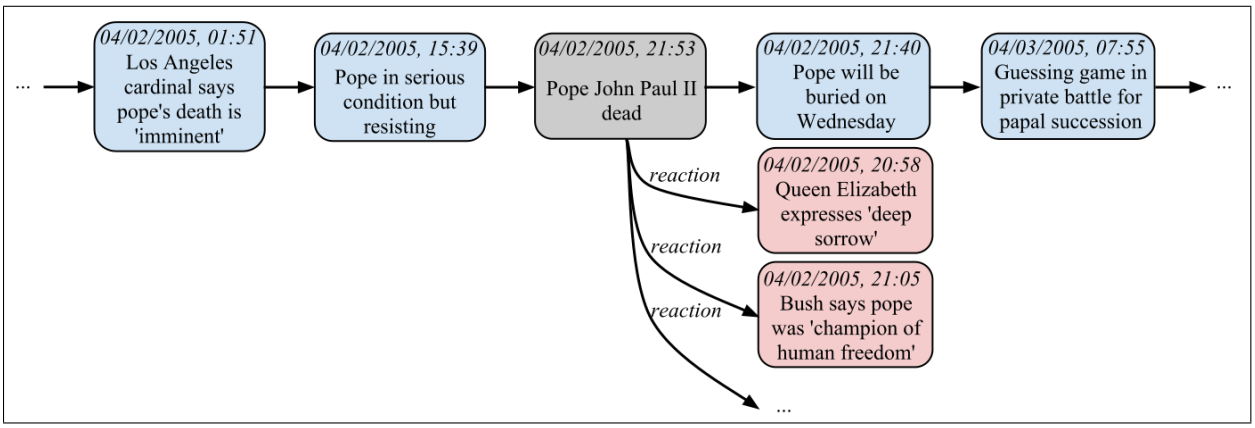


Figure 9: An example of temporal thread obtained on the death of John Paul II for user visualization (see corresponding relation graph in Figure 1).

- All articles within a period of 7 days around the event, sharing at least two keywords with the current document, are collected. All pairs are given to the system[4].
- When *same-event* cliques are found, only the longest article (often, the most recent one) of each clique is presented to the user. However, the date and time presented to the user are those of the first article relating the event.
- This leads to a graph with only *continuation* and *reaction* relations. Edges are "cleaned" so that a unique thread is visible: relations that can be obtained by transitivity are removed, edges between two documents are kept only if no document can be inserted in-between.
- Nodes are presented in chronological order. The user can visualize and navigate through this graph (the event thread shows only titles but full articles can be accessed by clicking on the node).
- When found, reactions are isolated from the main thread.
- Such a temporal thread is potentially infinite. If the user navigates through the end of the 7-day window, the system must be run again on the next time span.

[4]In case of very important events where "all pairs" would be too much, the temporal window is restrained. However, there is no real time performance issue in this system.

Figure 9 presents the result of this process on the partial temporal graph shown in Figure 1.

7 Conclusion

This article presents a task of multidocument temporal graph building. We make the assumption that each news article (after filtering) relates an event, and we present a system extracting relations between articles. This system uses simple features and algorithms but takes advantage of the important redundancy of information in a news corpus, by incorporating redundancy information in a cascade of classifiers, and by using transitivity of relations to infer new links.

Finally, we present an application presenting "event threads" to the user, in order to contextualize the information and recomposing the story of an event.

Now that the task is well defined and that encouraging results have been obtained, we envisage to enrich classifiers by more fine-grained temporal and lexical information, such as narrative chains (Chambers and Jurafsky, 2008) for *continuation* relation or event clustering (Barzilay et al., 2002) for *same-event* relation. There is no doubt that *reaction* detection can be improved a lot, by going beyond simple lexical features and discovering specific patterns. We also intend to adapt the described system to other languages than English.

References

A. Ahmed, Q. Ho, J. Eisenstein, E.P. Xing, A.J. Smola, and C.H. Teo. 2011. Unified Analysis of Streaming News. In *Proceedings of WWW*, Hyderabad, India.

A. Balahur, R. Steinberger, E. van der Goot, B. Pouliquen, and M. Kabadjov. 2009. Opinion Mining on Newspaper Quotations. In *Proceedings of International Joint Conference on Web Intelligence and Intelligent Agent Technologies*, Milano, Italy.

R. Barzilay, N. Elhadad, and K.R. McKeown. 2002. Inferring Strategies for Sentence Ordering in Multidocument News Summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

C. Bron and J. Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

N. Chambers and D. Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of the 46th Annual Meeting of the ACL*, Columbus, USA.

N. Chambers, S. Wang, and D. Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Prague, Czech Republic, June.

J. Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 43(6):551–558.

T. Fujiki, H. Nanba, and M. Okumura. 2003. Automatic Acquisition of Script Knowledge from a Text Collection. In *Proceedings of EACL*, Budapest, hungary.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).

2012. *Proceedings of i2b2/VA Shared-Tasks and Workshop on Challenges in Natural Language Processing for Clinical Data*, Chicago, USA.

R. Kessler, X. Tannier, C. Hagège, V. Moriceau, and A. Bittar. 2012. Finding Salient Dates for Building Thematic Timelines. In *Proceedings of the 50th Annual Meeting of the ACL*, Jeju Island, Republic of Korea.

R. Krestel, S. Bergler, and R. Witte. 2008. Minding the Source: Automatic Tagging of Reported Speech in Newspaper Articles. In *Proceedings of LREC*, Marrakech, Morocco.

I. Mani, M. Verhagen, B. Wellner, C. Lee, and J. Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, Sydney, Australia.

S.A. Mirroshandel and G. Ghassem-Sani. 2012. Towards Unsupervised Learning of Temporal Relations between Events. In *Journal of Artificial Intelligence Research*, volume 45.

J.C. Platt, 1998. *Advances in Kernel Methods - Support Vector Learning*, chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization. MIT Press.

B. Pouliquen, R. Steinberger, and C. Best. 2007. Automatic Detection of Quotations in Multilingual News. In *Proceedings of RANLP*, Borovets, Bulgaria.

P.P. Talukdar, D. Wijaya, and T. Mitchell. 2012. Acquiring Temporal Constraints between Relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, Hawaii.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. 2007. SemEval-2007 - 15: TempEval Temporal Relation Identification. In *Proceedings of SemEval workshop at ACL*, Prague, Czech Republic.

M. Verhagen, R. Sauri, T. Caselli, and J. Pustejovsky. 2010. SemEval-2010 - 13: TempEval-2. In *Proceedings of SemEval workshop at ACL*, Uppsala, Sweden.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of HLT-EMNLP*.

Tree Kernel-based Negation and Speculation Scope Detection with Structured Syntactic Parse Features

Bowei Zou **Guodong Zhou** **Qiaoming Zhu***
Natural Language Processing Lab, School of Computer Science and Technology
Soochow University, Suzhou, 215006, China
zoubowei@gmail.com, {gdzhou,qmzhu}@suda.edu.cn

Abstract

Scope detection is a key task in information extraction. This paper proposes a new approach for tree kernel-based scope detection by using the structured syntactic parse information. In addition, we have explored the way of selecting compatible features for different part-of-speech cues. Experiments on the BioScope corpus show that both constituent and dependency structured syntactic parse features have the advantage in capturing the potential relationships between cues and their scopes. Compared with the state of the art scope detection systems, our system achieves substantial improvement.

1 Introduction

The task of scope detection is to detect the linguistic scope dominated by a specific cue. Current researches in this field focus on two semantic aspects: negation and speculation. The negative scope detection is to detect the linguistic scope which is repudiated by a negative word (viz., negative cue, e.g., "*not*"). In other side, the speculative scope detection is to detect the uncertain part in a sentence corresponding to the speculative word (viz., speculative cue, e.g., "*seems*"). See the sentence 1) below, the negative cue "*not*" dominates the scope of "*not expensive*". Similarly, the speculative cue "*possible*" in sentence 2) dominates the uncertain scope "*the possible future scenarios*".

1) *The chair is* [***not*** *expensive*] *but comfortable*.
2) *Considering all that we have seen, what are now* [*the* ***possible*** *future scenarios*]*?*

The negative and speculative scope detection task consists of two basic stages. The first one is to identify the sentences involving negative or speculative meaning. The second stage is to detect the linguistic scope of the cue in sentences (Velldal et al, 2012). In this paper, we focus on the second stage. That is, by given golden cues, we detect their linguistic scopes.

We propose a tree kernel-based negation and speculation scope detection with structured syntactic parse features. In detail, we regard the scope detection task as a binary classification issue, which is to classify the tokens in a sentence as being inside or outside the scope. In the basic framework, we focus on the analysis and application of structured syntactic parse features as follows:

Both constituent and dependency syntactic features have been proved to be effective in scope detection (Özgür et al, 2009; Øvrelid et al, 2010). However, these flat features are hardly to reflect the information implicit in syntactic parse tree structures. Our intuition is that the segments of the syntactic parse tree around a negative or speculative cue is effective for scope detection. The related structures normally underlay the indirect clues to identify the relations between cues and their scopes, e.g., in sentence 1), "***but*** *something*", as a frequently co-occurred syntactic structure with "***not*** *something*", is an effective clue to determine the linguistic scope of "***not***".

The tree kernel classifier (Moschitti, 2006) based on support vector machines uses a kernel function between two trees, affording a comparison between their substructures. Therefore, a tree kernel-based scope detection approach with structured syntactic parse tree is employed. The tree

* Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 968–976,
Seattle, Washington, USA, 18-21 October 2013.

kernel has been already proved to be effective in semantic role labeling (Che et al, 2006) and relation extraction (Zhou et al, 2007).

In addition, the empirical observation shows that features have imbalanced efficiency for scope classification, which is normally affected by the part-of-speech (abbr., POS) of cues. Hence, we build the discriminative classifiers for each kind of POS of cues, then explore and select the most compatible features for them.

We construct a scope detection system by using the structured syntactic parse features based tree kernel classification. Compared with the state of the art scope detection systems, our system achieves the performance of accuracy 76.90% on negation and 84.21% on speculation (on Abstracts sub-corpus). Additionally, we test our system on different sub-corpus (Clinical Reports and Full Papers). The results show that our approach has better cross-domain performance.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces the corpus and corresponding usage in our experiments. Section 4 describes our approach and the experiments are presented in Section 5. Finally, there is a conclusion in Section 6.

2 Related Work

Most of the previous studies on negation and speculation scope detection task can be divided into two main aspects: the heuristic rule based methods and the machine learning based methods. We respectively introduce the aspects in below.

2.1 Heuristic Rule based Methods

The initial studies for scope detection are to compile effective heuristic rules (Chapman et al, 2001; Goldin et al, 2003). Recently, the heuristic rule based methods have further involved the syntactic features.

Huang et al (2007) implemented a hybrid approach to automated negation scope detection. They combined the regular expression matching with grammatical parsing: negations are classified on the basis of syntactic categories and located in parse trees. Their hybrid approach is able to identify negated concepts in radiology reports even when they are located at some distance from the negative term.

Özgür et al (2009) hypothesized that the scope of a speculation cue can be characterized by its part-of-speech and the syntactic structure of the sentence and developed rules to map the scope of a cue to the nodes in the syntactic parse tree. By given golden speculation cues, their rule-based method achieves the accuracies of 79.89% and 61.13% on the Abstracts and the Full-Papers sub-corpus, respectively.

Øvrelid et al (2010) constructed a small set of heuristic rules which define the scope for each cue. In developing these rules, they made use of the information provided by the guidelines for scope annotation in the BioScope corpus, combined with manual inspection of the training data in order to further generalize over the phenomena discussed by Vincze et al (2008) and work out interactions of constructions for various types of cues.

Apostolova et al (2011) presented a linguistically motivated rule-based system for the detection of negation and speculation scopes that performs on par with state-of-the-art machine learning systems. The rules are automatically extracted from the BioScope corpus and encode lexico-syntactic patterns in a user-friendly format. While their system was developed and tested using a biomedical corpus, the rule extraction mechanism is not domain-specific.

The heuristic rule based methods have bad robustness in detecting scopes crossing different meaning aspects (e.g., negative vs. speculative) and crossing different linguistic resources (e.g., Technical Papers vs. Clinical Reports).

2.2 Machine Learning based Methods

The machine learning based methods have been ignored until the release of the BioScope corpus (Szarvas et al, 2008), where the large-scale data of manually annotated cues and corresponding scopes can support machine learning well.

Morante et al (2008) formulated scope detection as a chunk classification problem. It is worth noting that they also proposed an effective proper post-processing approach to ensure the consecutiveness of scope. Then, for further improving the scope detection, Morante et al (2009a) applied a meta-learner that uses the predictions of the three classifiers (TiMBL/SVM/CRF) to predict the scope.

For the competitive task in CoNLL'2010 (Farkas et al, 2010), Morante et al (2010) used a

memory-based classifier based on the k-nearest neighbor rule to determine if a token is the first token in a scope sequence, the last, or neither. Therefore, in order to guarantee that all scopes are continuous sequences of tokens they apply a first post-processing step that builds the sequence of scope.

The existing machine learning based approaches substantially improve the robustness of scope detection, and have nearly 80% accuracy. However, the approaches ignore the availability of the structured syntactic parse information. This information involves more clues which can well reflect the relations between cues and scopes. Sánchez et al (2010) employed a tree kernel based classifier with CCG structures to identify speculative sentences on Wikipedia dataset. However, in Sánchez's approach, not all sentences are covered by the classifier.

3 Corpus

We have employed the BioScope corpus (Szarvas et al, 2008; Vincze et al, 2008)[1], an open resource from the biomedical domain, as the benchmark corpus. The corpus contains annotations at the token level for negative and speculative cues and at the sentence level for their linguistic scope (as shown in Figure 1).

<sentence id="S26.8"> *These findings* <xcope id="X26.8.2"> <cue type="**speculation**" ref="X26.8.2"> *indicate that* </cue> <xcope id="X26.8.1"> *corticosteroid resistance in bronchial asthma* <cue type="**negation**" ref="X26.8.1"> *can not* </cue> *be explained by abnormalities in corticosteroid receptor characteristics* </xcope></xcope> . </sentence>

(Note: <Sentence> denotes one sentence and the tag "id" denotes its serial number; <xcope> denotes the scope of a cue; <cue> denotes the cue, the tag "type" denotes the specific kind of cues and the tag "ref" is the cue's serial number.)

Figure 1. An annotated sentence in BioScope.

The BioScope corpus consists of three sub-corpora: biological Full Papers from FlyBase and BMC Bioinformatics, biological paper Abstracts from the GENIA corpus (Collier et al, 1999), and Clinical Reports. Among them, the Full Papers sub-corpus and the Abstracts sub-corpus come from the same genre. In comparison, the Clinical Reports sub-corpus consists of clinical radiology reports with short sentences.

[1] http://www.inf.u-szeged.hu/rgai/bioscope

In our experiments, if there is more than one cue in a sentence, we treat them as different cue and scope (two independent instances). The statistical data for our corpus is presented in Table 1 in below.

The average length of sentences in the negation portion is almost as long as that in speculation, while the average length of scope in negation is shorter than that in speculation. In addition, the length of sentence and scope in both Abstracts and Full Papers sub-corpora is comparative. But in Clinical Reports sub-corpus, it is shorter than that in Abstracts and Full Papers. Thus, looking for the effective features in short sentences is especially important for improving the robustness for scope detection.

		Abstract	**Paper**	**Clinical**
Negation	Sentences	1594	336	441
	Words	46849	10246	3613
	Scopes	1667	359	442
	Av. Len Sentence	29.39	30.49	8.19
	Av. Len Scope	9.62	9.36	5.28
Speculation	Sentences	2084	519	854
	Words	62449	16248	10241
	Scopes	2693	682	1137
	Av. Len Sentence	29.97	31.31	11.99
	Av. Len Scope	17.24	15.58	6.99

(Note: "Av. Len" stands for average length.)

Table 1. Statistics for our corpus in BioScope.

4 Methodology

We regard the scope detection task as a binary classification problem, which is to classify each token in sentence as being the element of the scope or not. Under this framework, we describe the flat syntactic features and employ them in our benchmark system. Then, we propose a tree kernel-based scope detection approach using the structured syntactic parse features. Finally, we construct the discriminative classifier for each kind of POS of cues, and select the most compatible features for each classifier.

4.1 Flat Syntactic Features

In our benchmark classification system, the features relevant to the cues or tokens are selected. Then, we have explored the constituent and dependency syntactic features for scope detection. These features are all flat ones which reflect the characteristic of tokens, cues, scopes, and the relation between them.

Basic Features: Table 2 shows the basic features which directly relate to the characteristic of cues or tokens in our basic classification.

Feature	Remark
B1	Cue.
B2	Candidate token.
B3	Part-of-speech of candidate token.
B4	Left token of candidate token.
B5	Right token of candidate token.
B6	Positional relation between cue and token.

Table 2. Basic features.

Constituent Syntactic Features: For improving the basic classification, we employ 10 constituent features belonging to two aspects. On the one hand, we regard the linguistic information of the neighbor locating around the candidate tokens as the coherent features (CS1~CS6 in Table 3). These features are used for detecting the close cooperation of a candidate token co-occurring with its neighbors in a scope. On the other hand, we regard the linguistic characteristics of the candidate tokens themselves in a syntactic tree as the inherent features (CS7~CS10 in Table 3). These features are used for determining whether the token has the direct relationship with the cue or not.

Features	Remarks
CS1	POS of left token.
CS2	POS of right token.
CS3	Syntactic category of left token.
CS4	Syntactic category of right token.
CS5	Syntactic path from left token to the cue.
CS6	Syntactic path from right token to the cue.
CS7	Syntactic category of the token.
CS8	Syntactic path from the token to the cue.
CS9	Whether the syntactic category of the token is the ancestor of the cue.
CS10	Whether the syntactic category of the cue is the ancestor of the token.

Table 3. Constituent syntactic features.

Features	Remarks
DS1	Dependency direction (*"head"or "dependent"*).
DS2	Dependency syntactic path from the token to cue.
DS3	The kind of dependency relation between the token and cue.
DS4	Whether the token is the ancestor of the cue.
DS5	Whether the cue is the ancestor of the token.

Table 4. Dependency syntactic features.

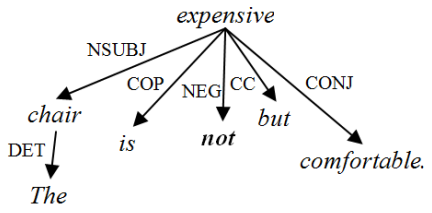
Dependency Syntactic Features: For the effectiveness to obtain the syntactic information far apart from cues, we use 5 dependency syntactic features which emphasize the dominant relationship between cues and tokens by dependency arcs as shown in Table 4.

The features in Table 2, 3, and 4 have imbalanced classification for the scope classification. Therefore, we adopt the greedy feature selection algorithm as described in Jiang et al (2006) to pick up positive features incrementally according to their contributions. The algorithm repeatedly selects one feature each time, which contributes most, and stops when adding any of the remaining features fails to improve the performance.

4.2 Structured Syntactic Features

Syntactic trees involve not only the direct bridge (e.g., syntactic path) between cue and its scope but also the related structures to support the bridge (e.g., sub-tree). The related structures normally involve implicit clues which underlay the relation between cue and its scope. Therefore, we use the constituent and dependency syntactic structures as the supplementary features to further improve the benchmark system.

Furthermore, we employ the tree kernel-based classifier to capture the structured information both in constituent and dependency parsing trees. The results of the constituent syntactic parser are typical trees which always consist of the syntactic category nodes and the terminal nodes. Thus, the constituent syntactic tree structures could be used in tree kernel-based classifier directly, but not for the dependency syntactic tree structures. As Figure 2 shows, in sentence "*The chair is not expensive but comfortable.*" the tree kernels cannot represent the relations on the arcs (e.g., "*CONJ*" between "*expensive*" and "*comfortable*"). It is hard to use the relations between tokens and cues in tree kernels.



Structure of Dependency Tree

Cue: ***not***

Scope: *not expensive*

Figure 2: The dependency tree of sentence "*The chair is not expensive but comfortable*."

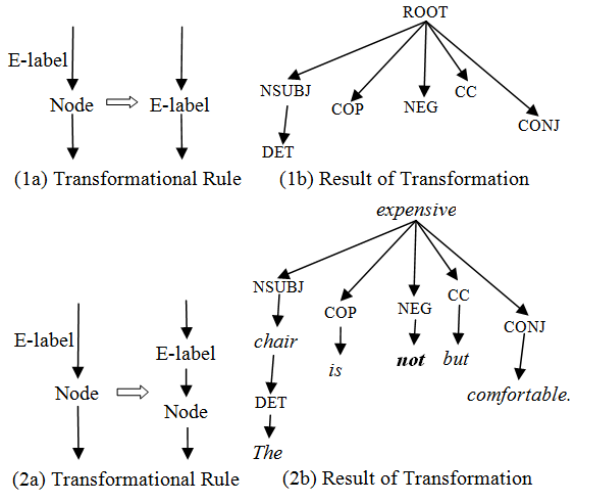


Figure 3. Two transformational rules.

To solve the problem, we transform the dependency tree into other two forms capable of being used directly as the compatible features in tree-kernel based classification. The transformational rules are described as below:

(1) Extracting the dependency relations to generate a tree of pure relations (named dependency relational frame), where the tokens on the nodes of original dependency tree are ignored and only the relation labels are used. E.g., the tokens "*chair*", "*is*", etc in Figure 2 are all deleted and replaced by the corresponding relation labels. E.g., "NSUBJ", "COP", etc are used as nodes in the dependency relational frame, see (1a) & (1b) in Figure 3.

(2) Inserting the tokens which have been deleted in step (1) into the dependency relational frame and making them follow and link with their original dependency relations. E.g., the tokens "*chair*", "*is*", etc are added below the nodes "NSUBJ", "COP", etc, see (2a) & (2b) in Figure 3.

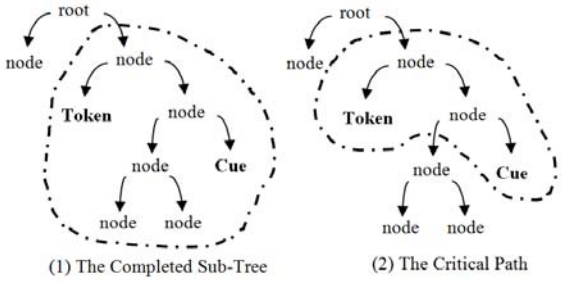


Figure 4. Two transformations for tree-kernel.

Within the constituent and dependency syntactic trees, we have employed both the Completed Sub-Tree and the Critical Path as the syntactic structure features for our classification. The former is a minimum sub-tree that involves the cues and the tokens, while the latter is the path from the cues to the tokens in the completed tree containing the primary structural information. Figure 4 shows them.

4.3 Part-of-Speech Based Classification Optimization

Motivating in part by the rule-based approach of Özgür et al (2009), we infer that features have imbalanced efficiency for scope classification, normally affected by the part-of-speech (POS) of cues.

POS of Cues	Number	POS of Cues	Number
CC	157	VB	31
IN	115	VBD	131
JJ	238	VBG	225
MD	733	VBN	112
NN	43	VBP	561
RB	137	VBZ	207

Table 5. Distribution of different POSs of speculative cues in Abstracts sub-corpus.

Table 5 shows the distribution for different POSs of cues in the Abstracts sub-corpus of BioScope for speculation detection task. The cues of different POS usually undertake different syntactic roles. Thus, there are different characteristics in triggering linguistic scopes. See the two examples below:

3) *TCF-1 contained a single DNA box in the* [***putative*** *mammalian sex-determining gene SRY*].
4) *The circadian rhythm of plasma cortisol* [***either*** *disappeared* ***or*** *was inverted*].

The speculative cue "*putative*" in sentence 3) is an adjective. The corresponding scope is its modificatory structure ("*putative mammalian sex-determining gene SRY*"). In sentence 4), "*either…or…*" is a conjunction speculation cue. Its scope is the two connected components ("*either disappeared or was inverted*"). Thus, the effective features for the adjectival cue are normally the dependency features, e.g., the features of DS1 and DS5 in Table 4, while the features for the conjunction cue are normally the constituent information, e.g., the features of CS9 in Table 3.

In Table 5, considering the different function of verb voice, we cannot combine the "VB(*)" POS. For instance, the POS of "*suggest*" in sentence 5) is "VBP" (the verb present tense). The corresponding scope does not involve the sentence subject.

The POS of "*suggested*" in sentence 6) is "VBN" (the past participle). The scope involves the subject "*An age-related decrease*".

5) *These results* [***suggest*** *that the genes might be involved in terminal granulocyte differentiation*].
6) [*An age-related decrease was* ***suggested*** *between subjects younger than 20 years*].

As a result, we have built a discriminative classifier for each kind of POS of cues, and then explored and selected the most compatible features for each classifier.

5 Experiments and Results

5.1 Experimental Setting

Considering the effectiveness of different features, we have split the Abstracts sub-corpus into 5 equal parts, within which 2 parts are used for feature selection (Feature Selection Data) and the rest for the scope detection experiments (Scope Detection Data). The Feature Selection Data are divided into 5 equal parts, within which 4 parts for training and the rest for developing. In our scope detection experiments, we divide the Scope Detection Data into 10 folds randomly, so as to perform 10-fold cross validation. As the experiment data is easily confusable, Figure 5 illustrates the allocation.

Checking the validity of our method, we use the Abstracts sub-corpus in Section 5.2, 5.3 and 5.4, while in Section 5.5 we use all of the three sub-corpora (Abstracts, Full Papers, and Clinical Reports) to test the robustness of our system when applied to different text types within the same domain.

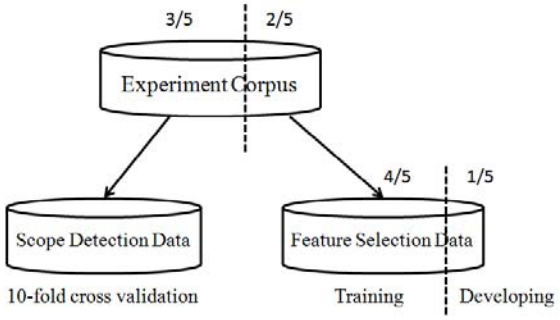


Figure 5. The allocation for experiment data.

The evaluation is made using the precision, recall and their harmonic mean, F1-score. Additionally, we report the accuracy in PCS (Percentage of Correct Scopes) applied in CoNLL'2010, within which a scope is fully correct if all tokens in a sentence have been assigned to the correct scope class for a given cue. The evaluation in terms of precision and recall measures takes a token as a unit, whereas the evaluation in terms of PCS takes a scope as a unit. The key toolkits for scope classification include:

Constituent and Dependency Parser: All the sentences in BioScope corpus are tokenized and parsed using the Berkeley Parser (Petrov et al, 2007) [2] which have been trained on the GENIA TreeBank 1.0 (Tateisi et al, 2005)[3], a bracketed corpus in PTB style. 10-fold cross-validation on GTB1.0 shows that the parser achieves 87.12% in F1-score. On the other hand, we obtain the dependency relations by the Stanford Dependencies Parser[4].

Support Vector Machine Classifier: SVM^{Light}[5] is selected as our classifier, which provides a way to combine the tree kernels with the default and custom SVM^{Light} kernels. We use the default parameter computed by SVM^{Light}.

Besides, according to the guideline of the BioScope corpus, scope must be a continuous chunk. The scope classifier may result in discontinuous blocks, as each token may be classified inside or outside the scope. Therefore, we perform the rule based post-processing algorithm proposed by Morante et al (2008) to obtain continuous scopes.

5.2 Results on Flat Syntactic Features

Relying on the results of the greedy feature selection algorithm (described in Section 4.1), we obtain 9 effective features {B1, B3, B6, CS3, CS4, CS9, DS1, DS3, DS5} (see Table 2, 3 and 4) for negation scope detection and 13 effective features {B3, B4, B5, B6, CS1, CS5, CS6, CS8, CS9, CS10, DS1, DS4, DS5} for speculation. Table 6 lists the performances on the Scope Detection Data by performing 10-fold cross validation. It shows that flat constituent and dependency syntactic features significantly improve the basic scope detection by 13.48% PCS for negation and 30.46% for speculation (χ^2; $p < 0.01$). It demonstrates that the selected syntactic features are effective for scope detection.

[2] http://code.google.com/p/berkeleyparser
[3] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA
[4] http://nlp.stanford.edu/software/lex-parser.shtml
[5] http://svmlight.joachims.org

	Features	P	R	F	PCS
Negation	Basic	89.89	68.72	77.86	39.50
	Con.	85.72	67.80	75.66	41.81
	Dep.	90.31	69.01	78.19	40.08
	Bas.&Con.	88.86	79.07	83.61	51.64
	Bas.&Dep.	90.44	73.62	81.17	49.36
	All	91.21	76.57	83.25	52.98
	Features	**P**	**R**	**F**	**PCS**
Speculation	Basic	89.67	86.86	88.24	40.09
	Con.	96.43	87.46	91.72	66.57
	Dep.	90.84	87.04	88.89	44.45
	Bas.&Con.	95.66	92.08	93.83	69.59
	Bas.&Dep.	92.39	88.27	90.28	67.49
	All	95.71	92.09	93.86	70.55

(Note: "Bas." denotes basic features; "Con." denotes Constituent features; "Dep." denotes Dependency features; "All" contains Basic, Constituent, and Dependency features being selected.)

Table 6. Performance of flat syntactic features.

The results also show that the speculative scope detection achieves higher performance (16.98% higher in PCS) (χ^2; $p < 0.01$) than the negation scope detection. The main reason is that although the average sentence length of negation and speculation are comparable (29.97 vs. 29.39 words, in Table 1), the average length of speculation scopes is much longer than the negation (17.24 vs. 9.62 words, in Table 1) in Abstracts sub-corpus. With the shorter scopes in training data, the classifier inevitably have more negative samples. Thus, by using a token as the basic unit in our classification, the imbalanced samples will seriously mislead the classifier and result in bias on the negative samples.

In addition, both constituent and dependency flat features can improve the scope classification, for the reason that the constituent features usually provide the nearer syntactic information of the cues, and that the further syntactic information between cues and scopes have been obtained by the dependency features.

5.3 Results on Structured Syntactic Parse Features

Table 7 and Table 8 give the scope detection performance using the different structured syntactic parse features on negation and speculation respectively. Compared to the optimal system (using all of the selected flat features in Table 6) in Section 5.2, the structured syntactic parse features at best improve the scope classification nearly 17.29% on negation (PCS=70.27%) and 12.32% on speculation (PCS=82.87%) (χ^2; $p < 0.01$). It indicates that the structured syntactic parse features can provide more implicit linguistic information, as supplementary clues, to support scope classification.

The improvements also show that both the completed syntactic sub-trees and critical paths in constituent and dependency parsing trees are effective. The reason is that the completed syntactic sub-trees contain the surrounding information related to cues and tokens, while there are more direct syntactic information in the critical paths between cue and its scope.

Features	P	R	F	PCS
Con. CT	91.12	83.25	86.89	54.57
Con. CT&CP	93.31	89.32	91.20	66.58
Dep. T1 CT	87.29	84.37	85.81	53.07
Dep. T1 CT&CP	90.03	86.77	88.37	59.53
Dep. T2 CT	88.17	84.58	86.34	53.76
Dep. T2 CT&CP	91.09	87.31	89.16	60.11
All	93.84	91.94	92.88	70.27

(Note: "Con." denotes Constituent features; "Dep." denotes Dependency features; "T1" use the transformational rule (1) in Section 4.2 to get the dependency tree; "T2" use the transformational rule (2) in Section 4.2 to get the dependency tree; CT-"Completed syntactic sub-Tree"; CP-"Critical Path"; "All" contains Con CT&CP, Dep T1 CT&CP and Dep T2 CT&CP)

Table 7. Performance of structured syntactic parse features on negation.

Features	P	R	F	PCS
Con. CT	95.89	93.37	94.61	75.17
Con. CT&CP	96.05	94.36	95.20	76.73
Dep. T1 CT	93.24	90.77	91.99	72.31
Dep. T1 CT&CP	94.28	92.30	93.28	73.75
Dep. T2 CT	93.76	89.68	91.67	73.06
Dep. T2 CT&CP	95.29	94.55	94.92	75.69
All	96.93	96.86	96.89	82.87

Table 8. Performance of structured syntactic parse features on speculation.

5.4 Results on Part-of-Speech Based Classification

To confirm the assumption in Section 4.3, we have built a discriminative classifier for each kind of POS of cues. Considering that the features involving the global structured syntactic parse information in Section 4.2 are almost effective to all instances, we only use the flat syntactic features in Section 4.1.

	System	P	R	F	PCS
Negation	All Features	91.21	76.57	83.25	52.98
	POS Classifier	91.79	78.29	84.50	56.77
Speculation	**System**	**P**	**R**	**F**	**PCS**
	All Features	95.71	92.09	93.86	70.55
	POS Classifier	95.79	93.13	94.44	71.68

(Note: "All Features" System is the optimal system in Section 5.2)

Table 9. Performances of POS based classification.

Table 9 shows the performance of POS based classification. Compared with the system which only uses one classifier for all cues in Section 5.2,

the POS based classification improves 1.13% on PCS (χ^2; $p < 0.01$), as different POS kinds of cues involve respectively effective features with more related clues between cue and its scope.

Table 10 lists the performance of each POS kind of cues in speculation scope classification. There are still some low performances in some kinds of POS of cues. We consider it caused by two reasons. Firstly, some kinds of POS of cues (e.g. NN etc.) have fewer samples (just 43 samples shown in Table 5). For this reason, the training for classifier is limited. Then, for these low performance kinds of POS of cues, we may have not found the effective features for them. Although there are some kinds of cues with low performance, the whole performance of part-of-speech based classification is improved.

Cue's POS	B1~B6						CS1~CS10										DS1~DS5					PCS
	1	2	3	4	5	6	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	
CC	√				√	√				√			√			√	√	√	√		√	38.45
IN	√				√	√				√			√	√		√	√		√	√	√	87.99
JJ	√					√				√				√		√					√	31.83
MD				√	√	√		√		√				√	√	√	√	√		√	√	79.84
NN	√					√	√	√			√				√		√		√			65.83
RB						√	√	√			√	√				√					√	37.03
VB						√	√									√	√					44.29
VBD				√	√				√	√			√	√	√	√	√	√		√		63.57
VBG				√	√				√	√			√		√	√	√	√		√	√	82.89
VBN				√	√				√				√		√	√	√	√		√		66.38
VBP				√	√	√	√			√	√			√	√	√	√	√		√		81.91
VBZ				√	√	√	√			√	√			√	√	√	√	√		√		77.16

Table 10. Performance of each POS kind of cues in speculation scope classification.

5.5 Results of Comparison Experiments

To get the final performance of our approach, we train the classifiers respectively by different effective features in Section 4.1 for POS kinds of cues, and use the structured syntactic parse features in Section 4.2 on Abstracts sub-corpus by performing 10-fold cross validation.

	System	Abstract	Paper	Clinical
Negation	Morante (2008)	57.33	N/A	N/A
	Morante (2009a)	73.36	50.26	87.27
	Ours	76.90	61.19	85.31
	System	**Abstract**	**Paper**	**Clinical**
Speculation	Morante (2009b)	77.13	47.94	60.59
	Özgür (2009)	79.89	61.13	N/A
	Ours	84.21	67.24	72.92

Table 11. Performance comparison of our system with the state-of-the-art ones in PCS.

The results in Table 11 show that our system outperforms the state of the art ones both on negation and speculation scope detection. Results also show that the system is portable to different types of documents, although performance varies depending on the characteristics of the corpus.

In addition, on both negation and speculation, the results on Clinical Reports sub-corpus are better than those on Full Papers sub-corpus. It is mainly due to that the clinical reports are easier to process than full papers and abstracts. The average length of sentence for negative clinical reports is 8.19 tokens, whereas for abstracts it is 29.39 and for full papers 30.49. Shorter sentences imply shorter scopes. The more unambiguous sentence structure of short sentence can make the structured constituent and dependency syntactic features easier to be processed.

6 Conclusion

This paper proposes a new approach for tree kernel-based scope detection by using the structured syntactic parse information. In particular, we have explored the way of selecting compatible features for different part-of-speech cues. Experiments show substantial improvements of our scope classification and better robustness.

However, the results on the Full Papers and the Clinical Reports sub-corpora are lower than those on the Abstracts sub-corpus for both negation and speculation. That is because the structured syntactic parse features contain some complicated and lengthy components, and the flat features cross corpus are sparse. Our future work will focus on the pruning algorithm for the syntactic structures and analyzing errors in depth in order to get more effective features for the scope detection on different corpora.

Acknowledgments

This research is supported by the National Natural Science Foundation of China, No.61272260, No.61373097, No.61003152, the Natural Science Foundation of Jiangsu Province, No.BK2011282, the Major Project of College Natural Science Foundation of Jiangsu Province, No.11KJA520003 and the Graduates Project of Science and Innovation, No.CXZZ12_0818. Besides, thanks to Yu Hong and the three anonymous reviewers for their valuable comments on an earlier draft.

References

Emilia Apostolova, Noriko Tomuro and Dina Demner-Fushman. 2011. Automatic Extraction of Lexico-Syntactic Patterns for Detection of Negation and Speculation Scopes. In *Proceedings of ACL-HLT short papers*, pages 283-287.

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. *Journal of Biomedical Informatics*, 34 (5): 301-310.

Wanxiang Che, Min Zhang, Ting Liu and Sheng Li. 2006. A Hybrid Convolution Tree Kernel for Semantic Role Labeling. In *Proceedings of ACL*, pages 73-80.

Nigel Collier, Hyun S. Park, Norihiro Ogata, et al. 1999. The GENIA Project: Corpus-Based Knowledge Acquisition and Information Extraction from Genome Research Papers. In *Proceedings of EACL*.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text. In *Proceedings of CoNLL: Shared Task*, pages 1-12.

Ilya M. Goldin and Wendy W. Chapman. 2003. Learning to Detect Negation with 'Not' in Medical Texts. In *SIGIR Workshop: Text Analysis and Search for Bioinformatics*.

Yang Huang and Henry Lowe. 2007. A Novel Hybrid Approach to Automated Negation Detection in Clinical Radiology Reports. *Journal of the American Medical Informatics Association*, 14(3):304-311.

Zhengping Jiang and Hwee T. Ng. 2006. Semantic Role Labeling of NomBank: A Maximum Entropy Approach. In *Proceedings of EMNLP*, pages 138-145.

Roser Morante, Anthony Liekens, and Walter Daelemans. 2008. Learning the Scope of Negation in Biomedical Texts. In *Proceedings of EMNLP*, pages 715-724.

Roser Morante and Walter Daelemans. 2009a. A Metalearning Approach to Processing the Scope of Negation. In *Proceedings of CoNLL*, pages 21-29.

Roser Morante and Walter Daelemans. 2009b. Learning the Scope of Hedge Cues in Biomedical Texts. In *Proceedings of the BioNLP Workshop*, pages 28-36.

Roser Morante, Vincent Van Asch and Walter Daelemans. 2010. Memory-Based Resolution of In-Sentence Scopes of Hedge Cues. In *Proceedings of CoNLL Shared Task*, pages 40-47.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. *In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113-120.

Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2010. Syntactic Scope Resolution in Uncertainty Analysis. In *Proceedings of COLING*, pages 1379-1387.

Arzucan Özgür and Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. In *Proceedings of EMNLP*, pages 1398-1407.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of NAACL*, pages 404-411.

Liliana M. Sánchez, Baoli Li, Carl Vogel. 2007. Exploiting CCG Structures with Tree Kernels for Speculation Detection. *In Proceedings of the Fourteenth Conference on Computational Natural Language Learning: Shared Task*, pages 126-131.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: Annotation for Negation, Uncertainty and their Scope in Biomedical Texts. In *Proceedings of BioNLP*, pages 38-45.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA Corpus. In *Proceedings of IJCNLP, Companion volume*, pages 222-227.

Erik Velldal, Lilja Øvrelid, Jonathon Read and Stephan Oepen. 2012. Speculation and Negation: Rules, Rankers, and the Role of Syntax. *Computational Linguistics*, 38(2):369-410.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9(Suppl 11):S9.

Guodong Zhou, Min Zhang, Donghong Ji, and Qiaoming Zhu. 2007. Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages, 728-736.

A temporal model of text periodicities using Gaussian Processes

Daniel Preoţiuc-Pietro, Trevor Cohn
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello Street
Sheffield, S1 4DP, United Kingdom
{daniel,t.cohn}@dcs.shef.ac.uk

Abstract

Temporal variations of text are usually ignored in NLP applications. However, text use changes with time, which can affect many applications. In this paper we model periodic distributions of words over time. Focusing on hashtag frequency in Twitter, we first automatically identify the periodic patterns. We use this for regression in order to forecast the volume of a hashtag based on past data. We use Gaussian Processes, a state-of-the-art bayesian non-parametric model, with a novel periodic kernel. We demonstrate this in a text classification setting, assigning the tweet hashtag based on the rest of its text. This method shows significant improvements over competitive baselines.

1 Introduction

Temporal changes in text corpora are central to our understanding of many linguistic and social phenomena. Social Media platforms and the digitalization of libraries provides a vast body of timestamped data. This allows studying of the complex temporal patterns exhibited by text usage including highly non-stationary distributions and periodicities. However, temporal effects have been mostly ignored by previous work on text analysis or at best dealt with by making strong assumptions such as smoothly varying parameters with time (Yogatama et al., 2011) or modelled using a simple uni-modal distri bution (Wang and McCallum, 2006). This paper develops a temporal model for classifying microblog posts which explicitly incorporates multimodal periodic behaviours using Gaussian Processes (GPs).

We expect text usage to follow multiple periodicities at different scales. For example, people on Social Media might talk about different topics during and after work on weekdays, talk every Friday about the weekend ahead, or comment about their favorite weekly TV show during its air time. Given this, text frequencies will display periodic patterns. This applies to other text related quantities like co-occurrence values or topic distributions over time, as well as applications outside NLP like user behaviour (Preoţiuc-Pietro and Cohn, 2013).

Modelling temporal patterns and periodicities can be useful to tasks like text classification. For example a tweet containing 'music' is normally attributed to a general hashtag about music like #np (now playing). However, knowing time, if it occurs during the (weekly periodic) air time of 'American Idol' it is more likely for it to belong to #americanidol or if its mentioned in the days building up to the Video Music Awards to be assigned to #VMA.

In NLP, temporal models have treated time in overly simplistic ways and without regard to periodicities. We propose a model that first broadly identifies several types of temporal patterns: a) periodic, b) constant in time, c) falling out of use after enjoying a brief spell of popularity (e.g. internet memes, news). This is performed automatically only using training data and makes no assumptions on the existence or the length of the periods we aim to model. We demonstrate the approach by modelling frequencies of hashtag occurrences in Twitter. Hashtags are user-generated labels included in tweets by their authors in order to assign them to a conversation and can be considered as a proxy for topics.

To this end, we make use of Gaussian Processes (GP) (Rasmussen and Williams, 2005), a

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 977–988,
Seattle, Washington, USA, 18-21 October 2013.

Bayesian non-parametric model for regression. Using the *Bayesian evidence* we automatically perform model selection to classify temporal patterns. We aim to use the most suitable model for extrapolation, i.e. predicting future values from past observations. The GP is fully defined by the covariance structure assumed between the observed points, and its hyperparameters, which can be automatically learned from data. We also introduce a new kernel suitable to model the periodic behaviour we observe in text: periods of low frequency followed by bursts at regular time intervals. We demonstrate that the GP approach is more general and gives better results than frequentist models (e.g. autoregressive models) because it incorporates uncertainty explicitly and elegantly, in addition to automatic model selection and parameter fitting.

To demonstrate the practical importance of our approach, we use our GP prediction as a prior in a Naïve Bayes model for text classification showing improvements over baselines which do not account for temporal periodicities. Our approach extends to more general uses, e.g. to discriminative text regression and classification. More broadly, we aim to establish GPs as a state-of-the-art model for regression and classification in NLP. To our knowledge, this is the first paper to use GP regression for forecasting and model selection within a NLP task.

All the hashtag time series data and the implementation of the PS kernel in the popular open-source Gaussian Processes packages GPML[1] and GPy[2] are available on the author's website[3].

2 Related Work

Time varying text patterns have been of particular interest in topic modelling. Griffiths and Steyvers (2004) analyse evolution of topics over time, but without modelling time explicitly. Extensions that model time make different assumptions, usually regarding smoothing proprieties in (Wang and McCallum, 2006; Blei and Lafferty, 2006; Wang et al., 2008; Hennig et al., 2012). Yogatama et al. (2011) proposed a regulariser for generalised linear models that encourages local temporal smoothness.

Modelling periodicities is one of the standard applications of Gaussian Processes (Rasmussen and Williams, 2005). Recent work by Wilson and Adams (2013) and Durrande et al. (2013) show how different periods can be identified from data. In general, methods that assume certain periodicities at daily or weekly levels were proposed e.g. in (McInerney et al., 2013). GPs were used with text by Polajnar et al. (2011) and for Quality Estimation regression in (Cohn and Specia, 2013; Shah et al., 2013).

Temporal patterns for short, distinctive lexical items such as hashtags and memes were quantitatively studied (Leskovec et al., 2009) and clustered (Yang and Leskovec, 2011) in Social Media. (Yang et al., 2012) studies the dual role of hashtags, of bookmarks of content and symbols of community membership, in the context of hashtag adoption. (Romero et al., 2011) analyses the patterns of temporal diffusion in Social Media finding that hashtags have also a persistence factor.

For predicting future popularity of hashtags, Tsur and Rappoport (2012) use linear regression with a wide range of features. (Ma et al., 2012; Ma et al., 2013) frame the problem as classification into a number of fixed intervals and applies all the standard classifiers. None of these studies model periodicities, although the former stresses their importance for accurate predictions. For predicting the hashtag given the tweet text, Mazzia and Juett (2011) uses the Naïve Bayes classifier with the uniform and empirical prior or TF-IDF weighting.

3 Gaussian Processes

In this paper we consider Gaussian Process (GP) models of regression (Rasmussen and Williams, 2005). GP is a probabilistic machine learning framework incorporating kernels and Bayesian non-parametrics which is widely considered as state-of-the-art for regression. The GP defines a prior over functions which applied at each input point gives a response value. Given data, we can analytically infer the posterior distribution of these functions assuming Gaussian noise. The kernel of the GP defines the covariance in response values as a function of its inputs.

We can identify two different set-ups for a regression problem. If the range of values to be predicted

[1] http://www.gaussianprocess.org/gpml/code

[2] https://github.com/SheffieldML/GPy

[3] http://www.preotiuc.ro

lies *within* the bounds of the training set we call the prediction task as interpolation. If the range of the prediction is *outside* the bounds, then our problem that of extrapolation. In this respect, extrapolation is considered a more difficult task and the covariance kernel which incorporates our prior knowledge plays a major role in the prediction.

There is the case when multiple covariance kernels can describe our data. For choosing the right kernel and its hyperparameters only using the training data we employ Bayesian model selection which makes a trade-off between the fit of the training data and model complexity. We now briefly give an overview of GP regression, kernel choice and model selection. We refer the interested reader to (Rasmussen and Williams, 2005) for a detailed introduction to GPs.

3.1 Gaussian Process Regression

Consider a time series regression task where we only have one feature, the value x_t at time t. Our training data consists of n pairs $\mathcal{D} = \{(t, x_t)\}$. The model will need to predict values x_t for values of t greater than those in the dataset.

GP regression assumes a latent function f that is drawn from a GP prior $f(t) \sim \mathcal{GP}(m, k(t, t'))$ where m is the mean and k a kernel. The prediction value is obtained by the function evaluated at the corresponding data point, $x_t = f(t) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is white-noise. The GP is defined by the mean m, here 0, and the covariance kernel function, $k(t, t')$.

The posterior at a test point t_* is given by:

$$p(x_*|t_*, \mathcal{D}) = \int_f p(x_*|t_*, f) \cdot p(f|\mathcal{D}) \qquad (1)$$

where x_* and t_* are the test value and time. The posterior $p(f|\mathcal{D})$ shows our belief over possible functions after observing the training set $\mathcal{D}$. The predictive posterior can be solved analytically with solution:

$$\begin{aligned} x_* \sim \mathcal{N}(&k_*^T (K + \sigma_n^2 I)^{-1} \mathbf{t}, \\ &k(t_*, t_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_*) \end{aligned} \qquad (2)$$

where $k_* = [k(t_*, t_1) ... k(t_*, t_n)]^T$ are the kernel evaluations between the test point and all the training points, $K = \{k(t_i, t_j)\}_{j=1..n}^{i=1..n}$ is the Gram matrix over the training points and $\mathbf{t}$ is the vector of training points. The posterior of x_* includes the mean response as well as its variance, thus expressing the uncertainty of the prediction. In this paper, we will consider the forecast as the expected value. Due to the matrix inversion in 2, inference takes $O(n^3)$ time where n is the number of training points.

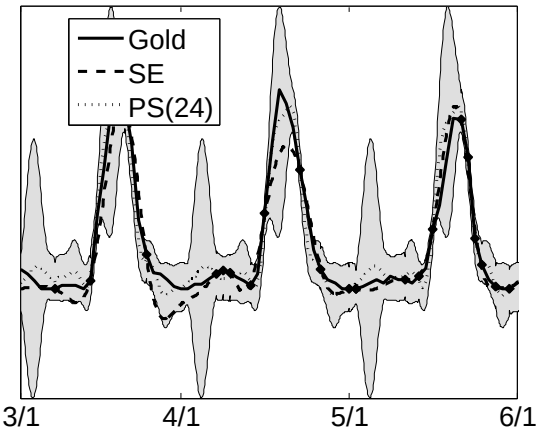


Figure 1: Interpolation for #goodmorning over 3 days with SE and PS(p=24,s=3) kernels. Prediction variance shown in grey for PS(24). Crosses represent training points.

3.2 Kernels

The covariance kernel together with its parameters fully define the GP (we assume 0 mean). The kernel induces similarities in the response between pairs of data points. Intuitively, if we want a smooth function, closer points should have high covariance compared to points that are further apart. If we want a periodic behaviour points at period length intervals should have the highest covariance. Usually, this is defined by an isotropic kernel, which means its invariant to all rigid motions.

For interpolation, a standard kernel (e.g. squared exponential) that encourages smooth functions is normally used. Figure 1 shows regression over 3 days for #goodmorning when only a random third of the values of the function are observed. We see that both the SE kernel and a periodic kernel (PS, see below) give good results.

However, for extrapolation, the choice of the kernel is paramount. The kernel encodes our prior belief about the type of function wish to learn. To illustrate this, in Figure 3, we show the time series for #goodmorning over 2 weeks and plot the regression for the

future week learned by using different kernels.

In this study we will use multiple kernels, each most suitable for a specific category of temporal patterns in our data. This includes a new kernel inspired by observed word occurrence patterns. The kernels we use are:

Constant (C): The constant kernel is $k_C(t,t') = c$. Its mean prediction will always be the value c and its assumption is that the signal is modeled only by Gaussian noise centred around this value. This describes the data best when we have a noisy signal around a stationary mean value.

Squared exponential (SE): The SE kernel or the Radial Basis Function (RBF) is the standard kernel used in most interpolation settings.

$$k_{SE}(t,t') = s^2 \cdot \exp -\frac{(t-t')^2}{2l^2} \tag{3}$$

This gives a smooth transition between neighbouring points and best describes time series with a smooth shape e.g. a uni-modal burst with a steady decrease. However, its uncertainty grows with for predictions well into the future. Its two parameters s and l are the characteristic lengthscales along the two axes. Intuitively, they control the distance of inputs on a particular axis from which the function values become uncorrelated. Using the SE kernel corresponds to Bayesian linear regression with an infinite number of basis functions (Rasmussen and Williams, 2005).

Linear (Lin): The linear kernel describes a linear relationship between outputs.

$$k_{Lin}(t,t') = s^2 + \|t \cdot t'\| \tag{4}$$

This can be obtained from linear regression by having $\mathcal{N}(0,1)$ priors on the corresponding regression weights and a prior of $\mathcal{N}(0,s^2)$ on the bias.

Periodic (PER): The periodic kernel represents a SE kernel in polar coordinates.

$$k_{PER}(t,t') = s^2 \cdot \exp -2 \cdot \left(\frac{\sin^2(2\pi(t-t')^2/p)}{l^2} \right) \tag{5}$$

It has a sinusoidal shape and is good at modelling periodically patterns that oscillate between low and high frequency. s and l are characteristic lengthscales as in the SE kernel and p is the period.

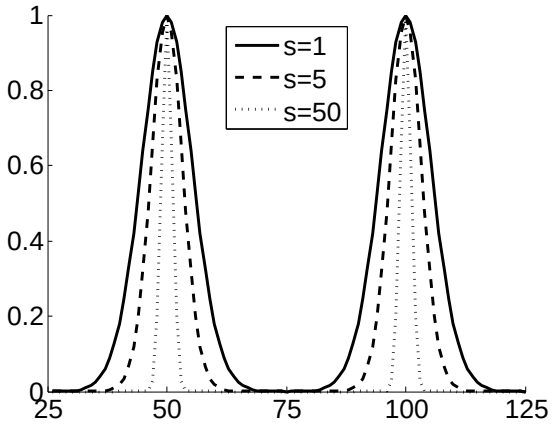


Figure 2: Behaviour of the PS kernel (p=50) with varying s. Values normalized in [0,1] interval.

Periodic spikes (PS): For textual time series, like word frequencies, we identify the following periodic behaviour: abrupt rise in usage, usually with a peak, followed by periods of low occurrence, which can be short (e.g. during the night) or long lived (e.g. the entire week except for a few hours). For modelling we introduce the following kernel:

$$k_{PS}(t,t') = \cos\left(\sin\left(\frac{2\pi \cdot (t-t')^2}{p}\right)\right) \cdot \exp\left(\frac{s\cos(2\pi \cdot (t-t')^2)}{p} - s\right) \tag{6}$$

The kernel is parameterised by its period p and a shape parameter s. The period indicates the time interval between the peaks of the function, while the shape parameter controls the *width* of the spike. The behaviour of the kernel is illustrated in Figure 2. We constrain $s \geq 1$.

In Figure 3 we see that the forecast is highly dependent on the kernel choice. We expect that for periodic data the PER and PS kernels will forecast best, maybe with the PS kernel doing a better job because it captures multiple modes of the daily increase in volume. We use for both kernels a period of 168 hours. This is because although a daily pattern exists, the weekly is stronger, with the day of the week influencing the volume of the hashtag. The NRMSE (Normalized Root Mean Square Error) in Table 1 on the held out data confirms this finding, with PS showing the lowest error.

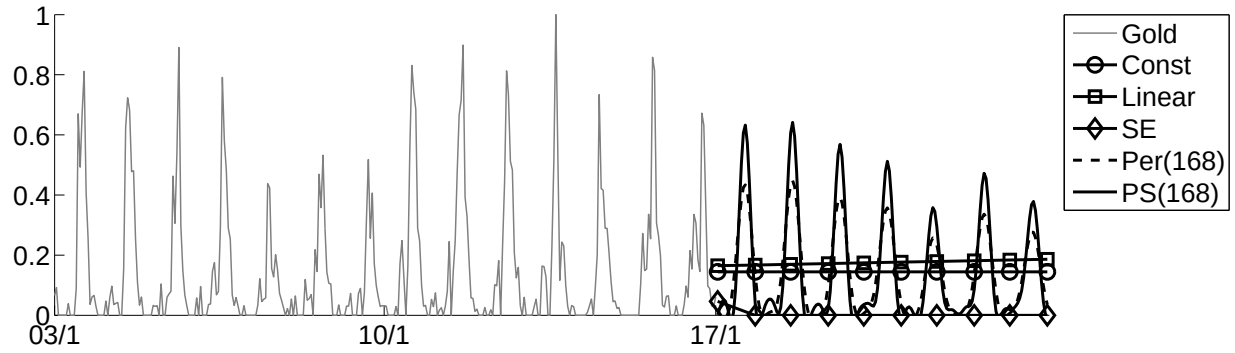


Figure 3: Extrapolation for #goodmorning over 3 weeks with GPs using different kernels.

	Const	Lin	SE	PER	PS
NLML	-41	-34	-176	-180	-192
NRMSE	0.213	0.214	0.262	0.119	0.107

Table 1: Negative Log Marginal Likelihood (NLML) shows the best fitted model for the time series in Figure 3. NRMSE computed on the third unobserved week. Lower values are better in both cases.

3.3 Model selection and optimisation

We now briefly discuss the concepts of model selection in the GP framework, by which we refer to choosing the model (kernel) from a set $\mathcal{H}_i$ and optimising the model hyperparameters θ. In our GP Bayesian inference scheme, we can compute the probability of the data given the model which involves the integral over the parameter space. This is called the *marginal likelihood* or *evidence* and is useful for model selection using only the training set:

$$p(x|\mathcal{D}, \theta, \mathcal{H}_i) = \int_f p(x|\mathcal{D}, f, \mathcal{H}_i) p(f|\theta, \mathcal{H}_i) \quad (7)$$

Our first goal is to fit the kernel by minimizing the negative log marginal likelihood (NLML) with respect to the kernel parameters θ. This approximation is also known as type II maximum likelihood (ML-II). Conditioned on kernel parameters, the evidence of a GP can be computed analytically.

Our second goal is to use the evidence for model selection because it balances the data fit and the model complexity by automatically incorporating Occam's Razor (Rasmussen and Ghahramani, 2000). Because the evidence must normalise, complex models which can account for many datasets achieve low evidence. One can think of the evidence as the probability that a random draw of the parameter values from the model class would generate the dataset $\mathcal{D}$. This way, complex models are penalised because they can describe many datasets, while the simple models can describe only a few datasets, thus the chance of a good data fit being very low. This is for example the case of the periodic bursts in Figure 3. Although the periodic kernel can fit the data, it will incur a high model complexity penalty. The PS kernel in this respect is a simpler model and can fit the data and is thus chosen as the right model.

When the dataset is observed, the evidence can select between the models. More generally, the model choice actually gives us an *implicit classification* of the temporal patterns into classes: a steady signal with noise (C kernel), a signal with local temporal patterns (SE kernel), an oscilating periodic pattern (PER kernel) or a pattern with abrupt periodic peaks (PS kernel).

We use the NLML for optimising the hyperparameters only using training data. For optimising the hyperparameters of the kernel defined in Equation 6, it is important to first identify the right period. We consider as possible periods all integer values less than half the size of the training set, and then tune the shape parameter using gradient descent to minimise NLML. We then take the $argmin$ value of those considered. We show the NLML for a sample regression in Figure 4.

The likelihood shows that there are multiple canyons in the likelihood, which can lead a convex optimisation method to local optima. These appear when p is equal or an integer multiple of the main period of the data, in this case 24. The lowest values are obtained when $p = 168$, allowing the model to accommodate the day of week effect. Our procedure is not guaranteed to reach a global optima, but

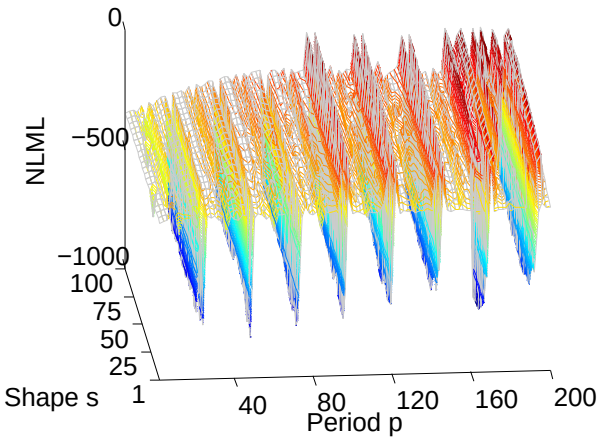


Figure 4: NLML for #goodmorning on the training set as a function of the 2 kernel parameters.

is a relatively standard technique for fitting periodic kernels (Duvenaud et al., 2013).

The flexibility of the GP framework allows us to combine kernels (e.g. $SE \cdot PS$ or $PS + Lin$) in order to identify a combination of trends (Duvenaud et al., 2013; Gönen and Alpaydin, 2011). Experiments on a subset of data showed no major benefits of combining kernels, but the computational time and model complexity increased drastically due to the extra hyperparameters. Because we will model a proportion of words within a limited time frame, there are few linear trends in the data. It might seem limiting that we only learn a single period, although we could combine periodic kernels with different periods together. But, as we have seen in the #goodmorning example (with overlapping weekly and daily patterns), if there is a combination of periods the model will select a single period which is the least common multiple.

4 Data

For our experiments we used data collected from Twitter using the public Gardenhose stream (10% representative sample of the entire Twitter stream). The data collection interval was 1 January – 28 February 2011. For simplicity in the classification task, we filtered the stream to include only tweets that have exactly one hashtag. These represent approximately 7.8% of our stream.

As text processing steps, we have tokenised all the tweets and filtered them to be written in English using the Trendminer pipeline (Preoţiuc-Pietro et al., 2012). We also remove duplicate tweets (retweets and tweets that had the same first 6 content tokens) because they likely represent duplicate content, automated messages or spam which would bias the dataset, as also stated by Tsur and Rappoport (2012). In our experiments we use the first month of data as training and the second month as testing. Note the challenging nature of this testing configuration where predictions must be made for up to 28 days into the future. We keep a total 1176 of hashtags which appear at least 500 times in both splits of the data. The vocabulary consists of all the tokens that occur more than 100 times in the dataset and start with an alphabetic letter. After processing, our dataset consists of 6,416,591 tweets with each having on average 9.55 tokens.

5 Forecasting hashtag frequency

We treat our task of forecasting the volume of a Twitter hashtag as a regression problem. Because the total number of tweets varies depending on the day and hour of day, we chose to model the proportion of tweets with the given tag in that hour. Given a time series of these values as the training set for a hashtag, we aim to predict the values in the testing set, extrapolating to the subsequent month.

Hashtags represent free-form text labels that authors add to a tweet in order to enable other users to search them to participate in a conversation. Some users use hashtags as regular words that are integral to the tweet text, some hashtags are general and refer to the same thing or emotion (#news, #usa, #fail), others are Twitter games or memes (#2010dissapointments, #musicmonday). Other hashtags refer to events which might be short lived (#worldcup2022), long lived (#25jan) or periodic (#raw, #americanidol). We chose to model hashtags because they group similar tweets (like topics), reflect real world events (some of which are periodic) and present direct means of evaluation. Note that this approach could be applied to many other temporal problems in NLP or other domains. We treat each regression problem independently, learning for each hashtag its specific model and set of parameters.

5.1 Methods

We choose multiple baselines for our prediction task in order to compare the effectiveness of our ap-

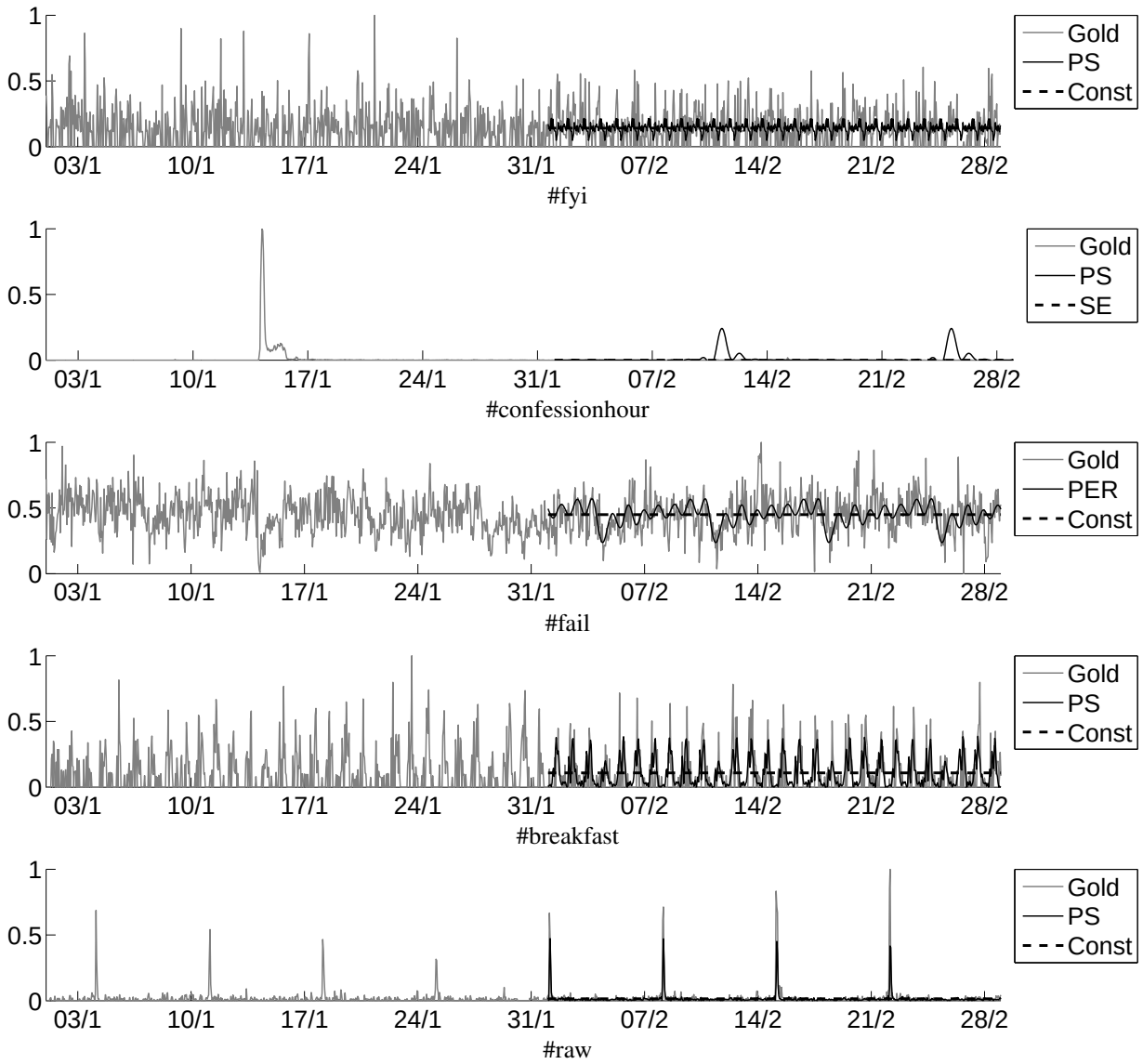


Figure 5: Sample regressions and their fit using different methods.

proach. These are:

Mean value (M): We use as prediction the mean of the values in the training set. Note that this is the same as using a GP model with a constant kernel (+ noise) with a mean equal to the training set mean.

Lag model with GP determined period (Lag+): The prediction is the mean value in the training set of the values at lag Δ where Δ is the period rounded to the closest integer as determined by our GP model. This is somewhat similar to an autoregressive (AR) model with all the coefficients except Δ set to 0. We highlight that given the period Δ this is a very strong model as it gives a mean estimate at each point. Comparing to this model we can see if the GP model can recover the underlying function that described the periodic variation and filter out the noise in the observations. Correctly identifying the period is very challenging as we discuss below.

GP regression: Gaussian Process regression using only the SE kernel (**GP-SE**), the periodic kernel (**GP-PER**), the PS kernel (**GP-PS**). The method that chooses between kernels using model selection as described in Section 3.3 is denoted as **GP+**. We will also compare to GP regression the linear kernel **(GP-Lin)**, but we will not use this as a candidate for model selection due the poor results shown below.

Hashtag	Lag(p)	Const		SE		PER		PS	
	NRMSE	NLML	NRMSE	NLML	NRMSE	NLML	NRMSE	NLML	NRMSE
#fyi	0.1578	**-322**	**0.1404**	-320	0.1898	-321	0.1405	-293	0.1456
#confessionhour	0.0404	-85	0.0107	**-186**	**0.0012**	-90	0.0327	-88	0.0440
#fail	0.1431	-376	0.1473	-395	0.4695	**-444**	**0.1387**	-424	0.1390
#breakfast	0.1363	-293	0.1508	-333	0.1773	-293	0.1514	**-367**	**0.1276**
#raw	0.0464	-1208	0.0863	-1208	0.0863	-1323	0.0668	**-1412**	**0.0454**

Table 2: NRMSE shows the best performance for forecasting and NLML shows the best model for all the regressions in Figure 5. Lower is better.

5.2 Results

We start by qualitatively analysing a few sample regressions that are representative of each category of time series under study. These are shown in Figure 5. For clarity, we only plotted a few kernels on each figure. The full evaluation statistics in NRMSE and the Bayesian evidence are show in Table 2.

For the hashtag #fyi there is no clear pattern. For this reason the model that uses the constant kernel performs best, being the simplest one that can describe the data, although the others give similar results in terms of NRMSE on the held-out testing set. While functions learned using this kernel never clearly outperform others on NRMSE on held-out data, this is very useful for interpretation of the time series, separating noisy time series from those that have an underlying periodic behaviour.

The #confessionhour example illustrates a behaviour best suited for modelling using the SE kernel. We notice a sudden burst in volume which decays over the next 2 days. This is actually the behaviour typical of 'internet memes' (this hashtag tags tweets of people posting things they would never tell anyone) as presented in Yang and Leskovec (2011). These cannot be modelled with a constant kernel or a periodic one as shown by the results on held-out data and the time series plot. The periodic kernels will fail in trying to match the large burst with others in the training data and will attribute to noise the lack of a similar peak, thus discovering wrong periods and making bad predictions. In this example, forecasts will be very close to 0 under the SE kernel, which is what we would desire from the model.

The periodic kernel best models hashtags that exhibit an oscillating pattern. For example, this best fits words that are used frequently during the day and less so during the night, like #fail. Here, the period is chosen to be one week (168) rather than one day (24) because of the weekly effect superimposed on the daily one. Our model recovers that there is a daily pattern with people tweeting about their or others' failures during the day. On weekends however, and especially on Friday evenings, people have better things to do.

The PS kernel introduced in this paper models best hashtags that have a large and short lived burst in usage. We show this by two examples. First, we choose #breakfast which has a daily and weekly pattern. As we would expect, a big rise in usage occurs during the early hours of the day, with very few occurrences at other times. Our model discovers a weekly pattern as well. This is used mainly for modelling the difference between weekends and weekdays. On weekends, the breakfast tag is more evenly spread during the hours of the morning, because people do not have to wake up for work and can have breakfast at a more flexible time than during the week. In the second example, we present a hashtag that is associated to a weekly event: #raw is used to discuss a wrestling show that airs every week for 2 hours on Monday evenings in the U.S.. With the exception of these 2 hours and the hour building up to it, the hashtag is rarely used. This behaviour is modelled very well using our kernel, with a very high value for the shape parameter ($s = 200$) compared to the previous example ($s = 11$) which captures the abrupt trend in usage. In all cases, our GP model chosen by the evidence performs better than the Lag+ model, which is a very strong method if presented with the correct period. This further demonstrates the power of the Gaussian Process framework to deal with noise in the training data and to find the underlying function of the time variation of words.

In Table 3 we present sample tags identified as

Const	SE	PER	PS
#funny	#2011	#brb	#ff
#lego	#backintheday	#coffee	#followfriday
#likeaboss	#confessionhour	#facebook	#goodnight
#money	#februarywish	#facepalm	#jobs
#nbd	#haiti	#funny	#news
#nf	#makeachange	#love	#nowplaying
#notetoself	#questionsidontlike	#rock	#tgif
#priorities	#savelibraries	#running	#twitterafterdark
#social	#snow	#xbox	#twitteroff
#true	#snowday	#youtube	#ww
49	**268**	**493**	**366**

Table 3: Sample hashtags for each category. The last line shows the total number of hashtags of each type.

Lag+	GP-Lin	GP-SE	GP-PER	GP-PS	GP+
7.29%	-3.99%	-34.5%	0.22%	7.37%	**9.22%**

Table 4: Average relative gain over mean (M) prediction for forecasting on the entire month using the different models

being part of the 4 hashtag categories, and the total number of hashtags in each.

As a means of quantitative evaluation we compute the relative NRMSE compared to the Mean (M) method for forecasting. We choose this, because we consider that NRMSE is not comparable between regression tasks due to the presence of large peaks in many time series, which distort the NRMSE values. The results are presented in Table 4 and show that our Gaussian Process model using model selection is best. Remarkably, it consistently outperforms the Lag+ model, which shows the effectiveness of the GP models to incorporate uncertainty. The GP-PS model does very well on its own. Although chosen in the model selection phase in only a third of the tasks, it performs consistently well across tasks because of its ability to model well all the periodic hashtags, be they smooth or abrupt. The GP-Lin model does worse than the average, mostly due to uni-modal time series which don't have high occurrences in the testing part of the data.

5.3 Discussion

Let us now turn to why the GP model is better for discovering periodicities than classic time series modelling methods. Measuring autocorrelation between points in the time series is used to discover the hidden periodicities in the data and in building AR models. However, the downsides of this method are: a) the incapacity of accurately finding the correct periods, because all integer multiples of the correct period will be feasible candidates and b) it leads to incorrect conclusions when there is autocorrelated noise. The second case is illustrated in Figure 6 where #confessionhour shows autocorrelation but, as seen in Figure 5, lacks a periodic component.

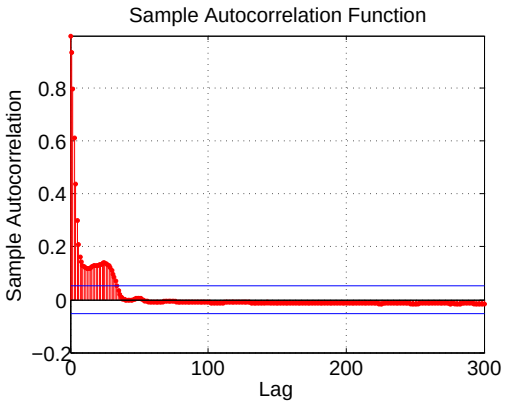


Figure 6: Sample autocorrelation for #confessionhour

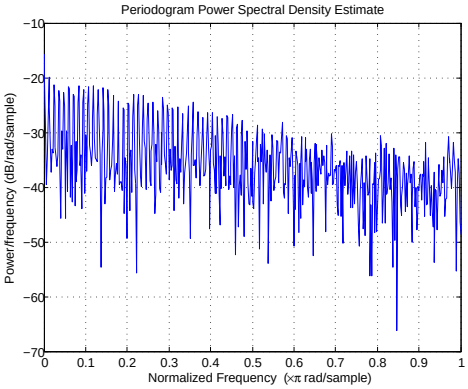


Figure 7: Power spectral density for #raw

Another approach to discovering periods in data is by computing the power spectral density. This has been used in the GP framework by Wilson and Adams (2013). For some time series, this gives a good indication of the period, as represented by a peak in the periodogram at that value. This fails to discover the correct period when dealing with large bursts like those exhibited by the #raw time series as shown in Figure 7. The lowest frequency spike corresponds to the correct period of 168, but also other candidate periods are shown as possible. The reason for this is its reliance on the Fourier Transform which decomposes the time series into a sum of oscillating patterns. These cannot model step-functions and other non-smoothly varying signals. A further discussion falls out of the scope and space constraints of this paper.

Tweet	Time	Prior	Rank	Prediction
Bruins Goal!!! Patrice Bergeron makes it 3-1 Boston	2-3am, 2 Feb 2011	E: 0.00017 P: 0.00086	7 1	#fb #bruins
i need some of Malik people	3-4am, 2 Feb 2011	E: 0.00021 P: 0.00420	7 1	#ff #thegame
Alfie u doughnut! U didn't confront Kay? SMH	7-8pm, 3 Feb 2011	E: 0.00027 P: 0.00360	8 1	#nowplaying #eastenders

Table 5: Example of tweet classification using the Naïve Bayes model with the two different priors (E - empirical, P - GP forecast). Rank shows the rank in probability of the correct class (hashtag) under the model. Time is G.M.T.

6 Text based prediction

In this section we demonstrate the usefulness of our method of modelling in an NLP task: predicting the hashtag of a tweet based on its text. In contrast to this classification approach for suggesting a tweet's hashtag, information retrieval methods based on computing similarities between tweets are very hard to scale to large data (Zangerle et al., 2011).

We choose a simple model for prediction, the Naïve Bayes Classifier. This method provides us with a straightforward way to incorporate our prior knowledge of how frequent a hashtag is in a certain time frame. This Naïve Bayes model (**NB-P**) uses the forecasted values for the respective hour as the prior on the hashtags.

For comparison we use the Most Frequent (**MF**) baseline and the Naïve Bayes with empirical prior (**NB-E**) which doesn't use any temporal forecasting information. Because there are more than 1000 possible classes we show the accuracy of the correct hashtag being amongst the top 1,5 or 50 hashtags as well as the Mean Reciprocal Rank (MRR). The results are shown in Table 6.

The results show that incorporating the forecasted values as a more informative prior for classification we obtain better predictions. The improvements are consistent in all the Match values. Also, we highlight that a 9% improvement in the forecasting task carries over to about a 2% improvement in classification. We show a few examples in which the GP learned prior makes a difference in classification in Table 5 together with the values for both priors.

With these experiments, we highlighted that there are performance gains even with only adding a more informative prior that uses periodicity information. This motivates future work to add this information to discriminative classifiers thus avoiding the need for the Naïve Bayes decomposition. The modelling framework offered by the GPs can accommodate classification, although scaling issues arise when using a large number of features or output classes. Efforts to scale GPs to a large number of variables are well understood (Candela and Rasmussen, 2005) and we will try to incorporate this in future work.

	MF	NB-E	NB-P
Match@1	7.28%	16.04%	**17.39%**
Match@5	19.90%	29.51%	**31.91%**
Match@50	44.92%	59.17%	**60.85%**
MRR	0.144	0.237	**0.252**

Table 6: Results for hashtag classification.

7 Conclusion

Periodicities play an important role when analysing the temporal dimension of text. We have presented a framework based on Gaussian Process regression for identifying periodic patterns and their parameters using only training data. We divided the periodic patterns into 2 categories: oscillating and periodic bursts by performing model selection using bayesian evidence. The periodicities we have discovered have proven useful in an NLP classification task.

In future work, we aim to model time continuously and to perform discriminative clustering in order to make better use of the learned periodicites. We will consider incorporating periodicities in other applications, such as topic models.

Acknowledgements

This research was funded by the Trendminer project, EU FP7-ICT Programme, grant agreement no.287863. The authors would like to thank James Hensman, Nicolas Durrande and Neil Lawrence for advice on Gaussian Processes, Chris Dyer and Noah Smith for discussions about periodicities in NLP.

References

David Blei and John Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd International conference on Machine learning*, ICML '06.

Joaquin Quiñonero Candela and Carl Edward Rasmussen. 2005. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research (JMLR)*, 6:1939–1959, December.

Trevor Cohn and Lucia Specia. 2013. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Proceedings of the Association of Computational Linguistics*, ACL '13.

Nicolas Durrande, James Hensman, Magnus Rattray, and Neil Lawrence. 2013. Gaussian Process models for periodicity detection. In *Submitted to JRSSb, http://arxiv.org/abs/1303.7090*.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. 2013. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the International Conference on Machine Learning*, ICML '13.

Mehmet Gönen and Ethem Alpaydin. 2011. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research (JMLR)*, 12:2211–2268, July.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, (Suppl 1):5228–5235, April.

Philipp Hennig, David H. Stern, Ralf Herbrich, and Thore Graepel. 2012. Kernel topic models. *Journal of Machine Learning Research (JMLR) - Proceedings Track*, 22:511–519.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International conference on Knowledge discovery and data mining*, KDD '09.

Zongyang Ma, Aixin Sun, and Gao Cong. 2012. Will this #hashtag be popular tomorrow? In *Proceedings of the 35th International ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12.

Zongyang Ma, Aixin Sun, and Gao Cong. 2013. On predicting the popularity of newly emerging hashtags in Twitter. *Journal of the American Society for Information Science and Technology*, 64(7):1399–1410.

Allie Mazzia and James Juett. 2011. Suggesting hashtags on Twitter. In *http://www-personal.umich.edu/ amazzia/pubs/545-final.pdf*.

James McInerney, Alex Rogers, and Nicholas R Jennings. 2013. Learning periodic human behaviour models from sparse data for crowdsourcing aid delivery in developing countries. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI '13.

Tamara Polajnar, Simon Rogers, and Mark Girolami. 2011. Protein interaction detection in sentences via Gaussian Processes: a preliminary evaluation. *International Journal Data Mining and Bioinformatics*, 5(1):52–72, February.

Daniel Preoţiuc-Pietro and Trevor Cohn. 2013. Mining User Behaviours: A Study of Check-in Patterns in Location Based Social Networks. In *Proceedings of the ACM Web Science Conference*, Web Science '13.

Daniel Preoţiuc-Pietro, Sina Samangooei, Trevor Cohn, Nicholas Gibbins, and Mahesan Niranjan. 2012. Trendminer: An architecture for real time analysis of social media text. *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media, Workshop on Real-Time Analysis and Mining of Social Streams*.

Carl Edward Rasmussen and Zoubin Ghahramani. 2000. Occam's razor. In *Advances in Neural Information Processing Systems*, NIPS 13.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. MIT Press.

Daniel M. Romero, Brendan Meeder, and Jon Kleinberg. 2011. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on Twitter. In *Proceedings of the 20th International conference on World wide web*, WWW '11.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. An Investigation on the Effectiveness of Features for Translation Quality Estimation. In *MT Summit '13*.

Oren Tsur and Ari Rappoport. 2012. What's in a hashtag? Content based prediction of the spread of ideas in microblogging communities. In *Proceedings of the fifth ACM International conference on Web search and data mining*, WSDM '12.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International conference on Knowledge discovery and data mining*, KDD '06.

Chong Wang, David M. Blei, and David Heckerman. 2008. Continuous time Dynamic topic models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI '08.

Andrew Gordon Wilson and Ryan Prescott Adams. 2013. Gaussian Process covariance kernels for pattern dis-

covery and extrapolation. In *Proceedings of the International Conference on Machine Learning*, ICML '13.

Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM International conference on Web search and data mining*, WSDM '11.

Lei Yang, Tao Sun, Ming Zhang, and Qiaozhu Mei. 2012. We know what @you #tag: does the dual role affect hashtag adoption? In *Proceedings of the 21st International conference on World Wide Web*, WWW '12.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R. Routledge, and Noah A. Smith. 2011. Predicting a scientific community's response to an article. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Eva Zangerle, Wolfgang Gassler, and Gunther Specht. 2011. Recommending #-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web*, UMAP '11.

Automatically Detecting and Attributing Indirect Quotations

Silvia Pareti[◇*] **Tim O'Keefe**[†*] **Ioannis Konstas**[◇] **James R. Curran**[†] **Irena Koprinska**[†]

[◇]ILCC, School of Informatics
University of Edinburgh
United Kingdom
{s.pareti,i.konstas}@sms.ed.ac.uk

[†]ə-lab, School of IT
University of Sydney
NSW 2006, Australia
{tokeefe,james,irena}@it.usyd.edu.au

Abstract

Direct quotations are used for opinion mining and information extraction as they have an easy to extract span and they can be attributed to a speaker with high accuracy. However, simply focusing on direct quotations ignores around half of all reported speech, which is in the form of indirect or mixed speech. This work presents the first large-scale experiments in indirect and mixed quotation extraction and attribution. We propose two methods of extracting all quote types from news articles and evaluate them on two large annotated corpora, one of which is a contribution of this work. We further show that direct quotation attribution methods can be successfully applied to indirect and mixed quotation attribution.

1 Introduction

Quotations are crucial carriers of information, particularly in news texts, with up to 90% of sentences in some articles being reported speech (Bergler et al., 2004). Reported speech is a carrier of evidence and factuality (Bergler, 1992; Saurí and Pustejovsky, 2009), and as such, text mining applications use quotations to summarise, organise and validate information. Extraction of quotations is also relevant to researchers interested in media monitoring.

Most quotation attribution studies (Pouliquen et al., 2007; Glass and Bangay, 2007; Elson and McKeown, 2010) thus far have limited their scope to direct quotations (Ex.1a), as they are delimited by quotation marks, which makes them easy to extract. However, annotated resources suggest that direct quotations represent only a limited portion of all quotations, i.e., around 30% in the Penn Attribution Relation Corpus (PARC), which covers Wall Street Journal articles, and 52% in the Sydney Morning Herald Corpus (SMHC), with the remainder being indirect (Ex.1c) or mixed (Ex.1b) quotations. Retrieving only direct quotations can miss key content that can change the interpretation of the quotation (Ex. 1b) and will entirely miss indirect quotations.

(1) a. *"For 10 million, you can move $100 million of stocks,"* a specialist on the Big Board gripes. *"That gives futures traders a lot more power."*
b. *Police would only apply for the restrictions when "we have a lot of evidence that late-night noise... is disturbing the residents of that neighbourhood"*, Superintendent Tony Cooke said.
c. Mr Walsh said *Rio was continuing to hold discussions with its customers to arrive at a mutually agreed price*.

Previous work on extracting indirect and mixed quotations has suffered from a lack of large-scale data, and has instead used hand-crafted lexica of reporting verbs with rule-based approaches. The lack of data has also made comparing the relative merit of these approaches difficult, as existing evaluations are small-scale and do not compare multiple methods on the same data.

In this work we address this lack of clear, comparable results by evaluating two baseline meth-

*These authors contributed equally to this work.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 989–999,
Seattle, Washington, USA, 18-21 October 2013.

	Method	Language	Test Size (quotations)	Results P	R
Krestel et al. (2008)	hand-built grammar	English	133	74%	99%
Sarmento and Nunes (2009)	patterns over text	Portuguese	570	88%	5%[1]
Fernandes et al. (2011)	ML and regex	Portuguese	205	64%[2]	67%[2]
de La Clergerie et al. (2011)	patterns over parse	French	40	87%	70%
Schneider et al. (2010)	hand-built grammar	English	N/D	56%[2]	52%[2]

Table 1: Related work on direct, indirect and mixed quotation extraction. Note that they are not directly comparable as they apply to different languages and greatly differ in evaluation style and size of test set. [1] Figure estimated by the authors for extracting 570 quotations from 26k articles. [2] Results are for quotation extraction and attribution jointly.

ods against both a token-based approach that uses a Conditional Random Field (CRF) to predict IOB labels, and a maximum entropy classifier that predicts whether parse nodes are quotations or not. We evaluate these approaches on two large-scale corpora from the news domain that together include over 18,000 quotations. One of these corpora (SMHC) is a a contribution of this work, while our results are the first presented on the other corpus (PARC). Instead of relying on a lexicon of reporting verbs, we develop a classifier to detect verbs introducing a quotation. To inform future research we present results for direct, indirect, and mixed quotations, as well as overall results.

Finally, we use the direct quotation attribution methods described in O'Keefe et al. (2012) and show that they can be successfully applied to indirect and mixed quotations, albeit with lower accuracy. This leads us to conclude that attributing indirect and mixed quotations to speakers is harder than attributing direct quotations.

With this work, we set a new state of the art in quotation extraction. We expect that the main contribution of this work will be that future methods can be evaluated in a comparable way, so that the relative merit of various approaches can be determined.

2 Background

Pareti (2012) defines an attribution as having a *source* span, a *cue* span, and a *content* span:

Source is the span of text that indicates who the content is attributed to, e.g. 'president Obama', 'analysts', 'China', 'she'.

Cue is the lexical anchor of the attribution relation, usually a verb, e.g. 'say', 'add', 'quip'.

Content is the span of text that is attributed.

Based on the type of attitude the source expresses towards a proposition or eventuality, attributions are subcategorised (Prasad et al., 2006) into *assertions* (Ex.2a) and *beliefs* (Ex.2b), which imply different degrees of commitment, *facts* (Ex.2c), expressing evaluation or knowledge, and *eventualities* (Ex.2d), expressing intention or attitude.

(2) a. Mr Abbott said *that he will win the election.*
b. Mr Abbott thinks *he will win the election.*
c. Mr Abbott knew *that Gillard was in Sydney.*
d. Mr Abbott agreed *to the public sector cuts.*

Only assertion attributions necessarily imply a speech act. Their *content* corresponds to a quotation span and their *source* is generally referred to in the literature as the *speaker*. Direct, indirect and mixed quotations differ in the degree of factuality they entail, since the former are by convention interpreted as a verbatim transcription of an utterance whereas indirect and the non-quoted portion of mixed quotations can be paraphrased forms of the original wording, and are thus filtered by the writer's perspective.

The first speaker attribution systems (Zhang et al., 2003; Mamede and Chaleira, 2004; Glass and Bangay, 2007) originate from the narrative domain and were concerned with the identification of different characters for speech synthesis applications. Direct quotation attribution, with direct quotations being given or extracted heuristically, has been the focus of further studies in both the narrative (Elson and McKeown, 2010) and news (Pouliquen et al., 2007; Liang et al., 2010) domains. The few studies that

have addressed the extraction and attribution of indirect and mixed quotations are discussed below.

Krestel et al. (2008) developed a quotation extraction and attribution system that combines a lexicon of 53 common reporting verbs and a hand-built grammar to detect constructions that match 6 general lexical patterns. They evaluate their work on 7 articles from the Wall Street Journal, which contain 133 quotations, achieving macro-averaged Precision (P) of 99% and Recall (R) of 74% for quotation span detection. PICTOR (Schneider et al., 2010) relies instead on a context-free grammar for the extraction and attribution of quotations. PICTOR yielded 75% P and 86% R in terms of words correctly ascribed to a quotation or speaker, while it achieved 56% P and 52% R when measured in terms of completely correct quotation-speaker pairs.

SAPIENS (de La Clergerie et al., 2011) extracts quotations from French news, by using a lexicon of reporting verbs and syntactic patterns to extract the complement of a reporting verb as the quotation span and its subject as the source. They evaluated 40 randomly sampled quotations and found that their system made 32 predictions and correctly identified the span in 28 of the 40 cases. Verbatim (Sarmento and Nunes, 2009) extracts quotations from Portuguese news feeds by first finding one of 35 speech verbs and then matching the sentence to one of 19 patterns. Their manual evaluation shows that 11.9% of the quotations Verbatim finds are errors and that the system identifies approximately one distinct quotation for every 46 news articles.

The system presented by Fernandes et al. (2011) also works over Portuguese news. Their work is the closest to ours as they partially apply supervised machine learning to quotation extraction. Their work introduces GloboQuotes, a corpus of 685 news items containing 1,007 quotations of which 802 were used to train an Entropy Guided Transformation Learning (ETL) algorithm (dos Santos and Milidiú, 2009). They treat quotation extraction as an IOB labelling task, where they use ETL with POS and NE features to identify the beginning of a quotation, while the inside and outside labels are found using regular expressions. Finally they use ETL to attribute quotations to their source. The overall system achieves 64% P and 67% R.

We have summarised these approaches in Table 1, which shows that the majority of evaluations thus far have been small-scale. Furthermore, the published results do not include any comparisons with previous work, which prevents a quantitative comparison of the approaches, and they do not include results broken down by whether the quotation is direct, indirect, or mixed. It is these issues that motivate our work.

	SMHC		PARC	
	Corpus	Doc	Corpus	Doc
Docs	965	-	2,280	-
Tokens	601k	623.3	1,139k	499.9
Quotations	7,991	8.3	10,526	4.6
Direct	4,204	4.4	3,262	1.4
Indirect	2,930	3.0	5,715	2.5
Mixed	857	0.9	1,549	0.6

Table 2: Comparison of the SMHC and PARC corpora, reporting their document and token size and per-type occurrence of quotations overall and per document (average).

3 Corpora

We perform our experiments over two large corpora from the news domain.

3.1 Penn Attribution Relations Corpus (PARC)

Our first corpus (Pareti, 2012), which we will refer to as PARC, is a semi-automatically built extension to the attribution annotations included in the PDTB (Prasad et al., 2008). The corpus covers 2,280 Wall Street Journal articles and contains annotations of assertions, beliefs, facts, and eventualities, which are altogether referred to as attribution relations (ARs). For this work we use only the assertions, as they correspond to quotations (direct, indirect and mixed). The drawback of this corpus is that it is not yet fully annotated, i.e., it comprises positive and unlabelled data.

The corpus includes a test set of 14 articles that are fully annotated, which enables us to properly evaluate our work and estimate that a proportion of 30-50% of ARs are unlabelled in the rest of the corpus. The test set was manually annotated by two expert annotators. The annotators identified 491 ARs, of which 22% were nested within another AR, with

an agreement score of 87%[1]. The agreement for the selection of the content and source spans of commonly annotated ARs was 95% and 94% respectively. In this work we address only non-embedded assertions, so the final test-set includes 267 quotes, totalling 321 non-discontinuous gold spans.

3.2 Sydney Morning Herald Corpus (SMHC)

We based our second corpus on the existing annotations of direct quotations within Sydney Morning Herald articles presented in O'Keefe et al. (2012). In that work we defined direct quotations as any text between quotation marks, which included the directly-quoted portion of mixed quotations, as well as scare quotes. Under that definition direct quotations could be automatically extracted with very high accuracy, so annotations in that work were over the automatically extracted direct quotations. As part of this work one annotator removed scare quotes, updated mixed quotations to include both the directly and indirectly quoted portions, and added whole new indirect quotations. The annotation scheme was developed to be comparable to the scheme used in the PARC corpus (Pareti, 2012), although the SMHC corpus only includes assertions and does not annotate the lexical *cue*.

The resulting corpus contains 7,991 quotations taken from 965 articles from the 2009 Sydney Morning Herald (we refer to this corpus as SMHC). The annotations in this corpus also include the speakers of the quotations, as well as gold standard Named Entities (NEs). We use 60% of this corpus as training data (4,872 quotations), 10% as development data (759 quotations), and 30% as test data (2,360 quotations). Early experiments were conducted over the development data, while the final results were trained on both the training and development sets and were tested on the unseen test data.

3.3 Comparison

Table 2 shows a comparison of the two corpora and the quotations annotated within them. SMHC has a higher density of quotations per document, 8.3 vs. 4.6 in PARC, since articles are fully annotated and were selected to contain at least one quotation. PARC is instead only partially annotated and comprises articles with no quotations. Excluding null-quotation articles from PARC, the average incidence of annotated quotations per article raises to 7.1. The corpora also differ in quotation type distribution, with direct quotations being largely predominant in SMHC while indirect are more common in PARC.

[1]The agreement was calculated using the *agr* metric described in Wiebe and Riloff (2005) as the proportion of commonly annotated ARs with respect to the ARs identified overall by Annotator A and Annotator B respectively

	P	R	F
B_{say}	94.4	43.5	59.5
B_{list}	75.4	71.1	73.2
k-NN	88.9	72.6	79.9

Table 3: Results for the k-NN verb-cue classifier. B_{say} classifies as verb-cue all instances of say while B_{list} marks as verb-cues all verbs from a pre-compiled list in Krestel et al. (2008).

4 Experimental Setup

4.1 Quotation Extraction

Quotation extraction is the task of extracting the *content span* of all of the direct, indirect, and mixed quotations within a given document. More precisely, we consider quotations to be acts of communication, which correspond to *assertions* in Pareti (2012). Some quotations have content spans that are split into separate, non-adjacent spans, as in example (1a). Ideally the latter span should be marked as a continuation of a quotation, however we consider this to be out of scope for this work, so we treat each span as a separate quotation.

4.2 Preprocessing

As a pre-processing step, both corpora were tokenised and POS tagged, and the potential speakers anonymised to prevent over-fitting. We used the Stanford factored parser (Klein and Manning, 2002) to retrieve both the Stanford dependencies and the phrase structure parse. Quotation marks were normalised to a single character, as the quotation direction is often incorrect for multi-paragraph quotations.

4.3 Verb-cue Classifier

Verbs are by far the most common introducer of a quotation. In PARC verbs account for 96% of all

cues, the prepositional phrase *according to* for 3%, with the remaining 1% being nouns, adverbials and prepositional groups. Attributional verbs are not a closed set, they can vary across styles and genres, and their attributional use is highly dependent on the context in which they occur. It is therefore not possible to simply rely on a pre-compiled list of common speech verbs. Quotations in PARC are introduced by 232 verb types, 87 of which are unique occurrences. Not all of the verbs are speech verbs, for example *add*, which is the second most frequent after *say*, or the manner verb *gripe* (Ex.1a).

We used the attributional cues in the PARC corpus to develop a separate component of our system to identify attribution verb-cues. The classifier predicts whether the head of each verb group is a verb-cue using the k-nearest neighbour (k-NN) algorithm, with k equal to 3. The classifier uses 20 feature types, including:

- Lexical (e.g. token, lemma, adjacent tokens)
- VerbNet classes membership
- Syntactic (e.g. node-depth in the sentence, parent and sibling nodes)
- Sentence features (e.g. distance from sentence start/end, within quotation markers).

We compared the system to one baseline, B_{say}, that marks every instance of *say* as a verb-cue, and another, B_{list}, that marks every instance of a verb that is on the list of 53 verbs presented in Krestel et al. (2008). We tested the system on the test set for PARC, which contains 1809 potential verb-cues, of which 354 are positive and 1455 are negative.

The results in Table 3 show that the verb-cue classifier can outperform expert-derived knowledge. The classifier was able to identify verb-cues with P of 88.9% and R of 72.6%. While frequently occurring verbs are highly predictive, the inclusion of VerbNet classes (Schuler, 2005) and contextual features allows for a more accurate classification of polysemous and unseen verbs.

Since PARC contains labelled and unlabelled attributions, which is detrimental for training, we used the verb-cue classifier to identify in the corpus sentences that we suspected contained an unlabelled attribution. Sentences containing a verb classified as a cue that do not contain a quotation were removed from the training set for the quotation extraction model.

4.4 Evaluation

We use two metrics, listed below, for evaluating the quotation spans predicted by our model against the gold spans from the annotation.

Strict The first is a strict metric where a predicted span is only considered to be correct if it exactly matches a span from the gold standard. The standard precision, recall, and F-score can be calculated using this definition of correctness. The drawback of this strict score is that if a prediction is incorrect by as little as one token it will be considered completely incorrect.

Partial We also consider an overlap metric (Hollingsworth and Teufel, 2005), which allows partially correct predictions to be proportionally counted. Precision (P), recall (R), and F-score for this method are:

$$P = \frac{\sum_{g \in gold} \sum_{p \in pred} overlap(g, p)}{|pred|} \quad (1)$$

$$R = \frac{\sum_{g \in gold} \sum_{p \in pred} overlap(p, g)}{|gold|} \quad (2)$$

$$F = \frac{2PR}{(P + R)} \quad (3)$$

Where $overlap(x, y)$ returns the proportion of tokens of y that are overlapped by x. For each of these metrics we report the micro-average, as the number of quotations in each document varies significantly. When reporting P for the typewise results we restrict the set of predicted quotations to only those with the requisite type, while still considering the full set of gold quotations. Similarly, when calculating R we restrict the set of gold quotations to only those with the required type.

4.5 Baselines

We have developed two baselines inspired by the current lexical/syntactic pattern-based approaches in the literature, which combine speech verbs and hand-crafted rules.

B_{lex} Lexical: cue verb + the longest of the spans before or after it until the sentence boundary.

B_{syn} Syntactic: cue verb + verb syntactic object. B_{syn} is close to the model in de La Clergerie et al. (2011).

Instead of relying on a lexicon of verbs, our baselines use those identified by the verb-cue classifier. As direct quotations are not always explicitly introduced by a cue-verb, we defined a separate baseline with a rule-based approach (B_{rule}) that returns text between quotation marks that has at least 3 tokens, and where the non-stopword and non-proper noun tokens are not all title cased. In our full results we apply each method along with B_{rule} and greedily take the longest predicted spans that do not overlap.

5 Supervised Approaches

We present two supervised approaches to quotation extraction, which operate over the tokens and the phrase-structure parse nodes respectively. Despite the difference in the item being classified, these approaches have some common features:

Lexical: unigram and bigram versions of the token, lemma, and POS tags within a window of 5 tokens either side of the target, all indexed by position.

Sentence: features indicating whether the sentence contains a quotation mark, a NE, a verb-cue, a pronoun, or any combination of these. There is also a sentence length feature.

Dependency: relation with parent, relations with any dependants, as well as versions of these that include the head and dependent tokens.

External knowledge: position-indexed features for whether any of the tokens in the sentence match a known role, organisation, or title. The titles come from a small hand-built list, while the role and organisation lists were built by recursively following the WordNet (Fellbaum, 1998) hyponyms of person and organization respectively.

Other: features for whether the target is within quotation marks, and whether there is a verb-cue near the end of the sentence.

		Strict			Partial		
		P	R	F	P	R	F
PARC	B_{rule}	75	94	83	96	94	95
	Token	97	91	94	98	97	97
SMHC	B_{rule}	87	93	90	98	94	96
	Token	94	90	92	99	97	98

Table 4: PARC and SMHC results on direct quotations. The token based approach is trained and tested on all quotations.

5.1 Token-based Approach

The token-based approach treats quotation extraction as analogous to NE tagging, where there are a sequence of tokens that need to be individually labelled. Each token is given either an I, an O, or a B label, where B denotes the first token in a quotation, I denotes the token is inside a quotation, and O indicates that the token is not part of a quotation. For NE tagging it is common to use a sentence as a single sequence, as NEs do not cross sentence boundaries. This does not work for quotations, as they can cross sentence and even paragraph boundaries. As such, we treat the entire document as a single sequence, which allows the predicted quotations to span both sentence and paragraph bounds.

We use a linear chain Conditional Random Field (CRF)[2] as the learning algorithm, with the common features listed above, as well as the following features:

Verb: features indicating whether the current token is a (possibly indirect) dependent of a verb-cue, and another for whether the token is at the start of a constituent that is a dependent of a verb-cue.

Ancestor: the labels of all constituents that contain the current token in their span, indexed by their depth in the parse tree.

Syntactic: the label, depth, and token span size of the highest constituent where the current token is the left-most token in the constituent, as well as its parent, and whether either of those contains a verb-cue.

[2]http://www.chokkan.org/software/crfsuite/

	Indirect			Mixed			All[1]		
Strict	P	R	F	P	R	F	P	R	F
B_{lex}	34	32	33	17	26	20	46	44	45
B_{syn}	78	46	58	61	40	49	80	63	70
Token	66	54	59	55	58	56	76	70	73
Constituent	61	50	55	50	38	43	70	64	67
Constituent$_G$	66	42	51	68	49	57	76	62	68
Partial	P	R	F	P	R	F	P	R	F
B_{lex}	56	66	61	78	79	78	73	79	76
B_{syn}	89	58	70	88	75	81	92	74	82
Token	79	74	76	85	90	87	87	86	87
Constituent	78	67	72	84	82	83	86	80	83
Constituent$_G$	80	54	65	90	80	85	90	74	81

Table 5: Results on PARC. [1]*All* reports the results over all quotations (direct, indirect and mixed). For the baselines, this is a combination of the strategy in B_{lex} or B_{syn} with the rules for direct quotations. Constituent$_G$ shows the results for the constituent model using the gold parse.

5.2 Constituent-based Approach

The constituent approach classifies whole phrase structure nodes as either *quotation* or *not a quotation*. Ideally each quotation would match exactly one constituent, however this is not always the case in our data. In cases without an exact match we label every constituent that is a subspan of the quotation as a *quotation* as long as it has a parent that is not a subspan of the quotation. In these cases multiple nodes will be labelled *quotation*, so a post-processing step is introduced that rebuilds quotations by merging predicted spans that are adjacent or overlapping within a sentence. Restricting the merging process this way loses the ability to predict quotations that cover more than a sentence, but without this restriction too many predicted quotations are erroneously merged.

This approach uses a maximum entropy classifier[3] with $L1$ regularisation. In early experiments we found that the constituent-based approach performed poorly when trained on all quotations, so for these experiments the constituent classifier is trained only on indirect and mixed quotations. The classifier uses the common features listed above as well as the following features:

Span: length of the span, features for whether there is a verb or a NE.

Node: the label, number of descendants, number of ancestors, and number of children of the target.

Context: dependency, node, and span features for the parent and siblings of the target.

In addition the lexical features described earlier are applied to both the start and end tokens of the node's span, as well as the highest token in the dependency parse that is within the span.

6 Results

6.1 Direct Quotations

Table 4 shows the results for predicting direct quotations on PARC and SMHC. In both corpora and with both metrics the token-based approach outperforms B_{rule}. Although direct quotations should be trivial to extract, and a simple system that returns the content between quotation marks should be hard to beat, there are two main factors that confound the rule-based system.

The first is the presence of mixed quotations, which is most clearly demonstrated in the difference between the strict precision scores and the partial precision scores for B_{rule}. B_{rule} will find all of the directly-quoted portions of mixed quotes, which do not exactly match a quotation, and so will receive a low precision score with the strict metric. However the partial overlap score will reward these

[3] http://scikit-learn.org/

	Indirect			Mixed			All[1]		
Strict	P	R	F	P	R	F	P	R	F
B_{lex}	37	42	40	15	36	21	50	50	50
B_{syn}	63	49	55	67	36	47	82	72	76
Token	69	53	60	80	91	85	82	75	78
Constituent	54	49	51	64	42	51	77	72	75
Partial	P	R	F	P	R	F	P	R	F
B_{lex}	52	68	59	87	77	82	77	84	81
B_{syn}	75	59	66	89	66	76	91	80	85
Token	82	67	74	88	84	86	92	86	89
Constituent	77	63	69	91	75	82	91	82	86

Table 6: Results on SMHC. [1]*All* reports the results over all quotations (direct, indirect and mixed). For the baselines, this is a combination of the strategy in B_{lex} or B_{syn} with the rules for direct quotations.

predictions, as they do partially match a quote, so there is a large difference in those scores. Note that the reduced strict score does not occur for the token method, which correctly identifies mixed quotations.

The other main issue is the presence of quotation marks around items such as book titles and scare quotes (i.e. text that is in quotation marks to distance the author from a particular wording or claim). In Section 4.5 we described the methods that we use to avoid scare quotes and titles, which are rule-based and imperfect. While these methods increase the overall F-score of B_{rule}, they do have a negative impact on recall, which is why the recall is lower than might be expected. These results demonstrate that although direct quotations can be accurately extracted with rules, the accuracy will be lower than might be anticipated and the returned spans will include a number of mixed quotations, which will be missing some content.

6.2 Indirect and Mixed Quotations

The token approach was also the most effective method for extracting indirect and mixed quotations as Tables 5 and 6 show. Indirect quotations were extracted with strict F-scores of 59% and 60% and partial F-scores of 76% and 74% in PARC and SMHC respectively, while mixed quotes were found with strict F-scores of 56% and 85% and partial F-scores of 87% and 86%.

Although there is a strong interconnection between syntax and attribution, results for B_{syn} show that merely considering attribution as a syntactic relation (Skadhauge and Hardt, 2005) has a large impact on recall: only a subset of inter-sentential quotations can be effectively matched by verb complement boundaries.

The constituent model yielded lower results than the token one, and in particular it greatly lowered the recall of mixed quotations in both corpora. Since the model heavily relies on syntax, it is particularly affected by errors made by the parser. The conjunction *and* in Example 3 is incorrectly attached by the parser to the cue *said*, leading the classifier to identify two separate spans. In order to verify the impact of incorrect parsing on the model, we ran the constituent model using gold standard parses for PARC. This resulted in an increase in strict P and increased the F-score for mixed quotations to 57%, similarly to the score achieved by the token model. However, it surprisingly negatively affected R for indirect quotations.

(3) Graeme Hugo, said *strong links between Australia's 700,000 ethnic Chinese and China could benefit both countries* ***and*** *were unlikely to pose a threat*.

The tables also report results for the extraction of all quotations, irrespective of their type. For this score, the baseline models for indirect and mixed quotations are combined with B_{rule} for direct quotations.

6.3 Model Comparison

We designed the features for the token and constituent models to be largely similar. This al-

lows us to conclude that the difference in performance between the token and constituent models is largely driven by the class labelling and learning method. Overall, the token-based approach outperformed both the baselines and the constituent method. Qualitatively we found that the token-based approach was making reasonable predictions most of the time, but would often fail when a quotation was attributed to a speaker through a parenthetical clause, as in Example 4.

(4) *Finding lunar ice*, said Tidbinbilla's spokesman, Glen Nagle, *would give a major boost to NASA's hopes of returning humans to the moon by 2020.*

The token-based approach has a reasonable balance of the various label types, and benefits from a decoding step that allows it to make trade-offs between good local decisions and a good overall solution. By comparison, the constituent-based approach has a large class imbalance, as there are many more negative (i.e. not quotation) parse nodes than there are positive, which makes finding a good decision boundary difficult. We experimented with reducing the number of negative nodes to consider, but found that the overall F-score was equivalent or worse, largely driven by a drop in recall. We also found that in many cases the constituent-approach predicted quotes that were too short, or that were only the second half of a conjunction, without the first half being labelled. We expect that these issues would be corrected with the addition of a decoding step, that forces the classifier to make a good global decision.

7 Speaker Attribution

While the focus of this paper is on extracting quotations, we also present results on finding the speaker of each quotation. As discussed in Section 2, quotation attribution has been addressed in the literature before, including some work that includes large-scale data (Elson and McKeown, 2010). However, the large-scale evaluations that exist cover only direct quotations, whereas we present results for direct, indirect, and mixed quotations.

For this evaluation we use four of the methods that were introduced in O'Keefe et al. (2012). The first is a simple rule-based approach (Rule) that returns the entity closest to the speech verb nearest the quotation, or if there is no such speech verb then the entity nearest the end of the quotation. The second method uses a CRF which is able to choose between up to 15 entities that are in the paragraph containing the quotation or any preceding it. The third method (No seq.) is a binary MaxEnt classifier that predicts whether each entity is the speaker or not the speaker, with the entity achieving the highest speaker probability predicted. In O'Keefe et al. (2012) this model achieved the best results on the direct quotations in SMHC, despite not using the sequence features or decoding methods that were available to other models. The final method that we evaluate (Gold) is the approach that uses sequence features that use the gold-standard labels from previous decisions. As noted by O'Keefe et al., this method is not realisable in practise, however we include these results so that we can reassess the claims of O'Keefe et al. when direct, indirect, and mixed quotations are included. For our results to be comparable we use the list of speech verbs that was presented in Elson and McKeown (2010) and used in O'Keefe et al. (2012).

Table 7 shows the accuracy of the two methods on both PARC and SMHC, broken down by the type of the quotation. The first observation that we make about these results in comparison to the O'Keefe et al. results, is that the accuracy is generally lower, even for direct quotations. This discrepancy is caused by differences in our data compared to theirs, notably that the sequence of quotations is altered in ours by the introduction of indirect quotations, and that some of the direct quotations that they evaluated would be considered mixed quotations in our corpora. The rule based method performs particularly poorly on PARC, which is likely caused by the relative scarcity of direct quotations and the fact that it was designed for direct quotations only. Direct quotations are much more frequent in SMHC, so the rules that rely on the sequence of speakers would likely perform relatively better than on PARC.

While the approach using gold-standard sequence features unsurprisingly performed the best, the most straightforward learned model (No seq.), trained without any sequence information, equalled or outperformed the two other non-gold approaches for all quotation types on both corpora. This indicates that the CRF model evaluated here was not able to effec-

Corpus	Method	Dir.	Ind.	Mix.	All
PARC	Rule	70	60	47	62
	CRF	82	68	65	73
	No seq.	85	74	65	77
	Gold	88	79	74	82
SMHC	Rule	89	76	78	84
	CRF	83	72	71	78
	No seq.	91	79	81	87
	Gold	93	81	83	89

Table 7: Speaker attribution accuracy results for both corpora over gold standard quotations.

tively use the sequence information that is present.

8 Conclusion

In this work we have presented the first large-scale experiments on the entire quotation extraction and attribution task: evaluating the extraction and attribution of direct, indirect and mixed quotations over two large news corpora. One of these corpora (SMHC) is a novel contribution of this work, while our results are the first presented for the other corpus (PARC). This work has shown that while rule-based approaches that return the object of a speech verb are indeed effective, they are outperformed by supervised systems that can take advantage of additional evidence. We also show that state-of-the-art quotation attribution methods are less accurate on indirect and mixed quotations than they are on direct quotations.

Future work will include extending these methods to extract all attributions, i.e. beliefs, eventualities, and facts, as well as the source spans. We will also evaluate the effect of adding a decoding step to the constituent approach. This work provides an accurate and complete quotation extraction and attribution system that can be used for a wide range of tasks in information extraction and opinion mining.

Acknowledgements

We would like to thank Bonnie Webber for her feedback and assistance. Pareti has been supported by a Scottish Informatics & Computer Science Alliance (SICSA) studentship; O'Keefe has been supported by a University of Sydney Merit scholarship and a Capital Markets CRC top-up scholarship. This work has been supported by ARC Discovery grant DP1097291 and the Capital Markets CRC Computable News project.

References

Sabine Bergler. 1992. *Evidential analysis of reported speech*. Ph.D. thesis, Brandeis University.

Sabine Bergler, Monia Doandes, Christine Gerard, and René Witte. 2004. Attributions. In *Exploring Attitude and Affect in Text: Theories and Applications*, Technical Report SS-04-07, pages 16–19. Papers from the 2004 AAAI Spring Symposium.

Eric de La Clergerie, Benoit Sagot, Rosa Stern, Pascal Denis, Gaelle Recource, and Victor Mignot. 2011. Extracting and visualizing quotations from news wires. *Human Language Technology. Challenges for Computer Science and Linguistics*, pages 522–532.

Cícero Nogueira dos Santos and Ruy Luiz Milidiú. 2009. Entropy guided transformation learning. In *Foundations of Computational, Intelligence Volume 1*, Studies in Computational Intelligence, pages 159–184. Springer.

David K. Elson and Kathleen R. McKeown. 2010. Automatic attribution of quoted speech in literary narrative. In *Proceedings of the Twenty-Fourth Conference of the Association for the Advancement of Artificial Intelligence*, pages 1013–1019.

Christine Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT press Cambridge, MA.

William Paulo Ducca Fernandes, Eduardo Motta, and Ruy Luiz Milidiú. 2011. Quotation extraction for portuguese. In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology (STIL 2011)*, pages 204–208.

Kevin Glass and Shaun Bangay. 2007. A naive salience-based method for speaker identification in fiction books. In *Proceedings of the 18th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA07)*, pages 1–6.

Bill Hollingsworth and Simone Teufel. 2005. Human annotation of lexical chains: Coverage and agreement measures. In *ELECTRA Workshop on Methodologies and Evaluation of Lexical Cohesion Techniques in Real-world Applications (Beyond Bag of Words)*, page 26.

Dan Klein and Christopher D Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10.

Ralf Krestel, Sabine Bergler, and René Witte. 2008. Minding the source: Automatic tagging of reported speech in newspaper articles. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Jisheng Liang, Navdeep Dhillon, and Krzysztof Koperski. 2010. A large-scale system for annotating and querying quotations in news feeds. In *Proceedings of the 3rd International Semantic Search Workshop*, pages 1–5.

Nuno Mamede and Pedro Chaleira. 2004. Character identification in children stories. *Advances in Natural Language Processing*, pages 82–90.

Tim O'Keefe, Silvia Pareti, James R. Curran, Irena Koprinska, and Matthew Honnibal. 2012. A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799.

Silvia Pareti. 2012. A database of attribution relations. In *Proceedings of the Eight International Conference on Language Resources and Evaluation*, pages 3213–3217.

Bruno Pouliquen, Ralf Steinberger, and Clive Best. 2007. Automatic detection of quotations in multilingual news. In *Proceedings of Recent Advances in Natural Language Processing*, pages 487–492.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Aravind Joshi, and Bonnie Webber. 2006. Annotating attribution in the Penn Discourse TreeBank. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 31–38.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0 annotation manual. In *Technical report, University of Pennsylvania: Institute for Research in Cognitive Science*.

Luis Sarmento and Sergio Nunes. 2009. Automatic extraction of quotes and topics from news feeds. In *4th Doctoral Symposium on Informatics Engineering*.

Roser Saurí and James Pustejovsky. 2009. Factbank: A corpus annotated with event factuality. *In Language Resources and Evaluation*, pages 227–268.

Nathan Schneider, Rebecca Hwa, Philip Gianfortoni, Dipanjan Das, Michael Heilman, Alan W. Black, Frederik L. Crabbe, and Noah A. Smith. 2010. Visualizing topical quotations over time to understand news discourse. Technical report, Carnegie Mellon University.

Karin K. Schuler. 2005. *Verbnet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, Faculties of Computer and Information Science of the University of Pennsylvania.

Peter R. Skadhauge and Daniel Hardt. 2005. Syntactic identification of attribution in the RST treebank. In *Proceedings of the Sixth International Workshop on Linguistically Interpreted Corpora*.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Computational Linguistics and Intelligent Text Processing*, pages 486–497. Springer.

Jason Zhang, Alan Black, and Richard Sproat. 2003. Identifying speakers in children's stories for speech synthesis. In *Proceedings of EUROSPEECH*, pages 2041–2044.

Identifying Web Search Query Reformulation using Concept based Matching

Ahmed Hassan
Microsoft Research
One Microsoft Way
Redmond, WA 98053, USA
hassanam@microsoft.com

Abstract

Web search users frequently modify their queries in hope of receiving better results. This process is referred to as "*Query Reformulation*". Previous research has mainly focused on proposing query reformulations in the form of suggested queries for users. Some research has studied the problem of predicting whether the current query is a reformulation of the previous query or not. However, this work has been limited to bag-of-words models where the main signals being used are word overlap, character level edit distance and word level edit distance. In this work, we show that relying solely on surface level text similarity results in many false positives where queries with different intents yet similar topics are mistakenly predicted as query reformulations. We propose a new representation for Web search queries based on identifying the concepts in queries and show that we can significantly improve query reformulation performance using features of query concepts.

1 Introduction

Web search is a process of querying, learning and reformulating queries to satisfy certain information needs. When a user submits a search query, the search engine attempts to return the best results to that query. Oftentimes, users modify their search queries in hope of getting better results. Typical search users have low tolerance to viewing lowly ranked search results and they prefer to reformulate the query rather than wade through result listings (Jansen and Spink, 2006). Previous studies have also shown that 37% of search queries are reformulations to previous queries (Jansen et al., 2007) and that 52% of users reformulate their queries (Jansen et al., 2005).

Understanding query reformulation behavior and being able to accurately identify reformulation queries have several benefits. One of these benefits is learning from user behavior to better suggest automatic query refinements or query alterations. Another benefit is using query reformulation prediction to identify boundaries between search tasks and hence segmenting user activities into topically coherent units. Also, if we are able to accurately identify query reformulations, then we will be in a better position to evaluate the satisfaction of users with query results. For example, search satisfaction is typically evaluated using clickthrough information by assuming that if a user clicks on a result, and possibly dwells for a certain amount of time, then the user is satisfied. Identifying query reformulation can be very useful for finding cases where the users are not satisfied even after a click on a result that may have seemed relevant given its title and summary but then turned out to be not relevant to the user's information need.

Previous work on query reformulation has either focused on automatic query refinement by the search system, e.g. (Jones et al., 2006; Boldi et al., 2008) or on defining taxonomies for query reformulation strategies, e.g. (Lau and Horvitz, 1999; Anick, 2003). Other work has proposed solutions for the query reformulation prediction problem or for the similar problem of task boundary identification (Radlinski and Joachims, 2005; Jones and Klinkner, 2008). These solutions have adopted the

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1000–1010,
Seattle, Washington, USA, 18-21 October 2013.

bag-of-words approach for representing queries and mostly used features of word overlap or character and word level edit distances. Take the queries "hotels in New York City" and "weather in New York City" as an example. The two queries are very likely to have been issued by a user who is planning to travel to New York City. The two queries have 5 words each, 4 of them are shared by the two queries. Hence, most of the solutions proposed in previous work for this problem will incorrectly assume that the second query is a reformulation of the first due to the high word overlap ratio and the small edit distance. In this work, we propose a method that goes beyond the bag-of-words method by identifying the concepts underlying these queries. In the previous example, we would like our method to realize that in the first query, the user is searching for "hotels" while for in the second query, she is searching for the "weather" in New York City. Hence, despite similar in terms of shared terms, the two queries have different intents and are not reformulations of one another.

To this end, we conducted a study where we collected thousands of consecutive queries and trained judges to label them as either reformulations or not. We then built a classifier to identify query reformulation pairs and showed that the proposed classifier outperforms the state-of-the-art methods on identifying query reformulations. The proposed method significantly reduces false positives (non-reformulation pairs incorrectly classified as reformulation) while achieving high recall and precision.

2 Related Work

There are three areas of work related to the research presented in this paper: (i) query reformulation taxonomies, (ii) automatic query refinement, and (iii) search tasks boundary identification. We cover each of these areas in turn.

2.1 Query Reformulation Taxonomies

Existing research has studied how web search engines can propose reformulations, but has given less attention to how people perform query reformulations. Most of the research on manual query reformulation has focused on building taxonomies of query reformulation. These taxonomies are generally constructed by examining a small set of query logs. Anick (2003) classified a random sample of 100 reformulations by hand into eleven categories. Jensen et al. (2007) identified 6 different kinds of reformulation states (New, Assistance, Content Change, Generalization, Reformulation, and Specialization) and provided heuristics for identifying them. They also used them to predict when a user is most receptive to automatic query suggestions. The same categories were used in several other studies (Guo et al., 2008; Lau and Horvitz, 1999). Huang and Efthimis (2010) proposed another reformulation taxonomy. Their taxonomy was lexical in nature (e.g., word reorder, adding words, removing words, etc.). They also proposed the use of regular expressions to identify them. While studying re-finding behavior, Teevan et al. (2006) constructed a taxonomy of query re-finding by manually examining query logs, and implemented algorithms to identify repeat queries, equal click queries and overlapping click queries. None of this work has built an automatic classifier distinguishing reformulation queries from other queries. Heuristics and regular expressions have been used in (Huang et al., 2010) and (Jansen et al., 2007) to identify different types of reformulations. This line of work is relevant to our work because it studies query reformulation strategies. Our work is different because we build a machine-learned predictive model to identify query reformulation while this line of work mainly focuses on defining taxonomies for reformulation strategies.

2.2 Automatic Query Refinement

A close problem that has received most of the research attention in this area is the problem of automatically generating query refinements. These refinements are typically offered as query suggestions to the users or used to alter the user query before submitting it to the search engine.

Boldi et al. (2008) introduced the concept of the query-flow graph where every query is represented by a node and edges connect queries if it is likely for users to move from one query to another. Mei et al. (2008) used random walks over a bipartite graph of queries and URLs to find query refinements. Query logs were used to suggest query refinements in (Baeza-Yates et al., 2005). Hierarchical agglomerative clustering was used to group similar queries that can be used as suggestions for one

another. Other research has adopted methods based on query expansion (Mitra et al., 1998) or query substitution (Jones et al., 2006). This line of work is different from our work because it focuses on automatically generating query refinements while this work focuses on identifying cases of manual query reformulations.

2.3 Search Task Boundary Identification

The problem of classifying the boundaries of the user search tasks within sessions in web search logs has been widely addressed before. This problem is closely related to the problem of identifying query reformulation. A search task has been defined in (Jones and Klinkner, 2008) as a single information need that may result in one or more queries. Similarly , Jansen et al. (2007) defined a session as a series of interactions by the user toward addressing a single information need. On the other hand, a query reformulation is intended to modify a previous query in hope of getting better results to satisfy the same information need. From these definitions, it is clear how query reformulation and task boundary detection are two sides of the same problem.

Boldi et al. (2008) presented the concept of the query-flow graph. A query-flow graph represents chains of related queries in query logs. They use this model for finding logical session boundaries and query recommendation. Ozmutlu (2006) proposed a method for identifying new topics in search logs. He demonstrated that time interval, search pattern and position of a query in a user session, are effective for shifting to a new topic. Radlinski and Joachims (2005) study sequences of related queries (query chains). They used that to generate new types of preference judgments from search engine logs to learn better ranked retrieval functions.

Arlitt (2000) found session boundaries using a calculated timeout threshold. Murray et al. (2006) extended this work by using hierarchical clustering to find better timeout values to detect session boundaries. Jones and Klinkner (2008) also addressed the problem of classifying the boundaries of the goals and missions in search logs. They showed that using features like edit distance and common words achieves considerably better results compared to timeouts. Lucchese et al. (Lucchese et al., 20011) uses a similar set of features as (Jones and Klinkner, 2008), but uses clustering to group queries in the same task together as opposed to identifying task boundary as in (Jones and Klinkner, 2008). This line of work is perhaps the closest to our work. Our work is different because it goes beyond the bag of words approach and tries to assess query similarity based on the concepts represented in each query. We compare our work to the state-of-the-art work in this area later in this paper.

3 Problem Definition

We start by defining some terms that will be used throughout the paper:

Definition: *Query Reformulation* is the act of submitting a query Q_2 to modify a previous search query Q_1 in hope of retrieving better results to satisfy the same information need.

Definition: A *Search Session* is group of queries and clicks demarcated with a 30-minute inactivity timeout, such as that used in previous work (Downey et al., 2007; Radlinski and Joachims, 2005).

Search engines receive streams of queries from users. In response to each query, the engine returns a set of search results. Depending on these results, the user may decide to click on one or more results, submit another query, or end the search session. In this work, we focus on cases where the user submits another query. Our objective is to solve the following problem: Given a query Q_1, and the following query Q_2, predict whether Q_2 is reformulation of Q_1.

4 Approach

In this section, we propose methods for predicting whether the current query has been issued by the user to reformulate the previous query.

4.1 Query Normalization

We perform standard normalization where we replace all letters with their corresponding lower case representation. We also replace all runs of whitespace characters with a single space and remove any leading or trailing spaces. In addition to the standard normalization, we also break queries that do not respect word boundaries into words. Word breaking is a well-studied topic that has proved to be

Table 1 : Examples of queries, the corresponding segmentation, and the concept representation. Phrases are separated by "|" and different tokens in a keyword are separated by "_"

Query	Phrases and Keywords	Concept Representation
hotels in new york city	hotels in new_york_city	Concept1 {head="hotels", modifiers = "new york city"}
hyundai roadside assistance phone number	hyundai roadside_assistance | phone_number	Concept1 {head = "roadside _assistance", modifiers = "hyundai"}, Concept2{"phone_number"}
kodak easyshare recharger chord	kodak_easyshare recharger_cord	Concept1{head ="recharger_cord", modifiers ="kodak easyshare"}
user reviews for apple iphone	user_reviews for apple_iphone	Concept1{head="user_reviews" , modifiers = "apple iphone"}
user reviews for apple ipad	user_reviews for apple_ipad	Concept1{head ="user_reviews", modifiers = "apple ipad"}
tommy bhama rug	tommy_bhama rug	Concept1{head ="rug", modifiers ="tommy bhama"}
tommy bhama perfume	tommy_bhama perfume	Concept1{head ="perfume", modifiers ="tommy bhama"}

useful for many natural language processing applications. This becomes a frequent problems with queries when users do not observe the correct word boundaries (for example: "southjerseycraigslist" for "south jersey craiglist") or when users are searching for a part of a URL (for example "quincycollege" for "quincy college"). We used a freely available word breaker Web service that has been described at (Wang et al., 2011).

4.2 Queries to Concepts

Lexical similarity between queries has been often used to identify related queries (Jansen et al., 2007). The problem with lexical similarity is that it introduces many false negatives (e.g. synonyms) , but this can be handled by other features as we will describe later. More seriously, it introduces many false positives. Take the following query pair as an example Q_1: weather in new york city and Q_2: "hotels in new york city". Out of 5 words, 4 words are shared between Q_1 and Q_2. Hence, any lexical similarity feature would predict that the user submitted Q_2 as a reformulation of Q_1. What we would like to do is to have a query representation that recognizes the difference between Q_1 and Q_2.

If we look closely at the two queries, we will notice that in the first query, the user is looking for the "weather", while in the second query the user is looking for "hotels". We would like to recognize "weather", and "hotels" as the head keywords of Q_1 and Q_2 respectively, while "new york city" is a modifier of the head keyword in both cases. To build such a representation, we start by segmenting each query into phrases. Query segmentation is the process of taking a users search query and dividing the tokens into individual phrases or semantic units (Bergsma and Wang, 2007). Many approaches to query segmentation have been presented in recent research. Some of them pose the problem as a supervised learning problem (Bergsma and Wang, 2007; Yu and Shi, 2009). Many of the supervised methods though use expensive features that are difficult to reimplement.

On the other hand, many unsupervised methods for query segmentation have also been proposed (Hagen et al., 2011; Hagen et al., 2010). Most of these methods use only raw web n-gram frequencies and are very easy to re-implement. Additionally, Hagen et al. (2010) have shown that these methods can achieve segmentation accuracy comparable to current state-of-the-art techniques using supervised learning. We opt for the unsupervised techniques to perform query segmentation. More specifically, we adopt the mutual information

method (MI) used throughout the literature. A segmentation for a query is obtained by computing the pointwise mutual information score for each pair of consecutive words. More formally, for a query $x = \{x_1, x_2, ..., x_n\}$

$$PMI(x_i, x_{i+1}) = log \frac{p(x_i, x_{i+1})}{p(x_i)p(x_{i+1})} \quad (1)$$

where $p(x_i, x_{i+1})$ is the joint probability of occurrence of the bigram (x_i, x_{i+1}) and $p(x_i)$ and $p(x_{i+1})$ are the individual occurrence probabilities of the two tokens x_i and x_{i+1} .

A segment break is introduced whenever the point wise mutual information between two consecutive words drops below a certain threshold τ. The threshold we used, $\tau = 0.895$, was selected to maximize the break accuracy (Jones et al., 2006) on the Bergsma-Wang-Corpus (Bergsma and Wang, 2007). Furthermore, we do not allow a break to happen between a noun and a proposition (e.g. no break can be introduced between "hotels" and "in" or "in" and "new York" in the query "hotels in new york city"). We will shorty explain how we obtained the part-of-speech tags.

In addition to breaking the query into phrases, we were also interested in grouping multi-word keywords together (e.g. "new york", "Michael Jackson", etc.). The intuition behind that is that a query containing the keyword "new york" and another containing the keyword "new mexico" should not be awarded because they share the word "new". We do that by adopting a hierarchical segmentation technique where the same segmentation method described above is reapplied to every resulting phrase with a new threshold $\tau_s < \tau$. We selected the new threshold, $\tau = 1.91$, to maximize the break accuracy over a set of a random sample of 10,000 Wikipedia title of persons, cities, countries and organizations and a random sample of bigrams and trigrams from Wikipedia text.

In our implementation, the probabilities for all words and n-grams have been computed using the freely available Microsoft Web N-Gram Service (Huang et al., 2010).

Now that we have the phrases and keywords in each query, we assume that every phrase corresponds to a semantic unit. Every semantic unit has a head and a zero or more modifiers. Dependency parsing could be used to identify the head and modifiers from every phrase. However, because queries are typical short and not always well-formed sentences, this may pose a challenge to the dependency parser. But as we are mainly interested in short noun phrases, we can apply a simple set of rules to identify the head keyword of each phrase using the part of speech tags of the words in the phrase. A part-of-speech (POS) tagger assigns parts of speech to each word in an input text, such as noun, verb, adjective, etc. We used the Stanford POS tagger, using Stanford CoreNLP, to assign POS tags to queries (Toutanova et al., 2003). To identify the head and attributes of every noun phrase, we use the following rules:

- For phrases with of the form: "NNX+" (i.e. one more nouns, where NNX could be NN: noun, singular, NNS: noun, plural, NNP: proper noun, singular or NNPS: proper noun, plural), the head is the last noun keyword and all other keywords are treated as attributes/modifiers.

- For the phrases of the form "NNX+ IN NNX+", where IN denotes a preposition or a subordinating conjunction (e.g. "in", "of", etc.), the head is the last noun keyword before the preposition.

Table 1 shows different examples of queries, the corresponding phrases, keywords, and concepts. For example the query "kodak easyshare recharger chord" consists of a single semantic unit (phrase) and two keywords "Kodak easyshare" and "recharger cord". The head of this semantic unit is the keyword "recharger cord" and "kodak easyshare" is regarded as an attribute/modifier. Another example is the two queries "tommy bhama rug" and "tommy bhama perfume". The head of the former is "rug", while the head of the latter is "perfume". Both share the attribute "tommy bhama". This shows that the user had two different intents even though most of the words in the two queries are shared.

4.3 Matching Concepts

Phrases in two concepts may have full term overlap, partial term overlap, or no direct overlap yet are semantically similar. To capture concept similarity,

we define four different ways of matching concepts ranked from the most to the least strict:

- Exact Match: The head and the attributes of the two concepts match exactly.
- Approximate Match: To capture spelling variants and misspelling, we allow two keywords to match if the Levenshtein edit distance between them is less than 2.
- Lemma Match: Lemmatization is the process of reducing an inflected spelling to its lexical root or lemma form. We match two concepts if the lemmas of their keywords can be matched.
- Semantic Match: We compute the concept similarity by measuring the semantic similarity between the two phrases from which the concepts where extracted. Let $Q = \{q_1, ..., q_I\}$ be one phrase and $S = \{s_1, ..., s_J\}$ be another, the semantic similarity between these two phrases can be measured by estimating the probability of one of them being a translation of another. The translation probabilities can be estimated using the IBM Model 1 (Brown et al., 1993; Berger and Lafferty, 1999). The model was originally proposed to model the probability of translating from one sequence of words in one language to another. It has been also used in different IR applications to estimate the probability of translating from one sequence of words to another sequence in the same language (e.g. (Gao et al., 2012), (Gao et al., 2010) and (White et al., 2013)). More formally, the similarity between two sequences of words, $Q = \{q_1, ..., q_I\}$ and $S = \{s_1, ..., s_J\}$, can be defined as:

$$P(S|Q) = \prod_{i=1}^{I} \sum_{j=1}^{J} P(s_i|q_j) P(q_j|Q) \quad (2)$$

where $P(q|Q)$ is the unigram probability of word q in query Q. The word translation probabilities $P(s|q)$ are estimated using the query-title pairs derived from the clickthrough search logs, assuming that the title terms are likely to be the desired alternation of the paired query. The word translation probabilities $P(s|q)$ (i.e. the model parameters θ) are optimized by maximizing the probability of generating document titles from queries over the entire training corpus:

$$\theta^* = argmax_\theta \prod_{i=1}^{M} P(S|Q, \theta) \quad (3)$$

where $P(S|Q, \theta)$ is defined as:

$$P(S|Q, \theta) = \frac{\epsilon}{(J+!)^I} \prod_{i=1}^{I} \sum_{j=1}^{J} P(s_i|q_j) \quad (4)$$

where ϵ is a constant, I is the token length of S, and J is the token length of Q. The query-title pairs used for model training are sampled from one year worth of search logs from a commercial search engine. The search logs do not intersect with the search logs where the data described in Section5.1 has been sampled from. S and Q are considered a match if $P(S|Q, \theta) > 0.5$.

4.4 Features

4.4.1 Textual Features

Jones and Klinkner (2008) showed that word and character edit features are very useful for identifying same task queries. The intuition behind this is that consecutive queries which have many words and/or characters in common tend to be related. The features they used are:

- normalized Levenshtein edit distance
- 1 if $lev > 2$, 0 otherwise
- Number of characters in common starting from the left
- Number of characters in common starting from the right
- Number of words in common starting from the left
- Number of words in common starting from the right
- Number of words in common
- Jaccard distance between sets of words

4.4.2 Concept Features

As we explained earlier the word and character edit features capture similarity between many pairs of queries. However, they also tend to mis-classify many other pairs especially when the two queries share many words yet have different intents. We used the conceptual representation of queries described in the previous subsection to compute the following set of features, notice that every feature has two variants one at the concept level and the other at the keyword (head or attribute) level:

- Number of "exact match" concepts in common
- Number of "approximate match" concepts in common
- Number of "lemma match" concepts in common
- Number of "semantic match" concepts in common
- Number of concepts in Q_1
- Number of concepts in Q_2
- Number of concepts in Q_1 but not in Q_2
- Number of concepts in Q_1 but not in Q_2
- 1 if Q_1 contains all Q_2s concepts
- 1 if Q_2 contains all Q_1s concepts
- all above features recomputed for keywords instead of concepts

4.4.3 Other Features

Other features, that have been also used in (Jones and Klinkner, 2008), include temporal features:

- time between queries in seconds
- time between queries as a binary feature (5 mins, 10 mins, 20 mins, 30 mins, 60 mins, 120 mins)

and search results feature:

- cosine distance between vectors derived from the first 10 search results for the query terms.

4.5 Predicting Reformulation Type

There are different strategies users use to reformulate a query which results in different types of query reformulations:

- Generalization: A *generalization* reformulation occurs when the second query is intended to seek more general information compared to the first query
- Specification: A *specification* reformulation occurs when the second query is intended to seek more specific information compared to the first query
- Spelling: A *spelling* reformulation occurs when the second query is intended to correct one or more misspelled words in the first query
- Same Intent: A *same intent* reformulation occurs when the second query is intended to express the same intent as the first query. This can be the result of word substitution or word reorder.

We used the following features to predict the query reformulation type:

- Length (num. characters and num. words) of Q_1, Q_2 and difference between them
- Number of out-of-vocabulary words in Q_1, Q_2 and the difference between them
- num. of "exact match" concepts in common
- num. of "approximate match" concepts in common
- num. of "lemma match" concepts in common
- num. of "semantic match" concepts in common
- num. of concepts in Q_1, Q_2 and the difference between them
- num. of concepts in Q_1 but not in Q_2
- num. of concepts in Q_1 but not in Q_2
- 1 if Q_1 contains all Q_2s concepts
- 1 if Q_2 contains all Q_1s concepts
- all concept features above recomputed for keywords instead of concepts

5 Experiments and Results

5.1 Data

Our data consists of query pairs randomly sampled from the queries submitted to a commercial search engine during a week in mid-2012. Every record in our data consisted of a consecutive query pair (Q_i,Q_{i+1}) submitted to the search engine by the same user and in the same session (i.e. within less than 30 minutes of idle time, the 30 minutes threshold has been frequently used in previous work, e.g. (White and Drucker, 2007)). Identical queries were excluded from the data because they are always labeled as reformulation and their label is very easy to predict. Hence, when included, they result in unrealistically high estimates of the performance of the proposed methods. All data in the session to which the sampled query pair belongs were recorded. In addition to queries, the data contained a timestamp for each page view, all elements shown in response to that query (e.g. Web results, answers, etc.), and visited Web page or clicked answers. Intranet and secure URL visits were excluded. Any personally identifiable information was removed from the data prior to analysis.

Annotators were instructed to exhaustively examine each session and "re-enact" the user's experience. The annotators inspected the entire search results page for each of Q_i and Q_{i+1}, including URLs, page titles, relevant snippets, and other features. They were also shown clicks to aid them in their judgments. Additionally, they were also shown queries and clicks before and after the query pair of interest. They were asked to then use their assessment of the user's objectives to determine whether Q_{i+1} is a reformulation of Q_i. Each query pair was labeled by three judges and the majority vote among judges was used. Because the number of positive instances is much smaller than the number of negative instances, we used all positive instances and an equal number of randomly selected negative instances leaving us with approximately 6000 query pairs.

Judges were also asked to classify reformulations into one of four different categories: Generalization (second query is intended to seek more general information), Specification (second query is intended to seek more specific information), Spelling (second query is intended to correct spelling mistakes), and Same Intent (second query is intended to express the same intent in a different way).

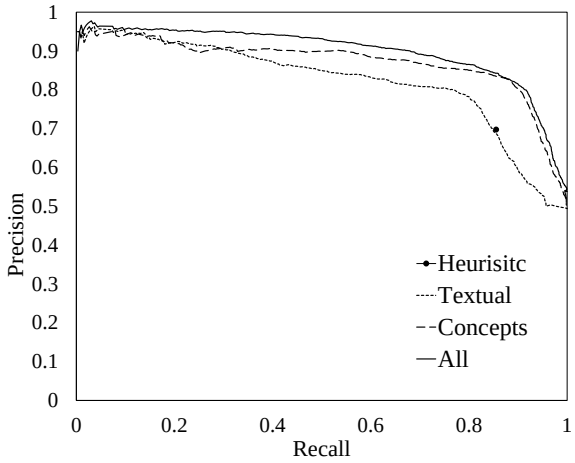


Figure 1 Precision-Recall Curves for the Reformulation Prediction Methods

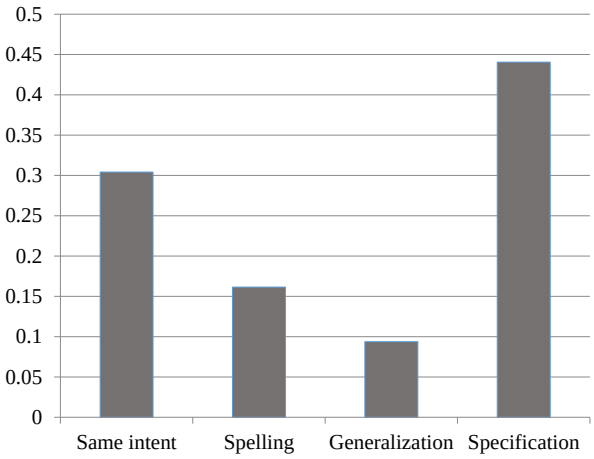


Figure 2 Distribution of Query Reformulation Types

5.2 Predicting Query Reformulation

In this section we describe the experiments we conducted to evaluate the reformulation prediction classifier. We perform experiments using the data described in the previous section. We compare the performance of four different systems:

- The first one, *Heuristic*, simply computes the similarity between two queries as the percentage of common words to the length of the longer query in terms of the number of words.

Table 2 : Heuristics vs. Textual vs. Concept Features for Reformulation Prediction

	Accuracy	Reform. F1	No-Reform. F1
Heuristics	77.10%	75.60%	69.07%
Textual	82.90%	71.75%	87.75%
Concepts	87.60%	81.20%	90.78%
All	89.02%	83.63%	91.75%

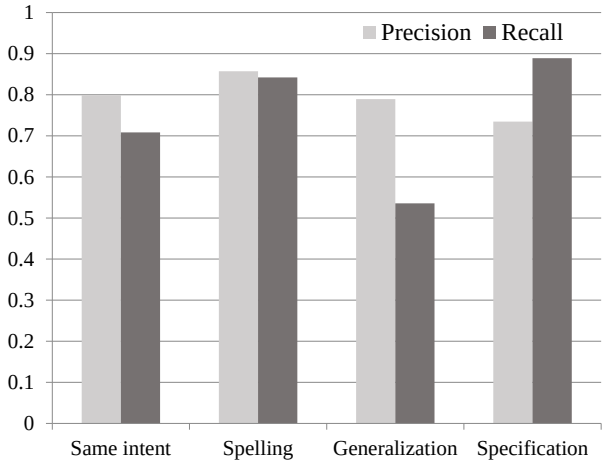


Figure 3 Precision and Recall for Query Reformulation Type Prediction

When finding common words, it allows two words to be matched if their Levenshtein edit distance is less than or equals 2. The second query is predicted to be a reformulation of the first if similarity $\geq \tau_{sim}$ and the time difference $\leq \tau_{time}$ minutes. The two thresholds were set to 0.35 and 5 minutes respectively using grid search to maximize accuracy over the training data.

- The second system, *Textual*, uses the textual features from previous work that have been described in Section 4.4.1 and the temporal and results features described in Section 4.4.3.

- The third system, *Concepts*, uses the concept features that we presented in Section 4.4.2 and the temporal and results features described in Section 4.4.3.

- Finally, the last system, *All*, uses both the textual features, the conceptual features and the temporal and results features.

For all methods, we used gradient boosted regression trees as a classifier with 10-fold cross validation. We also tried other classifiers like SVM and logistic regression but we got the best performance using the gradient boosted regression trees. All reported differences are statistically significant at the 0.05 level according to a two-tailed student t-test.

The accuracy, positive (reformulation) F1, and negative (non-reformulation) F1 for the four methods are shown in Table 2. The precision recall curves for all methods are shown in Figure 1; the heuristic method uses fixed thresholds resulting in a single operating point. The results show that the concept features outperform the textual features. Combining them together results in a small gain over using the concept features only. The concept features were able to achieve higher precision rates while not sacrificing recall because they were more effective in eliminating false reformulation cases.

We examined the cases where the classifier failed to predict the correct label to understand when the classifier fails to work properly. We identified several cases where this happens. For example, the classifier failed to match some terms that have the same semantic meaning. Many of these cases were acronyms (e.g. “AR” and “Accelerated Reader”, ”GE” and ”General Electric”). These cases can be handled by using a semantic matching method that yields higher coverage especially in cases of acronyms.

The classifier also failed in cases where the keyword extractor and/or the POS tagger failed to correctly parse the queries (e.g. “last to know”was not recognized as a song name). These cases can be handled by identifying named entities as a preprocessing step and treating them accordingly when identifying keywords or assigning POS tags to keywords.

Another dominant class of cases where the classifier failed were cases where the dependency rules failed to correctly identify the head keyword in a query. In many such cases, the query was a non well-formed sequence of words (e.g. "dresses Christmas toddler"). This is the hardest class to handle. Since it is hard to correctly parse short text and it is even harder when the text it is not well-formed.

5.3 Predicting Reformulation Type

We conducted another experiment to evaluate the performance of the reformulation type classifier. We performed experiments using the data described earlier where judges were asked to select the type of reformulation for every reformulation query. The distribution of reformulations across types is shown in Figure 2. The figure shows that most popular reformulations types are those where users move to a more specific intent or express the same intent in a different way. Reformulations with spelling suggestions and query generalizations are less popular. We conducted a one-vs-all experiment using gradient boosted regression trees with 10-fold cross validation. The precision and recall of every type are shown in Figure 3. The micro-averaged and macro-averaged accuracy was 78.13% and 72.52% respectively.

6 Conclusions

Identifying query reformulations is an interesting and useful application in Information Retrieval. Reformulation identification is useful for automatic query refinements, task boundary identification and satisfaction prediction. Previous work on this problem has adopted a bag-of-words approach where lexical similarity and word overlap are the key features for identifying query reformulation. We proposed a method for identifying concepts in search queries and using them to identify query reformulations. The proposed method outperforms previous work because it can better represent the information intent underlying the query and hence can better assess query similarity. We showed that the proposed method significantly outperforms the other methods. We also showed that we can reliably predict the type of the reformulation with high accuracy.

References

Peter Anick. 2003. Learning noun phrase query segmentation. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 88–95.

M. Arlitt. 2000. Characterizing web user sessions. *ACM SIGMETRICS Performance Eval Review*, 28(2):50–63.

Ricardo Baeza-Yates, Carlos Hurtado, Marcelo Mendoza, and Georges Dupret. 2005. Modeling user search behavior. In *LA-WEB '05: Proceedings of the Third Latin American Web Congress*, Washington, DC, USA. IEEE Computer Society.

A. L. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1999)*, pages 222–229.

S. Bergsma and I. Q. Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 816–826.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The query-flow graph: model and applications. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM 2008)*, pages 609–618.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Doug Downey, Susan Dumais, and Eric Horvitz. 2007. Models of searching and browsing: Languages, studies, and applications. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(6):862–871.

J. Gao, X. He, and J. Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceeding of the ACM conference on Information and knowledge management (CIKM 2010)*, pages 1139–1148.

J. Gao, S. Xie, X. He, and A. Ali. 2012. Learning lexicon models from search logs for query expansion. In *Proceeding of the Conference on Emprical Methods for Natural Language Processing (EMNLP 2012)*.

J. Guo, G. Xu, H. Li, and X. Cheng. 2008. A unified and discriminative model for query refinement. In *Proceedings of the annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386.

M. Hagen, M. Potthast, B. Stein, and C. Brautigam. 2010. The power of naiv query segmentation. In *Proceeding of the ACM Conference of the Special Interest*

Group on Information Retrieval (SIGIR 2010), pages 797–798,.

M. Hagen, M. Potthast, B. Stein, and C. Brautigam. 2011. Query segmentation revisited. In *Proceeding of the ACM World Wide Web Conference (WWW 2011)*, pages 97–106.

J. Huang, J. Gao, J. Miao, X. Li, K. Wang, and F. Behr. 2010. Exploring web scale language models for search query processing. In *Proceeding of the ACM World Wide Web Conference (WWW 2010)*, pages 451–460.

Bernard J. Jansen and Amanda Spink. 2006. How are we searching the world wide web?: a comparison of nine search engine transaction logs. *Inf. Process. Manage.*, 42:248–263, January.

B. J. Jansen, A. Spink, and J. Pedersen. 2005. A temporal comparison of altavista web searching. *Journal of the American Society for Information Science and Technology*, 56:559–570.

B. J. Jansen, M. Zhang, and A. Spink. 2007. Patterns and transitions of query reformulation during web searching. *International Journal of Web Information Systems*.

Rosie Jones and Kristina Klinkner. 2008. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM 2008)*.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substititions. In *Proceedings of the Fifteenth International Conference on the World-Wide Web (WWW06)*, pages 387–396.

Tessa Lau and Eric Horvitz. 1999. Patterns of search: Analyzing and modeling web query refinement. In ACM Press, editor, *Proceedings of the Seventh International Conference on User Modeling*.

C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. 20011. Identifying task-based sessions in search engine query logs. In *Proceedings of ACM Conference on Web Search and Data Mining(WSDM 2011)*.

Q. Mei, D. Zhou, and K. Church. 2008. Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM 2008)*, pages 469–478.

M. Mitra, A. Singhal, and C. Buckley. 1998. Improving automatic query expansion. In *Proceedings of the 21th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214.

G. V. Murray, J. Lin, and A. Chowdhury. 2006. Identification of user sessions with hierarchical agglomerative clustering. *ASIST*, 43(1):934–950.

Seda Ozmutlu. 2006. Automatic new topic identification using multiple linear regression. *Information Processing and Management*, 42(4):934–950.

Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In Robert Grossman, Roberto Bayardo, and Kristin P. Bennett, editors, *KDD*, pages 239–248. ACM.

Jamie Teevan, Eytan Adar, Rosie Jones, and Michael Potts. 2006. History repeats itself: Repeat queries in yahoo's logs. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 703–704.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceeding of the Human Language Technologies Conference and the Annual Meeting of the North American Association of Computational Linguists (HLT-NAACL 2003)*, pages 252–259.

K. Wang, C. Thrasher, and B. Hsu. 2011. Web scale nlp: A case study on url word breaking. In *Proceeding of the ACM World Wide Web Conference (WWW 2011)*, pages 357–366.

Ryen W. White and Steven M. Drucker. 2007. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*.

Ryen W. White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 1411–1420.

X. Yu and H. Shi. 2009. Query segmentation using conditional random fields. In *Proceedings of the Workshop on Keyword Search on Structured Data (KEYS)*, pages 21–26.

The Answer is at your Fingertips: Improving Passage Retrieval for Web Question Answering with Search Behavior Data

Mikhail Ageev*
Moscow State University
mageev@yandex.ru

Dmitry Lagun
Emory University
dlagun@emory.edu

Eugene Agichtein
Emory University
eugene@mathcs.emory.edu

Abstract

Passage retrieval is a crucial first step of automatic Question Answering (QA). While existing passage retrieval algorithms are effective at selecting document passages most similar to the question, or those that contain the expected answer types, they do not take into account which parts of the document the searchers actually found useful. We propose, to the best of our knowledge, the first successful attempt to incorporate searcher examination data into passage retrieval for question answering. Specifically, we exploit detailed examination data, such as mouse cursor movements and scrolling, to infer the parts of the document the searcher found interesting, and then incorporate this signal into passage retrieval for QA. Our extensive experiments and analysis demonstrate that our method significantly improves passage retrieval, compared to using textual features alone. As an additional contribution, we make available to the research community the code and the search behavior data used in this study, with the hope of encouraging further research in this area.

1 Introduction

Automated Question Answering (QA), is an attractive variation of search where the QA system automatically returns an *answer* to a user's question, instead of a list of document results. Passage retrieval is a first critical step of QA system, where candidate passages are identified and scored as likely to contain an answer. While significant progress has been made recently on incorporating syntactic and semantic analysis for improving the QA system performance, this analysis is typically applied only on the (limited) set of candidate passages retrieved. The main reason is that it is generally not practical to perform deep analysis on all documents in a large collection, and not yet feasible for the Web at large.

In the web search setting, automated question answering presents additional challenges and opportunities. On the downside, the questions and queries from real users are often not grammatical or well-formed, differing from the questions used in the traditional TREC Question Answering evaluations (Kelly and Lin, 2007; Sun et al., 2005). On the upside, by interacting with a search engine, the millions of searchers implicitly provide additional clues about usefulness of documents, result ranking, and other aspects of the search process. In this paper, we explore making use of the search behavior data to improve passage retrieval for automated Question Answering on the web.

Our basic observation is that when a user is attempting to answer a question, he or she will more carefully examine the parts of the document that contain an answer. This observation is intuitive, and is strongly supported by numerous eye tracking studies (e.g., Buscher et al. (2008) and Buscher et al. (2009a)). Based on this, we hypothesize that the passages containing the answers can be *automatically identified* from the naturalistic searcher behavior, and this prediction can be subsequently used to improve passage ranking. To the best of our knowledge, our work is the first to successfully incorporate searcher examination into passage ranking for Question Answering.

Our approach is primarily aimed at recurring (repeated) questions, which comprise a large fraction of the search volume (while the exact statistics vary, over 50% of search queries are submitted by multiple users). For such questions, a system would track the clicked result URLs, as well as the user interactions on the landing pages. Then the system would use this information to present the improved results to new users who ask the same (or similar) question. Intuitively, our method uses the same general idea of result click data mining, used by the major search engines to improve result ranking, but takes

*Work done at Emory University.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1011–1021,
Seattle, Washington, USA, 18-21 October 2013.

it a step further to exploit user interactions on the actual landing pages. A key point to emphasize is that our approach exploits the *natural browsing behavior* of the users, not requiring any additional effort from the searchers.

Specifically, our contributions include:

- A novel approach to passage retrieval for question answering, that naturally integrates textual and behavioral evidence.
- A robust infrastructure for connecting fine-grained searcher behavior to precise page contents.
- Thorough experiments over hundreds of search sessions and thousands of page views, demonstrating significant improvements to passage retrieval by harnessing the user's page examination data.

Next we describe related work, to place our contribution in context.

2 Related Work

Our work brings together two areas of research: passage retrieval for question answering, and mining searcher behavior data.

Passage retrieval has long been recognized as the first crucial step of automatic question answering. In some cases, passage retrieval can even serve as the final product of a Question Answering system (Clarke et al., 2000). As another example, redundancy in the retrieved passages has been used by the AskMSR system (Brill et al., 2002) to select answers. Tellex et al. (2003) report a thorough comparison of passage retrieval methods for QA, up to 2003. Additional improvements have been achieved by using deeper analysis of the text. For example, Cui et al. (2005) exploited dependency relations between the question terms, Aktolga et al. (2011) incorporated syntactic structure and answer typing, while Harabagiu et al. (2005) used semantic analysis at all stages of the question answering process. In this paper, we pursue a complementary direction, by exploiting searcher examination behavior, with the assumption that human searchers can easily zoom in on relevant passages as part of normal searching.

It has been previously recognized that searcher interactions could be valuable for question answering, and a task on Complex Interactive QA has been ran as part of TREC 2007 (Kelly and Lin, 2007). Our work goes much further by considering not only explicit interactions, but also the searcher examination behavior (i.e., detailed information on which text passages were examined) – which, as we show, provides additional valuable information for passage retrieval. Furthermore, it has been recognized that the questions used in traditional TREC QA evaluation may not be reflective of the "real" questions, posed by users (Bernardi and Kirschner, 2010). Our paper uses a subset of the real questions posted by users on Community Question Answering (CQA) sites, and searches and interactions from real users – which makes our task unique and more challenging than the previous settings.

In particular, our work builds on the rich history of using eye tracking technology to identify areas of interest and attention, and to study reading behavior. In the context of web search document examination, Buscher et al. (2008) extracted sub-documents by tracking eye movements as implicit feedback and expanded search queries to improve the search result ranking. Buscher et al. also studied the prediction of salient Web page regions using eye-tracking (Buscher et al., 2009a). This work, and others, have shown that user attention can help identify regions of documents of particular relevance or usefulness for the query. While eye tracking equipment limits the applicability of these findings to lab studies, these studies served as inspiration to our work to detect the *inferred* areas of interest. Specifically, we use mouse cursor tracking as a natural proxy for user's attention, to replace the requirement for eye tracking equipment. As originally reported by Rodden et al. (2008), the authors discovered the coordination between a user's eye movements and mouse movements when scanning a web search results page. This work was further extended by Huang et al. (2012) to predict the gaze position from mouse cursor movement, with mean error of about 150 px. In summary, there is mounting evidence that the user's attention in web search can be approximated using mouse cursor, scrolling, and other interaction data. In particular, Hijikata (2004) proposed a method to extract text passages of Web pages based on the user's mouse activity and found that extracted passages based on mouse ac-

tivity such as *text tracing, link pointing, link clicking and text selection* enable more accurate extraction of key words of interest than using the whole text of the page. Recently, White and Buscher (2012) proposed a method that uses text selections as implicit feedback for document ranking. Most closely related to this work is a contemporaneous effort on improving web search result summaries, or snippets, by exploiting searcher behavior on the examined documents, described by Ageev et al. (2013). However, to the best of our knowledge, there has been no prior work on modeling searcher interaction on result documents to improve Question Answering performance, and in particular the passage retrieval step.

3 Problem Statement and Approach

This section first states the problem we are addressing more precisely. Then, we describe the key parts of our approach (Section 3.2), and the required infrastructure we had to develop to accomplish the required data collection (Section 3.4).

3.1 Problem Statement

Our goal is to incorporate the searcher behavior (in particular, page examination) into passage retrieval. That is, by analyzing the searcher behavior data, we aim to identify the parts of the page that contain relevant passages for answering a question. Specifically, given a question, a set of queries generated by searchers attempting to answer this question, and a set of documents retrieved by a search engine for each of the queries, our goal to retrieve a set of *passages* that contain correct answers for the question.

That is, our goal is to identify, from searcher behavior, the passages in the documents most likely to contain correct answers to a question, which could then be incorporated into a fully automated question answering system, or returned to the user directly, for example, by incorporating these passages into the result abstracts or "snippets".

3.2 Approach

Our approach accomplishes the goal above by incorporating both textual and behavioral evidence. Specifically, we combine together traditional text-based passage retrieval features, and the inferred user interest in specific parts of a document based on searcher behavior.

First, a passage score is obtained from the QA-SYS system (Ng and Kan, 2010), resulting in a strong text-only baseline that generates *candidate passages*. Separately, examination behavior data is collected over the landing pages, using our logging infrastructure described in the next section. Then, a behavior model is trained to identify the passages of interest to the user, based on user examination data (Section 4.2). Finally, the behavior-based prediction of interest in each candidate passage is combined with the original (text-based) passage score, in order to generate the final *behavior-biased* passage ranking (Section 4.3). Note that by decoupling the behavior modeling from the candidate generation method, our approach can be used with any other passage retrieval approach that provides scores for the candidate passages (that could be combined with the behavior scores for the final ranking step).

While general and flexible, our approach makes two key assumptions, resulting in potential limitations. First, our approach is primarily targeted (and evaluated for) informational questions – that is, questions for which the user expects to find an answer in the text of the page. For other question classes (e.g., opinion), passage retrieval might have to be optimized differently. We also assume that the user interactions on landing pages can be collected by a search engine or a third party. This is not far-fetched: already, browser plug-ins and toolbars collect some form of user interactions on web pages, major organizations can (and sometimes do) use proxies, and common page widgets like banner ads and visit counters commonly inject JavaScript to monitor basic user interactions – and can be easily extended to collect the examination data described in this paper. The privacy and security of these methods are beyond the scope of this paper, we merely point out that these behavior gathering tools, assumed by our approach, already exist and are already widely deployed. The interested reader can obtain an overview of the relevant privacy issues and proposed solutions in references (Mayer and Mitchell, 2012; Krishnamurthy and Wills, 2009).

3.3 Acquiring Search Behavior Data

Our infrastructure for acquiring search behavior was developed with two goals in mind: (1) to obtain behavior data similar to real-world search, with the ability to track fine-grained search behavior such as

a mouse cursor movement (as there are no publicly available data of this kind); (2) to create a controlled and clean ground truth set, to train our system and evaluate the effectiveness of our approach.

To collect sufficient amount of search behavior data, we adapted for our task the publicly available UFindIt architecture, described in reference Ageev et al. (2011). The participants played several search contests, or "games", each consisting of 12 search tasks (questions) to solve. The stated goal of the game was to submit the highest possible number of correct answers within the allotted time. After the searcher decided that they found the answer, they were instructed to type the answer together with the supporting URL into the corresponding fields in the game interface. Each search session (for one question) was completed by either submitting an answer or clicking the "skip question" button to pass to the next question.

Participants were recruited through the Amazon Mechanical Turk (MTurk) service. As a first step, the workers had to solve a ReCaptcha puzzle to verify that they are human and not an automated "bot". A browser verification check was performed to confirm that the browser was compatible with our JavaScript tracking code. During the data postprocessing stage, we filtered out the users who did not answer even the easy, trivial questions, as it indicated either poor understanding of the game rules, or an attempt to make a quick buck without effort.

In order to capture all of the participants' search actions, they were instructed to use only our search interface (and not a separate browser window). The search interface performed the web searches using the public API of a popular web search engine, and showed result pages to the users using the original page design, layout and stylesheets, so the user's search experience is not affected.

3.4 Page Examination Behavior Logging

A key part of our system is a mechanism for collecting searcher interactions on web pages, and tying them precisely to the page content at the word level. As the HTML page passed through the proxy, a JavaScript code is embedded to track the user's interactions, including mouse movements and scrolling, as well as the properties of the visited page. The behavioral (interaction) events are logged by the search interface proxy and written to the server log.

To connect the tracked mouse cursor positions to exact text passages we employed the following trick. After the HTML page is rendered in the browser window, our JavaScript code modifies the page DOM tree so that each word is wrapped by a separate DOM Element. Then for each DOM Element, the window coordinates of that element are evaluated and saved in an Element's attribute. The processed HTML page is then saved to the server by an asynchronous request. The saved coordinates are updated if the page layout is changed due to *resize* window event or AJAX action.

As a result of this instrumentation, for each page visit we know the searcher's intent (question), a search engine query that the user issued, a URL and HTML page, the bounding boxes of each word in the HTML text, and all of the searcher actions, e.g., mouse movement coordinates, mouse clicks, and scrolling.

4 Behavior-Biased Passage Retrieval

We now present the details of our behavior-biased passage retrieval algorithm (BePR). First, we describe the text-only retrieval system. Then, we introduce our method for inferring the most interesting or useful parts of the document from user behavior (Section 4.2).

4.1 Text-Based Passage Retrieval

We adopt an open-source question answering framework QANUS (Ng and Kan, 2010) (version v29Nov2012). The QANUS distribution contains the fully functional factoid QA system QA-SYS that we use as a baseline for our experiments. QA-SYS implements many of the state-of-the-art question answering techniques, and is similar to a top-performing QA system from TREC (Sun et al., 2005). The QA-SYS distribution is configured for processing documents and questions in TREC QA format, and we adopted QA-SYS for answer extraction from web documents. QA-SYS takes a set of documents and a question as an input, and processes the input in three stages: (1) information source preparation, (2) question processing, and (3) answer retrieval.

In the first stage, the downloaded HTML pages are pre-processed with Natural Language Tool Kit (NLTK, Bird (2006)). Extracted text is divided into sentences using *Punkt* unsupervised sentence split-

ter (Kiss and Strunk, 2006). The QA-SYS performs Part of Speech tagging using Stanford POS tagger (Toutanova et al., 2003), and Named Entity Recognition using Stanford NER (Finkel et al., 2005), and then builds a Lucene index over the set of input documents. In the second stage the QA-SYS performs POS tagging, NE recognition, and question type classification for an input question.

To answer a question, QA-SYS creates a query from the question, performs the search over the indexed text collection, and retrieves top 50 documents. Each document is split by sentences, and for each sentence a *QA-SYS Passage Retrieval Score* ($TextScore$) is computed as a linear combination of term frequency score, proximity score, and term coverage score. After that 40 passages with the highest $TextScore$ are retrieved, for each passage QA-SYS performs pattern based answer extraction based on the identified expected answer type of the question.

As the focus of this paper is to improve Passage Retrieval performance, we use the $TextScore$ sentence ranking as a baseline, and improve on it by adding the new search behavioral features indicating the passage relevance, as described next.

4.2 Inferring Relevant Passages from Search Behavior

To rank passages by their "interestingness" – that is, to identify the passages that have been carefully examined by the searcher, we use a learning-to-rank approach, and apply regression algorithms to predict the probability that a specific passage is interesting for a user. A passage is labeled as "interesting", if the user submitted an answer in the current session, and both the passage and the answer have at least one common word, after stemming and stop-word removal.

For each passage, a set of behavior features that could represent passage interestingness is created. To associate behavioral features with a given document passage, we match the sequence of behavior events and the set of bounding boxes for each word and DOM Element of a page. For efficiency, we build a spatial R-Tree index of these bounding boxes, which allows us to quickly find the matching DOM Elements for each event.

One key feature is the duration of the time interval when a mouse cursor was hovering over the specific text passage, or very close to the passage. We also take a scrollbar and event count features from papers (Buscher et al., 2009b), and (Guo and Agichtein, 2012) to detect evidence of "reading" vs. "skimming" behavior, and adopt those features to represent the behavior near the specific location of a page. The full set of our passage behavior features are reported in Table 1.

Feature	Description
MouseOverTime	Time duration when the mouse cursor was over the text passage
MouseNearTime	Time duration when the mouse cursor was close to the text passage in the window $(x \pm 100px,\ y \pm 70px)$
MouseOverEvents	The number of mouse events during *MouseOverTime*
MouseNearEvents	The number of mouse events during *MouseNearTime*
DispTime	Time duration when the text passage has been visible in the browser window (depends on scrollbar position)
DispMiddleTime	Time duration when the text passage was visible in the middle part of the browser window

Table 1: Behavior features for text passages

To implement the passage ranker, we experimented with a variety of learning-to-rank (LTR) algorithms, and chose two implementations of Regression Trees, due to their strong performance for general web search ranking tasks. The first algorithm is Regression Tree (Friedman et al., 2001), and the second is Gradient Boosting Regression Tree algorithm (Friedman, 2001). They are named *BePR-BTree*, and *BePR-GBM* respectively.

The dataset consists of a set of questions, with associated search behavior data collected from all the users who tried to find an answer to this question, the answers submitted by the users, and a set of validated answers. These sets are divided into training, validation, and test, so that the training and validation set URLs are disjoint, and the test set have no intersection with training and validation set by URLs, questions, and users. The training set is created from only those page visits where the document text has non-empty intersection with the user's answer, and the answer is correct. The trained regres-

sion algorithm is applied to all page visits in the test set. When the trained model is applied at test time, it has no information about the user's intent, the correct answer, or the current query, but rather uses only the behavioral features of the current page visit to identify the "interesting" passages.

The predicted probability of passage interestingness is averaged over all the users and page visits, and the resulting passage interestingness is then used as the $BScore$ of the passage. Note that $BScore$ is defined for only *visited* pages; to incorporate the overall clickthrough information (i.e., the fraction of the time a page was visited, indicating relevance), we introduce a generalized version, designated as $BSscore_{All}$, defined as: $\gamma \cdot CTR + (1-\gamma) \cdot BScore$, where CTR is the clickthrough rate for the page, defined as the fraction of time the result was clicked for all searches. Intuitively, this version reduces the weight of the behavior score for the pages with insufficient behavior data by "backing off" to the document clickthrough rate, according to the parameter γ. For the cases where only the visited pages are considered (ignoring the searches when the page was not visited), γ is set to 0, reverting the score to the original $BScore$ definition. The resulting behavior-based passage score is then used as the aggregate value of searcher interest in the passage for the combined passage retrieval step, described next.

4.3 Combining Textual and Behavioral Evidence

The final step in our approach is to *combine* the text-based score $TextScore(f)$ for a sentence (Section 4.1) with the interestingness score $BScore(f)$ (Section 4.2), inferred from the examination data. In our current implementation we combine these scores by linear combination:

$$FScore(f) = \lambda \cdot BScore(f) + (1 - \lambda) \cdot TextScore(f)$$

Other more sophisticated ways to combine text and behavior evidence are possible, such as jointly learning over both text and behavior features. However, we chose to follow the simpler linear approach for interpretability of the results (e.g., by varying the λ parameter).

5 Data Collection and Experimental Setup

This section presents the methodology used for selecting the questions (Section 5.1), the corresponding search behavior data (Section 5.2), and the experimental collections and metrics (Section 5.3).

5.1 Questions

The search tasks were selected from community question answering sites such as wiki.answers.com and Yahoo! Answers by the researchers. The criteria used were that the question should be clearly stated, had a clear answer, and that finding this answer was not a trivial task, that is, the answer was not retrieved simply by submitting the question verbatim to Google, Bing, or Yahoo! Search engines. Overall, 36 such questions were selected, posing (as it turned out) greatly varying levels of difficulty for participants. These questions were randomly split into three game rounds of 12 questions each.

5.2 Browsing Behavior Dataset

The search behavior data for each of the questions above was acquired as described in Section 3.3. A total of 270 participants finished the game. After filtering out users who did not follow the game rules, we have 3047 search sessions performed by 265 users. Our data for these users consists of 7800 queries, 3910 unique queries, 8574 SERP clicks on 1544 distinct URLs. For 5683 page visits (66%) and 883 distinct URLs the on-page behavioral data is collected. For the rest 34% of page visits the behavioral data were not collected due to conflicts between our JavaScript tracking code and other code presented on the page. For each page view there are about 400 atomic browsing events (mouse movements, scrolling, key pressing) on average. All the source and derived data are available at `http://ir.mathcs.emory.edu/intent`.

The dataset is divided into training, validation, and test set in the following way. The behavior dataset for the first game is divided randomly into equal-sized training and validation sets that are disjoint by URLs. The training set was used to train the regression algorithm for predicting passage attractiveness, and the validation set was used to explore the influence of behavior weight λ on passage retrieval performance, and to select the parameter λ for using on a test set. The validation set consists of 254 different URLs spread over 11 questions, and

for each of them there is a collected browsing behavior.

The test set consists of 441 URLs spread over 24 questions, and the test set has no intersection with training and validation set by URLs, questions, and users.

5.3 Candidate Document Selection Strategies

The first step for question answering is a selection of a candidate document set. In our settings, we may select a subset of web documents in a different way. We explore passage retrieval effectiveness using three different strategies of document set selection.

- For each question select *All* documents that are in top 10 documents returned by a search engine for any query that was issued during search for the specific question. For our dataset this gives around 500 candidate documents per question on average.

- For each question select only documents that were *Clicked* by a user. This restricts a candidate document set to set of most promising documents. For our dataset this gives around 25 candidate documents per question on average.

- For each pair of question and *Relevant* document apply passage retrieval to the specific document. In this experiment we label a document "Relevant" if a correct answer was extracted from it. In a real-world scenario, while document relevance could be estimated by a variety of click-based methods, we address the challenge of how to actually extract the correct answer from the document, automatically, with the help of the natural behavior data. We perform this experiment to estimate the performance of passage retrieval for the case when relevant documents are known with high confidence.

Evaluation Metrics: We evaluate passage retrieval performance by standard Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP) metrics for top 20 retrieved sentences (Voorhees and Tice, 1999). We also evaluate ROUGE-1 metric (Lin, 2004) for the first retrieved passage.

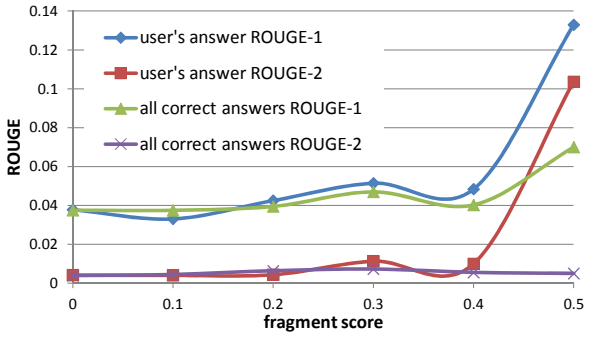


Figure 1: The actual passage interestingness, measured by intersection with user's answer, vs. the passage relevance score $BScore$ predicted from behavior data

6 Results

We now present the empirical results. First, we report the intermediate result of using behavior data to infer the interesting (useful) passages in the document. Then, we report the main results of the paper where the quality of the generated snippets with and without using behavior data is compared using human judgments.

6.1 Prediction of Passage Interestingness

This experiment evaluates how well we can predict interesting passages by observing a user's on-page behavior. We suppose that the passage is interesting if it is related to the answer for the question. For each visited page, we collect the user's answer (if submitted), and all correct answers from all users who answered this question. Then, we compare those answers to each text passage in the document using ROUGE metrics (Lin, 2004).

Figure 1 shows the relationship between the interestingness of a passage and behavior score. The graph shows that when the score is high ($\geq$ 0.5), then average intersection between the passage and user's answer is much higher than those when the passage score is low. All ROUGE-N metrics significantly grow when the behavior score grows, although ROUGE-2 over all correct answers are always very small (it grows from 0.003 to 0.007). ROUGE-1 is much greater than ROUGE-2 for high scores, as the interesting passage might contain useful information for the answer, but the user reformulates the obtained information and submits reformulated answer. The ROUGE-N metrics for a user's answer are much greater than those for all correct

Feature	Feature Importance
DispMiddleTime	0.51
MouseOverTime	0.34
DispTime	0.12
MouseNearTime	0.02
MouseOverEvents	0.01
MouseNearEvents	0.01

Table 2: Feature importance for behavioral features, as measured by Gini coefficient

answers, as other users might obtain valuable information from other documents, and some questions have distinct correct answers.

Behavior Feature Importance Analysis: To estimate relative importance of behavior features we evaluated the Gini importance index (Breiman, 1996) for each behavior feature from the Table 1. The Table 2 shows that the most important features are the time duration when the text passage was visible in the middle part of the scrolling window, and the time duration when the mouse cursor was over the text passage. The first feature has been shown to be a good feature for re-ranking search results in reference (Buscher et al., 2009b), and we have shown that it is also useful for passage retrieval. The *MouseOverTime* feature has been previously shown to be correlated with examination time, measured by eye-tracking experiments (Guo and Agichtein, 2010), and it helps us detect local behavior in the neighborhood of a specific text passage.

Analysis of Searcher Attention: In order to better understand what characteristics of the textual passages attract the searcher's attention, we explored 21 linguistic features for each sentence. Our features were designed to estimate text readability, and the overlap of a passage with the query that was used to find the document. We implemented the readability features from (Kanungo and Orr, 2009), and query matching features from (Metzler and Kanungo, 2008). Table 3 reports the top 10 features with the highest absolute value of the correlation coefficient with passage interestingness score $BScore$. Interestingly, the most highly correlated features are related to readability, while query matching features are less important.

Feature description	*corr*
Number of distinct words in the passage	0.31
Total number of words in the passage	0.28
Number of letter ([a-zA-z]) characters	0.27
Relative location of the passage in the document	-0.25
Number of unique words in the passage divided by total number of words	-0.24
Number of punctuation characters	-0.20
Number of words with first letter capitalized	-0.17
Overlap of query terms expanded with synonyms and the passage	0.15
Absolute count of query terms matched in the passage	0.15
Average position of query term within the passage	-0.14

Table 3: Correlation of passage interestingness $BScore$ with linguistic properties of a sentence

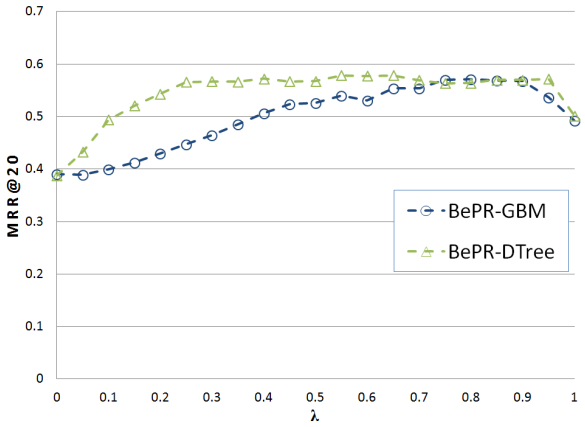


Figure 2: MRR for passage retrieval for varying behavior weight λ and interestingness prediction algorithms *BePR-DTree*, and *BePR-GBM*

6.2 Passage Retrieval with Behavior Data

This section reports the main results of the paper. First, we describe the parameter tuning, followed by the main performance results.

Parameter Tuning: To tune the passage retrieval performance, we use the validation set to find the optimal value for λ. Figure 2 reports the passage retrieval MRR for varying λ, for two learning algorithms BePR-GBM and BePR-DTree. The figure shows that both BePR-GBM and BePR-BTree improve over the QA-SYS baseline. BePR-GBM algorithm achieves the best performance with $\lambda = 0.8$, and also exhibits more robust behavior compared to BePR-BTree, so we use BePR-GBM with $\lambda = 0.8$ for the main experiments described next. Similarly, using the training and validation sets, we optimized

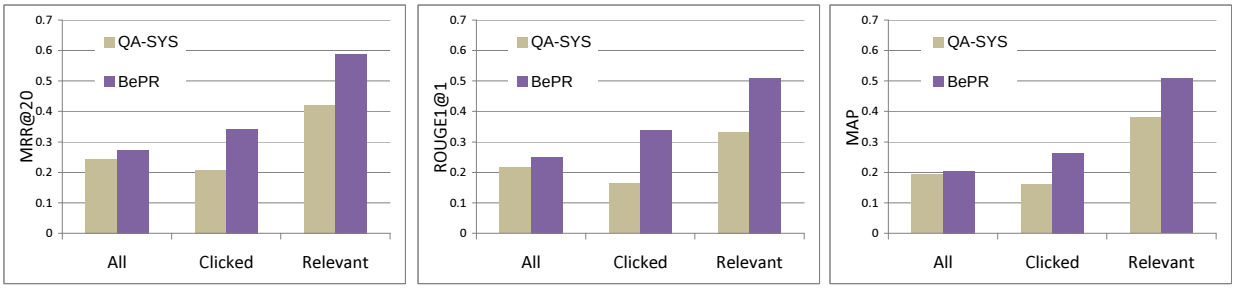


Figure 3: Passage retrieval MRR (a), ROUGE1 (b), and MAP (c) for the BePR and QA-SYS systems, on the test set.

the value of the clickthrough rate weight $\gamma = 0.05$ (used for the $BScore_{All}$ score) for the *All* document set only (as for the *Clicked* and *Relevant* document sets, γ is always set to 0 by construction).

Main retrieval results: We now compare the baseline algorithm for passage retrieval implemented in QA-SYS system and described in section 4.1 with the BePR algorithm (section 4.2-4.3) that combines the textual passage score and the behavior score using the λ parameter for the relative weight of the behavior evidence.

Figure 3 reports the main results of the paper, namely the MRR, ROUGE-1@1 and MAP passage retrieval metrics for the baseline QA-SYS algorithm, and BePR-GBM, on the test set. As the figure shows, BePR achieves higher performance on all metrics, and for all document sets. The improvements are statistically significant ($p < 0.01$) for experiments with Clicked and Relevant document sets. Not surprisingly, the improvements are smallest when *All* documents are considered, as unclicked documents do not provide any associated behavior data. As the results show, our simple back-off strategy (using the document clickthrough rate with the γ parameter) is moderately successful, but could be further refined in the future.

Finally, we illustrate how behavior features affect passage ranking. Let's consider a question "*How many Swedes speak English as a percentage?*". The perfect relevant page for this question is a Wikipedia page "Languages of Sweden". A sentence "Main foreign language(s): English 89%, German 30%, French 11%." contains an answer to the question, but it has only a small intersection with question terms, and QA-SYS ranks this question in the 13th place. Other sentences that contain a country name, a number, or have more terms that match the question are ranked higher. In contrast, as searchers examined this sentence carefully to find the answer, BePR is able to promote this sentence to the second place in the ranking.

7 Resources and Data

All the code and the collected data used in this research are available at `http://ir.mathcs.emory.edu/intent/`. The dataset contains the set of questions used for the experiments, and user's behavior: queries submitted by users to search engine, result pages, visited URLs, downloaded landing pages, on-page browsing behavior (mouse movements, scrollbar events, resize actions, clicks). By sharing our code and data, we hope to encourage further research in this area.

8 Conclusions and Future Work

We presented the first successful approach to incorporating naturalistic searcher behavior data into passage retrieval for question answering. Specifically, we developed a robust method to infer searcher interest in specific parts of the document, which could then be combined with more traditional textual features used for passage retrieval. Our results show significant improvements over a strong baseline, derived from a competitive Question Answering system.

To implement the proposed method in a real-world search engine for Web QA, the proposed infrastructure and/or the released data could be used as a training set for the algorithm that predicts fragment interestingness from user behavior. Such a system would need to track document examination data. This can already be done by incorporating our released tracking code or a similar method into a

browser toolbar, banner ad system, visit counters or other JavaScript widgets that already track user visits. While we acknowledge user privacy as an important concern, it is beyond the scope of this work.

In the future, we plan to extend this work to more precisely pinpoint the answer location on a page, and consequently incorporate searcher behavior into subsequent answer extraction and ranking stages of question answering. We also plan to further investigate the examination data to better understand how searchers find correct (and incorrect) answers using both general web search engines and QA systems – in order to inform and further improve query suggestion, result snippet generation, and result ranking algorithms.

Acknowledgments

This work was supported by the National Science Foundation grant IIS-1018321, the DARPA grant D11AP00269, the Yahoo! Faculty Research Engagement Program, and by the Russian Foundation for Basic Research Grant 12-07-31225.

References

Mikhail Ageev, Qi Guo, Dmitry Lagun, and Eugene Agichtein. 2011. Find it if you can: a game for modeling different types of web search success using interaction data. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 345–354, New York, NY, USA. ACM.

Mikhail Ageev, Dmitry Lagun, and Eugene Agichtein. 2013. Improving search result summaries by using searcher behavior data. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '13.

Elif Aktolga, James Allan, and David A. Smith. 2011. Passage reranking for question answering using syntactic structures and answer types. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 617–628, Berlin, Heidelberg. Springer-Verlag.

Raffaella Bernardi and Manuel Kirschner. 2010. From artificial questions to real user interaction logs: Real challenges for interactive question answering systems. In *Proceedings of Workshop on Web Logs and Question Answering*, pages 8–15.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Leo Breiman. 1996. Bagging predictors. *Mach. Learn.*, 24(2):123–140.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the askmsr question-answering system. In *Proc. of ACL*, EMNLP '02, pages 257–264, Stroudsburg, PA, USA. Association for Computational Linguistics.

Georg Buscher, Andreas Dengel, and Ludger van Elst. 2008. Query expansion using gaze-based feedback on the subdocument level. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 387–394, New York, NY, USA. ACM.

Georg Buscher, Edward Cutrell, and Meredith Ringel Morris. 2009a. What do you see when you're surfing?: using eye tracking to predict salient regions of web pages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 21–30. ACM.

Georg Buscher, Ludger van Elst, and Andreas Dengel. 2009b. Segment-level display time as implicit feedback: a comparison to eye tracking. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 67–74, New York, NY, USA. ACM.

Charles Clarke, Gordon Cormack, Derek Kisman, and Thomas Lynam. 2000. Question answering by passage selection (multitext experiments for trec-9). In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 400–407, New York, NY, USA. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*, volume 1. Springer Series in Statistics.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):pp. 1189–1232.

Qi Guo and Eugene Agichtein. 2010. Towards predicting web searcher gaze position from mouse movements. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 3601–3606, New York, NY, USA. ACM.

Qi Guo and Eugene Agichtein. 2012. Beyond dwell time: estimating document relevance from cursor movements and other post-click searcher behavior. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 569–578, New York, NY, USA. ACM.

Sanda Harabagiu, Dan Moldovan, Christine Clark, Mitchell Bowden, Andrew Hickl, and Patrick Wang. 2005. Employing two question answering systems in trec-2005. In *Proceedings of the fourteenth text retrieval conference*.

Yoshinori Hijikata. 2004. Implicit user profiling for on demand relevance feedback. In *Proceedings of the 9th international conference on Intelligent user interfaces*, IUI '04, pages 198–205, New York, NY, USA. ACM.

Jeff Huang, Ryen White, and Georg Buscher. 2012. User see, user point: gaze and cursor alignment in web search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1341–1350, New York, NY, USA. ACM.

Tapas Kanungo and David Orr. 2009. Predicting the readability of short web summaries. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 202–211, New York, NY, USA. ACM.

Diane Kelly and Jimmy Lin. 2007. Overview of the trec 2006 ciqa task. In *ACM SIGIR Forum*, volume 41, pages 107–116. ACM.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Comput. Linguist.*, 32(4):485–525, December.

Balachander Krishnamurthy and Craig Wills. 2009. Privacy diffusion on the web: a longitudinal perspective. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 541–550, New York, NY, USA. ACM.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. Barcelona, Spain, July. Association for Computational Linguistics.

Jonathan R. Mayer and John C. Mitchell. 2012. Third-party web tracking: Policy and technology. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 413–427, Washington, DC, USA. IEEE Computer Society.

D. Metzler and T. Kanungo. 2008. Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*.

Jun-Ping Ng and Min-Yen Kan. 2010. Qanus: An open-source question-answering platform http://www.comp.nus.edu.sg/ junping/docs/qanus.pdf.

Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. 2008. Eye-mouse coordination patterns on web search results pages. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 2997–3002, New York, NY, USA. ACM.

Renxu Sun, Jing Jiang, Yee Fan Tan, Hang Cui, Tat-Seng Chua, and Min-Yen Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *TREC*.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 41–47, New York, NY, USA. ACM.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ellen Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *Proceedings of The Eighth Text REtrieval Conference (TREC-8), http://trec. nist. gov/pubs/trec8/t8_proceedings. html*.

Ryen W. White and Georg Buscher. 2012. Text selections as implicit relevance feedback. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1151–1152, New York, NY, USA. ACM.

Assembling the Kazakh Language Corpus

Olzhas Makhambetov, Aibek Makazhanov, Zhandos Yessenbayev,
Bakhyt Matkarimov, Islam Sabyrgaliyev, and Anuar Sharafudinov
Nazarbayev University
Research and Innovation System
53 Kabanbay batyr ave., Astana, Kazakhstan
{omakhambetov, aibek.makazhanov, zhyessenbayev, bmatkarimov,
islam.sabyrgaliyev, anuar.sharaphudinov}@nu.edu.kz

Abstract

This paper presents the Kazakh Language Corpus (KLC), which is one of the first attempts made within a local research community to assemble a Kazakh corpus. KLC is designed to be a large scale corpus containing over 135 million words and conveying five stylistic genres: literary, publicistic, official, scientific and informal. Along with its primary part KLC comprises such parts as: (i) annotated sub-corpus, containing segmented documents encoded in the eXtensible Markup Language (XML) that marks complete morphological, syntactic, and structural characteristics of texts; (ii) as well as a sub-corpus with the annotated speech data. KLC has a web-based corpus management system that helps to navigate the data and retrieve necessary information. KLC is also open for contributors, who are willing to make suggestions, donate texts and help with annotation of existing materials.

1 Introduction

This article describes theoretical and practical issues experienced during the construction of the Kazakh Language Corpus. Kazakh is an agglutinative and highly inflected language which belongs to the Turkic group. It is official state language of Kazakhstan and a mother tongue for more than 10 million people all around the world. However, up until the early 90's of 20th century, due to historical reasons of the Soviet era, Russian language was the predominant language in spoken and written communication in Kazakhstan. This fact in turn caused the problem of underrepresentation of Kazakh language in various fields such as science, entertainment, official documentation, etc. For this reason, while assembling the corpus, we had to group categories that are generally presented as separate in other corpora into five stylistic genres. Also, in contrast to other corpora (Aksan et al., 2012; Chen, 1996), we included texts as they were available, i.e we did not try to fill a predefined set of categories. Substantial part of materials was collected using source-customized web crawlers and donated texts.

KLC also contains a manually annotated sub-corpus with morpho-syntactic and structural markups encoded in XML following general notions outlined in CES (Ide, 1998). Our syntactic tagset comprises a set of syntactic categories well-defined in a classical Kazakh grammar, and the part of speech (POS) tagset is based on a positional system in which the tags are formed by concatenation of POS of a word form and a chain of encoded linguistic properties, such as number, case, voice etc. The annotations have been carried out manually by philology students specializing in morphology and syntax. Trying to make the annotation process as comfortable as possible, we have designed a web-based annotation tool with a user-friendly interface. We took a great care for the annotation quality, and to do that we (i) arranged the validation process, and (ii) equipped the tool with a recommendation system that, as we will show, improves the inter-annotator agreement.

As a part of KLC we have also compiled the annotated read-speech corpus (RSC), which includes audio recordings of words, phrases, sentences (from all genres), news articles and excerpts from books, that were carefully chosen from the primary part of the corpus. All text materials were read by volunteers who represented different age, gender, region and education backgrounds in a balanced way. Each audio file is accompanied with a label file and a corresponding text transcript. Moreover, some of the transcripts have been grammatically annotated, i.e. in addition to a word-level segmentation of audio information a portion of our data has lexical, and morpho-syntactic annotations. In total RSC contains 10GB or more than 40 hours of speech.

This paper is organized as follows. Section 2 reviews the existing work. Section 3 provides detailed information about the primary corpus. Sections 4 and 5 thoroughly describe annotated text and speech sub-corpora respectively. Finally, we draw conclusions and discuss future work in Section 6.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1022–1031,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Since the pioneering corpus of Brown University was completed in 1964 by Francis and Kučera (1979), corpus linguistics has become a thriving research field. Over the past two decades researchers all around the world released many corpora, including well known British National Corpus (BNC) (Burnard, 2007) developed between 1991 and 1994, and containing more than 100 million words of written and spoken language from a wide range of sources (Ide and Macleod, 2001; Al-Sulaiti and Atwell, 2006). All materials were selected on a basis of three independent criteria (medium, domain and time), where each criterion had predefined target proportions. The spoken part (remaining 10%) consists of orthographic transcriptions of unscripted informal conversations and spoken language collected in different contexts. BNC is tagged for part of speech (POS) using the CLAWS4 (Constituent Likelihood Automatic Word-tagging System) (Leech et al., 1994) tagging system developed at Lancaster University. BNC is generally accepted as a balanced corpus, and many researchers, such as the creators of Turkish National Corpus (Aksan et al., 2012), Korean National Corpus (Kim, 2006) etc., adopted it as a model for compiling their own corpora.

The Russian National Corpus (RNC) has been released by the group of specialists from different organizations led by the Institute of Russian language, Russian Academy of Sciences (Ruscorpora, 2003). The corpus covers primarily a period from the middle of the XVIII to the early XXI centuries. It includes both written texts (fiction, memoirs, science, religious literature and others) and recorded spoken data (public speeches and private conversations). Currently RNC contains over 350 million word forms that are automatically POS-tagged and lemmatized. The corpus also includes semantic tags for words and texts (Apresjan et al., 2006). Along with its main part, RNC contains such subcorpora as: Deeply Annotated Corpus, that contains sentences with a complete morphological and syntax structure markup, where the syntax structure is largely based on the Meaning-Text Theory introduced by Aleksandr Žolkovskij and Igor Mel'čuk; English – Russian, German – Russian, Ukrainian – Russian, Belorussian – Russian parallel corpora; Dialect corpus; Poetry corpus and others.

Unfortunately, up until now, not too much work has been accomplished in developing a corpus that will represent Kazakh language. To the best of our knowledge there has been a limited number of attempts to compile one, but resulting corpora are too small in size and scope, or not available to the public. A Kazakh corpus has been initiated by the Committee on Languages of the Ministry of Culture of the Republic of Kazakhstan (CLMCRK, 2009). This corpus is small in size and not annotated, as it remains in its very early stage of development. The new sub-corpus for Kazakh has been recently built by Baisa et al. as a part of larger corpus of Turkic languages (Baisa and Suchomel, 2012). This corpus was compiled using a web crawler that selected texts based on a language model trained on Wikipedia texts. Although the obtained corpus is relatively large in size, the data was not categorized by genres. Also, since a crawler was not source-customized, the corpus may contain some noise coming in the form of text in Russian or other languages. We also could not find enough information about a Kazakh corpus that has been developed at Xinjiang University and used in their research (Altenbek and Xiao-long, 2010). The absence of an available corpus that will be large enough to represent Kazakh language decelerates many research activities (Mukan, 2012). We believe that building an open Kazakh corpus will have a significant impact and it will be very useful tool in the analysis of Kazakh.

3 KLC Primary Corpus

KLC is one of the first attempts to build a large scale, general purpose corpus that represents the present state of Kazakh language. Currently, the size of the primary corpus is more than 135 million words and it contains approximately more than 400 000 documents classified by genres into the following five sections: (1) *literary* section contains Kazakh literary texts that were published in the range from the beginning of the XX century till present; (2) *official* section includes mainly official statutes, orders, acts and other materials produced by the governmental organizations within the period of 2009-2012; (3) *scientific* section includes books, research monographs, dissertations, articles and essays from various fields (informatics, biology, chemistry, etc.); (4) *publicistic* section contains periodicals and articles from online sources, i.e. newspapers and magazines published over the last ten years; (5) *informal language* section includes documents with colloquial Kazakh texts extracted from the popular blog platforms starting from 2009. We have to note that while compiling this corpus we intentionally relaxed the document selection criteria by not restricting the collected data to particular domains, media, and time. This was mainly dictated by the lack of materials, and partially due to the reasons mentioned in the introduction.

Our main sources of data were Internet websites as well as digitized forms of books, dissertations and articles from public and personal libraries. For each website we designed a source-specific crawler, thereby increasing the precision of the meta data (e.g. authors, news categories, etc.) extraction. Additionally, we filtered out documents with a high consistency of Russian texts by aligning them to a language model trained on pure Russian texts. We also filtered out all documents with the size

Genre	# docs	# all words	# unique words
Literary	8 255	7 733 456	423 445
Publicistic	404 884	79 302 154	951 659
Official	25 302	44 670 856	335 264
Scientific	527	2 227 878	153 877
Informal	6 110	1 337 953	162 074
TOTAL	**445 078**	**135 272 297**	**1 365 202**

Table 1: A quantitative description of the corpus.

documents, total	1213
documents, %	0.3
all words, total	613 511
all words, %	0.4
unique words, total	80 368
unique words, %	5.9
lemmata, total	42 901

Table 2: A quantitative description of the annotated data

less than 1kB. It took about 7 months to grow the corpus to its current size. Table 1 provides a general quantitative description of the corpus.

We release the data under a license that in accordance with Kazakhstan's law allows distribution of some materials in whole (official documents, news articles) and some only in part (literature, scientific texts, analytics) provided that sources are properly cited. This license does not allow printed or electronic publications or similar use of substantial portions of text drawn from the corpus without the permission of its original publisher(s) or copyright holder(s).

3.1 Text Documents Description

Each document is stored in a plain text format in the UTF-8 encoding. Documents contain both the content and the meta-data in a single file, and have the following simple structure:

- TITLE – the title of a document;
- SOURCE – the source of a document
- AUTHOR – the author(s) of a document;
- DATE – the date when a document was published;
- META – additional information;
- TEXT – the content of a document.

Provided that the corresponding information is present in a source, the <META> tag contains both the name of the section of the corpus to which a document belongs and a further categorical sub-division, such as the type of a literary work, e.g. a poem. That is, whenever possible such categories are assigned automatically, e.g. some websites provide this information. For sources that lack meta data, such as the digitized books, dissertations and scientific papers, the corresponding categories (informatics, biology, chemistry, etc.) are assigned manually.

3.2 Writing System of Kazakh language

Kazakh adopts different writing systems depending on the regions where it is spoken (Cyrillic alphabet in Kazakhstan, Arabic and Latin graphics in other countries). Recently the government of Kazakhstan has decided to adopt Kazakh alphabet to a Latin graphic. In this regard we believe that KLC could become a valuable tool. Indeed, we have already provided a group working on this problem with statistical information about letter distributions in Kazakh texts. This information could also aid in designing various speech corpora as well as a proper Kazakh keyboard layout. It can be stated that the latter was done rather carelessly just as a simple adjustment to a Russian keyboard (Wikipedia, 2012). Current Kazakh Cyrillic alphabet consists of 42 letters, whereas 9 of them are pure Kazakh letters and the others adopt the Russian symbolic. Figure 1 shows the distribution of Kazakh letters in the corpus. It can be seen that there is a small non-zero distribution of pure Russian letters (underlined). This can be explained by the ineluctable use of Russian words due to the lack of a proper translation or inheritance of Russian vocabulary.

4 The Annotated Sub-corpus

In order to enhance the effectiveness of the corpus as a research tool, we have annotated a portion of the data for syntactic and POS tags, lemmata, and for morpheme types and boundaries. Table 2 provides net amount and the percentages (with respect to the current size of the corpus) of the annotated data in terms of documents, words, unique words, and lemmata.

The annotation process has been carried out completely manually. We favored a manual annotation over a semi-automatic one, for the following two reasons: (i) finding language independent tools (not to mention Kazakh-specific) which support a fine grained level of annotation that we employ turned out to be rather challenging; (ii) though we refused and partially could not afford a semi-automatic annotation we provided the annotators with a *semi-automatic-like annotation* experience by equipping our annotation tool with a fairly advanced recommendation system. The annotation was performed mainly by the undergraduate students majoring in Kazakh philology. As a quality control measure, two validators (a graduate student majoring in Kazakh philology and one of the authors) were assigned to check a random sample of about 10% of the annotated data. Validators did not just fix errors, we also held regular "work-through-errors" sessions in an attempt to synchronize annotations. Our analysis of validated data suggest that the annotation

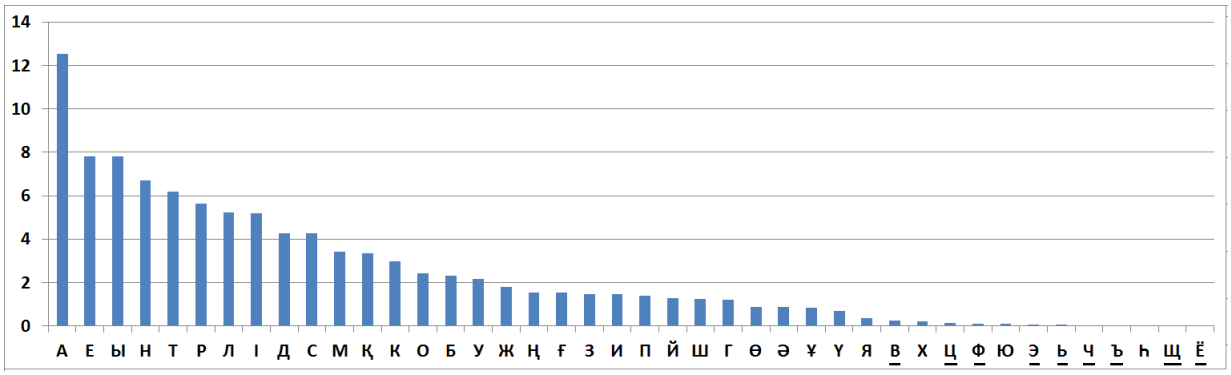


Figure 1: The distribution of letters across the corpus, in %.

Tag	Description	PTB equivalents
S	Simple declarative clause	S
BSS	Independent clause	S
BGS	Dependent clause	SBAR(Q)
BAS	Subject	NP
BND	Predicate	VP
TOL	Object	(WH)NP
ANT	Modifier	ADJP
PYS	Adverbial	(WH)ADVP (WH)PP
X	Void, unknown, uncertain	X

Table 3: The syntactic tagset description

#	Linguistic property	Code	Cardinality
1	Animacy	A	2
2	Number	N	2
3	Possessiveness	S	10
4	Person	P	8
5	Case	C	7
6	Negation	G	2
7	Tense	T	3
8	Mood	M	4
9	Voice	V	5

Table 4: Linguistic properties considered in the POS tagset design

quality was fairly high, as roughly only 6% of annotated tokens were fixed.

To the best of our knowledge this is the first attempt to annotate Kazakh texts with various linguistic markups. Given this, in the following subsections we would like to describe the tagsets (syntactic and POS), the annotation scheme (the format in which the annotated data is stored and distributed), and the annotation tool itself.

4.1 Designing the Tagsets

The syntactic tagset. At the initial stage of the corpus development we did not plan to build a detailed treebank, leaving this task for the future work. Therefore, our syntactic tagset comprises a compact set of syntactic categories well-defined in a classical grammar. Table 3 contains the tagset description along with the equivalent tags defined in a widely used Penn Treebank (Marcus et al., 1993) tagset[1]. In addition to that, we also label proverbials which are rather common elements of Kazakh language. We do not treat them as a separate syntactic category, for they typically serve as a single syntactic unit (e.g. predicate, adverbial, clause, etc.) Instead each syntactic tag has a corresponding binary property that marks the proverbial case.

The POS tagset. Kazakh is an agglutinative Turkic language, in which word forms are generated by means of the affix inflection. Different affix types mark different linguistic properties. For instance, consider a translation of a simple Kazakh sentence:

Mektepke bardym. - school.Dat go.Past.1sg - I went to school.

In this example pronoun *"I"* and preposition *"to"* are "hidden" in the affixes of case and person, i.e.:

Mektep(NN = *a school*) + *ke* (dative case = *to school*)

bar(VB, imperative = *go*) + *dy* (past tense = *he/she went*) + *m* (1st person = *I went*)

As the example shows, inflected affix chains contain important information that is not always present in the context, hence a tagset should be designed in a way to capture this information to the extent possible. For this reason, we design a positional tagset (Oflazer et al., 2003; Hajič and Hladká, 1998; Hana and Feldman, 2010), in which the final tags are constructed by the concatenation of the basic tag (often POS of a word form) and the en-

[1]For ease of presentation we used bracketing instead of listing, i.e. SBAR(Q) should be read as SBARQ, SBAR; (WH)NP as WHNP, NP; etc.

#	Tag	Description	LPs	Cap.	#	Tag	Description	LPs	Cap.
		Noun:					**Pronoun:**		
1	ZEP	*non-personal*	ANSPC	314	20	SIMZ	*personal*[3]	NSPC	229
2	ZEQ	*personal*	ANSPC	314	21	SIMU	*demonstrative*	NSPC	157
		Verb:			22	SIMS	*interrogative*	NSPC	157
3	ET	*regular*	GTMVP	840	23	SIMD	*reflexive*	NSPC	157
4	ETU	*infinitive*	GSC	196	24	SIMB	*indefinite*	NSPC	157
5	ETK	*auxiliary*	P	8	25	SIMY	*indefinite, negative*	NSPC	157
6	ETB	*auxiliary, negative*	P	8	26	SIMP	*indefinite, universal*	NSPC	157
7	KEL	*auxiliary, desiderative*	GT	6			**Adposition:**		
8	ESM	*present participle*	GNSPC	314	27	KOM	*auxiliary nominal*	C	7
9	KSE	*past participle*	G	2	28	SHS	*preposition*	-	1
		Adjective:			29	SHZ	*conjunction*	-	1
10	SE	*regular*	P	8	30	SHD	*particle*	-	1
11	SES	*comparative*	P	8			**Interjection:**		
12	SEA	*superlative*	P	8	31	OSP	*vocative*	-	1
		Numeral:			32	OSQ	*thought*	-	1
13	SN	*cardinal*	NSPC	157	33	OSO	*emotion*	-	1
14	SNR	*ordinal*	NSPC	157					
15	SNZ	*collective*	NSPC	157	34	ELK	**Onomatopoeia**	-	1
16	SNB	*fraction*	NSPC	157	35	MOD	**Modal word**	-	1
		Adverb:							
17	US	*regular*	-	1	36	BOS	*foreign word*	-	1
18	USS	*comparative*	-	1					
19	USA	*superlative*	-	1			**Total capacity:**	-	**3844**

Table 5: The POS tagset description

coded chains of linguistic properties (LPs). Table 4 contains main LPs defined in Kazakh grammar along with their codes and *cardinalities*, i.e. a number of values they accept. Although integrating a rich set of LPs may considerably enlarge the size of a tagset, we tried to consider as many LPs as possible for the following two reasons: (i) previous research shows that increasing the size of a tagset does not necessarily decrease the tagging accuracy (Elworthy, 1995) and that for agglutinative languages omitting grammatical aspects may hurt the accuracy of *n*-gram tagging (Feldman, 2008); (ii) it is easier to reduce a detailed tagset than to re-annotate data for the missed information. Table 5 provides a detailed description of the designed tagset (not including punctuation) both qualitatively and quantitatively. The table contains a list of tags grouped by the ten major POS (in bold). For each tag we provide a set of LPs it accepts and *generative capacities*, i.e. the upper bound on a number of possible tags that can be generated from a given basic tag and the different combinations of the corresponding LPs[2]. The list of 36 basic tags was compiled following the best practices of Penn tagset design (Marcus et al., 1993), and bearing in mind the specifics of Kazakh grammar. Particularly, we broke down the major POS categories in sub-categories, in order to capture semantic distinctions and various usage patterns. For instance, negative (tag #6) and desiderative (tag #7) auxiliary verbs in conjunction with main verbs are used to mark uninflected negation (via *no* and *not*) and desiderative mood construction (via usage of *to come* in the meaning of *to want*) respectively. Similarly, auxiliary nominals (tag #27) are used as prepositional phrases such as, *in front of*, *at the top of*, etc. Also, apart from the ordinal and cardinal numerals we distinguish *collectives* (tag #15), that are used to emphasize completeness of quantities as in *both*, *all three*, etc.; and *fractions* (tag #16) as in *half*, *quarter*, etc. Finally, following classical Kazakh grammar, we treat onomatopoeias (tag #34), i.e. sound imitations as in *tic-tac* or *knock-knock*, as a distinct part of speech.

The *maximum* size of the tagset equals to the total generative capacity, or 3844 tags. However, depending on the

[2]The multiplication of cardinalities of LPs does not always give the exact number of possible tags, for there are rules that restrict certain combinations of LPs. Moreover, some LP combinations may be technically valid but semantically incorrect as they would make no sense, e.g. *bala + m + myn - I am a son of my son*. Where possible we tried to account for such exceptions, checking the combinations and providing exact numbers.

[3]Unlike any other part of speech that accepts the NSPC LP chain and must be in the third person (singular or plural) to be in any case other than nominative, personal pronouns can be in any case for any person, thus having a larger capacity.

morpheme → *token*	*direct speech* → *sentence*
token → *syntactic unit*	*sentence* → *direct speech*
syntactic unit → *sentence*	*list item* → *sentence*
sentence → *paragraph*	*sentence* → *list item*
paragraph → *chapter*	*dialog* → *sentence*
chapter → *document*	*sentence* → *dialog*

Figure 2: Structural markup hierarchy

level of granularity required for an application, some or even all LPs may be dropped or added back in, providing additional flexibility. Even the minimal tagset of 36 basic tags can be further reduced to a universal tagset (Petrov et al., 2011) that consists of 11 tags, with the first seven major POS groups being mapped to their direct equivalents, and the latter four (Interjection through Foreign word) being mapped to the *catch all* category.

Lastly, given the designed tagset the aforementioned Kazakh sentence can be tagged as follows:

Mektepke/ZEP_A0N0S0P3C3 (ZEP - non-personal noun; A0 - inanimate; N0 - singular; S0 - no possessor; P3 - 3rd person; C3 - dative case) ***bardym***/ET_G0T3M1V0P1 (ET - regular verb; G0 - not negated; T3 - past tense; M1 - indicative mood; V0 - active voice; P1 - 1st person) **./.**

4.2 The Annotation Scheme

We have developed an XML-based annotation scheme that follows paradigms of the CES (Ide, 1998) and is convertible into the XCES standard (Ide et al., 2000). The main difference with the latter is that in our scheme the raw text and all markup types (i.e. lexical, syntactic and structural annotation; cf. Section 4) are stored in a single document. For the morpho-lexical and syntactic markups we have corresponding tags, i.e. <TOK> - *token* and <SU> - *syntactic unit*, respectively. Main linguistic characteristics, such as POS, lemmata, morpheme segmentation and syntactic labels are marked through the corresponding sub-tags and properties. All the aforementioned tags have their place in the global hierarchy of the structural markup. In turn, this hierarchy is integrated into the structure of an XML document itself. Figure 2 shows the schematic representation of the developed structural markup. A statement *A* → *B* represents *"A is contained by B"* relation.

4.3 The Annotation Tool

To ease the process of annotation we have developed a special tool that was designed as a web application with a logging and a document management system. The tool allows for (auto)saving current work and reviewing and revising the already annotated documents.

Functionality-wise the tool consists of the following three modules: (1) the syntactic module is designed to parse sentences using the syntactic tagset described in subsection 4.1, and to simultaneously mark sentence boundaries; (2) the morpho-lexical module is designed to perform a morphological analysis, and to comprise such functionalities as morpheme segmentation, POS tagging and lemma identification; (3) finally, the structural module is designed to mark up the logical structure of a document, i.e. paragraphs, dialogues, direct speech, lists, etc. Annotation of a given document is performed in the order in which we described the modules. The decision on such an order, as many other major design decisions, was made accounting for the annotators' feedback, suggestions and requests, thus making the annotation experience as convenient as possible. The validators have almost identical interface with additional functionality, such as a quick look up and correction of word-level (morphology) and sentence-level (syntax) markups. Also, both the validators and the annotators have means to correct orthography and punctuation. However, the originals of each and every annotated document are kept. In fact we have already collected data on misspellings to use in our ongoing research in spelling correction.

	Before	After
inter-annotator agreement	0.81	0.84
average MAE	0.08	0.07
average speed, words/hour	212.1	322.6

Table 6: Various characteristics of the annotation process before and after introducing the recommendation system

We have also developed a recommendation system for morphological analysis based on the already annotated data. For a given word or a given morpheme, the system generates a list of recommended markups ordered by the decreasing frequency of the previous usage. While this approach arguably has a potential to propagate errors, our experiments suggest the opposite. We have measured the inter-annotator agreement, average mean absolute error (MAE), and the annotation speed with and without the recommendation system. All measurements were taken for five annotators, who had been working with the tool for two weeks. For the experiments the annotators were given a randomly chosen news article containing about 300 words. The agreement was calculated using Fleiss' kappa (Fleiss and others, 1971). The average MAE was calculated as

$$\Delta MAE = \frac{1}{|A|}\sum_{a \in A}\frac{W - C_a}{W}$$

where A is a set of annotators, W is a number of words in a test document, and C_a is a number of words correctly annotated by the annotator a. The golden truth annotation was provided by the validators. The comparison of the measurements is given in Table 6. As we can see

Age group	I		II		III		IV		
Region	F1	M1	F2	M2	F3	M3	F4	M4	**Sum**
1	3	3	2	1	2	1	2	1	**15**
2	2	3	2	1			2	1	**11**
3	1	1	2	3	2	1	1		**11**
4	3	2		1		1			**7**
5	2	2	2	1	2	2	2	1	**14**
6	2	2	2	2	2		1	2	**13**
7	2	2	1	2	2		2	1	**12**
8	2	1	1	2	1	1	2	1	**11**
9	3	2	2	1	3	1	1	1	**14**
10	1	1	2	2	1	1	2	1	**11**
11	2	1	2	1	1		2		**9**
12	2	2	2		2	1	2	1	**12**
13	2	2	2	1	1	1	1	1	**11**
14	2	1	1	1	1	2	1	2	**11**
15	1	3		1	2				**7**
Total	**30**	**28**	**23**	**20**	**22**	**12**	**21**	**13**	**169**
Age group, %	**35%**		**25%**		**20%**		**20%**		

Table 7: The distribution of the speakers.

the inter-annotator agreement improves with the incorporation of the recommendations, while, in contrast to the error propagation assumption, the error rate slightly decreases. Moreover, we get more than 100 words/hour increase in the labeling speed. Thus, we conclude that as long as the quality of the already annotated data is high, the recommendation system will help to produce quality annotations at a higher speed. One can argue that we used a rather small sample of data to evaluate our recommendation system. However, we drew conclusions not only from the experimental results but also from opinions of validators, who confirmed that they noticed that after integrating the recommendation system annotations grew more coherent and synchronized.

Finally, let us provide a brief technical description of the tool. The design and structure of the front end is based on HTML5 and CSS3. We also use JQuery for HTML elements manipulation and various event handling. The tool can be tried out at `http://kazcorpus.kz/klcweb/annotated/#annotsample`, and the detailed information about it can be found at `http://kazcorpus.kz/klcweb/annotated/#annotdemo`.

5 Read Speech Corpus

Most of the modern speech processing systems require a large amount of audio and text data for training acoustic and language models. Depending on the type of an application required data varies from high quality microphone read speech (Garofalo et al., 2007) to conversational telephone speech (Godfrey and Holliman, 1997; Canavan and Zipperlen, 1996), from continuous speech (Garofolo et al., 1993) to connected (Leonard and Doddington, 1993) and isolated words (Pitrelli et al., 1995). In our current work, we collected a corpus of more than 40 hours of high quality microphone read Kazakh speech of 169 native speakers for the large vocabulary continuous speech recognition tasks.

5.1 Text Materials

The text materials to be uttered were carefully selected from the primary section of the corpus and divided into two parts: *sentences* and *stories*. The "sentences" part has more than 12 000 different sentences randomly and equally extracted from all of the five genre specific sections of the corpus. The sentences are chosen so that in total they contain more than 120 000 words which belong to the set of the most frequent words that cover the 95% of all the texts in the corpus. Additionally, the sentences were grouped according to their length in words. Thus, we have ten groups of sentences, so that the first group contains the sentences of length six, the second – of length seven, and so on up until the length of 15.

The "stories" part contains short online news extracted from publicistic genre section of the corpus. Each story consists of up to 300 words. All the materials were subdivided into non-intersecting sets of texts and distributed among the speakers in the following manner. Each speaker was assigned exactly 75 sentences and one story. Of the 75 sentences 50 belonged to the first five

"short-sentenced" groups (10 sentences per each group), and the remaining 25 belonged to the last five "long-sentenced" groups (5 sentences per each group).

5.2 Speakers

The main criteria of a speaker selection were the following: a region where (s)he learned Kazakh or spent most of his/her life; age; gender; and the ability to read Kazakh.

The first criterion helped us to capture various accents attributed to speakers' settlement both local and external. From the regional perspective we divide the speakers into 15 groups: 14 domestic (one per each administrative region, i.e. "oblast", of Kazakhstan) and one abroad (all foreign countries). Furthermore, the speakers are divided into the following four age groups (not including children and school students): (i) 18-27 years, (ii) 28-37 years, (iii) 38-47 years, (iv) 48 years and above. We did not strictly balance the speakers by their gender due to the difficulties in finding the volunteers, but still tried to choose no more than three speakers of the same gender per one age-regional group. A female-to-male distribution of speakers is 57% to 43%, respectively.

The other important criterion is the ability to read Kazakh, since not all of the interviewees could read in Kazakh sufficiently fluent, which is a common issue in a bilingual country such as Kazakhstan. Additionally, we kept a record of the speakers' education, i.e. whether they attended and graduated from a university, or graduated from a school or a college without attending any universities.

The speakers were encoded using the following scheme: <Region><Gender><Year of birth><Initials><Education>, where "Region" holds the values in the range of [1-15], "Gender" – F or M, "Year of birth" – the last two digits of a year of birth, "Initials" – initials of a name followed by a surname, "Education" – 1 for school, 2 for college, and 3 for university, e.g. 06F70ZK3.

In total, we have recorded 169 speakers. Table 7 presents a distribution of the speakers across the age, gender and regional groups. The blank spots show the speaker profiles that we could not recruit. Mostly, these cases correspond to the distant regions and elder male groups.

5.3 Recording Setup

The actual recording sessions took place in a sound-proof studio of the university with the assistance of a sound operator. Before the recordings, the speakers were instructed, documented and given some time to prepare, as well as asked to fill in the copyright transfer form for the audio data with their voice. They were not constrained on the manner, speed or time except for the correctness of reading. The average time for a recording session per speaker was about 40-45 minutes, though there were cases that lasted for two hours. Audio data was captured using a professional vocal microphone Neumann TLM 49 and digitized by LEXICON I-ONIX U82S sound card. The format of the recorded audio files is 44.1 kHz 16-bit PCM-encoded mono WAVE file format. All the recorded audio files were manually post-processed to have each utterance (sentences and stories) in a separate file and in the corresponding directories. The size of the speech corpus is about 8.5 GB on disk. A collective duration of the audio files is more than 40 hours long.

Letter	ASCII version	Letter	ASCII version
А	a	П	p
Ә	Ae	Р	r
Б	b	С	s
В	v	Т	t
Г	g	У	u
Ғ	Gh	Ұ	Ue
Д	d	Ү	Uu
Е	e	Ф	f
Ё	Jo	Х	x
Ж	Zh	Һ	h
З	z	Ц	c
И	Ij	Ч	Ch
Й	j	Ш	Sh
К	k	Щ	W'
Қ	q	Ъ	"
Л	l	Ы	y
М	m	І	i
Н	n	Ь	'
Ң	Ng	Э	3
О	o	Ю	Ju
Ө	Oe	Я	Ja
#	pause		

Figure 3: ASCII version of the Kazakh letters.

5.4 Transcription and Annotation

Each audio file is provided with its corresponding orthographic transcription and TIMIT-style word-level segmentation, as well as morpho-syntactic annotation files. Both the transcript generation and the annotation were performed manually by trained linguists. The transcription files contain the exact orthographic transcriptions of the utterances, which may differ from the original text. For example, the numbers, abbreviation, foreign words and dates are expanded depending on how they were uttered by the speakers. In addition, the transcription of the stories have the sentence boundaries labeled with <s> and </s> tags. For the segmentation we used WaveSurfer (2013), an open-source tool for sound

visualization and manipulation, which supports TIMIT word-level transcription format. Although, it supports Unicode, it does provide a proper support for Kazakh symbols. Therefore, we used an ASCII version of the Kazakh letters depicted on Figure 3. Also, we used the # symbol for the pauses and silence, and ^ symbol for other non-speech events.

6 Conclusion and Future Work

In this work we have described the design and compilation process of the Kazakh Language Corpus. KLC is oriented for a wide range of users and we believe that it will be a valuable tool for research communities, especially given that a portion of the data has been labeled with multiple levels of annotation, including word-level segmentation of audio information. We are already using the annotated data in our initial experiments in morpheme segmentation and error correction.

One can explore the corpus through the website (`http://kazcorpus.kz`) that was designed to provide the best experience in the analysis of data.

For the future work we plan to use the corpus as a research tool to tackle the following problems: (i) automatic part of speech tagging, (ii) morphological disambiguation, (iii) statistical machine translation. For the latter we have already started collecting parallel text in Russian and English.

Acknowledgments

We would like to thank the Ministry of Education and Science of the Republic of Kazakhstan for supporting this work through a grant under the 055 research program.

We express our gratitude to Dr. A. Sharipbayev for his valuable advice on methodology of constructing the read-speech corpus.

We also would like to sincerely thank our validators and annotators: Bobek A., Asemgul R., Aidana Zh., Nazerke G., Nazym K., Ainur N., Sandughash A., Zhuldyzai S., Dinara O., Aigerim Zh. The annotation and validation work they have done helped a great deal in designing tagsets. This work would not be possible without their contribution.

References

Yesim Aksan, Mustafa Aksan, Ahmet Koltuksuz, Taner Sezer, Umit Mersinli, Umut Ufuk Demirhan, Hakan Yilmazer, Gulsum Atasoy, Seda Oz, Ipek Yildiz, and Ozlem Kurtoglu. 2012. Construction of the turkish national corpus (tnc). In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

L. Al-Sulaiti and E.S. Atwell. 2006. The design of a corpus of contemporary arabic. *International Journal of Corpus Linguistics*, 11:135–171.

Gulila Altenbek and WANG Xiao-long. 2010. Kazakh segmentation system of inflectional affixes. In *Joint Conference on Chinese Language Processing*, pages 183–190. CIPS-SIGHAN.

Juri Apresjan, Igor Boguslavsky, Boris Iomdin, Leonid Iomdin, Andrei Sannikov, and Victor Sizov. 2006. A syntactically and semantically tagged corpus of russian: State of the art and prospects. In *The fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Vt. Baisa and Vt. Suchomel. 2012. Large corpora for turkic languages and unsupervised morphological analysis. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 28–32, Istanbul, Turkey. European Language Resources Association (ELRA).

Lou Burnard, editor. 2007. *Reference Guide for the British National Corpus*. Research Technologies Service at Oxford University Computing Services, February.

Alexandra Canavan and George Zipperlen. 1996. Callhome japanese speech.

K.-j. Chen. 1996. Sinica corpus: Design methodology for balanced corpora. In B.S. Park and J.B. Kim, editors, *Proceeding of the 11th Pacific Asia Conference on Language, Information and Computation*, pages 167–176, Seoul. Kyung Hee University.

CLMCRK. 2009. The corpus of kazakh language. [visited 29/08/2012].

David Elworthy. 1995. Tagset design and inflected languages. In *In EACL SIGDAT workshop iFrom Texts to Tags: Issues in Multilingual Language Analysis*, pages 1–10.

Anna Feldman. 2008. Tagset design, inflected languages, and n-gram tagging. *Editors: Paul Robertson and John Adamson*, 3(1):151.

J.L. Fleiss et al. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Winthrop Nelson Francis and Henry Kučera. 1979. *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Lingustics.

John Garofalo, David Graff, Doug Paul, and David Pallett. 2007. Csr-i (wsj0) complete linguistic data consortium, philadelphia.

John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, and Victor Zue. 1993. Timit acoustic-phonetic continuous speech corpus.

JJ Godfrey and E Holliman. 1997. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia*.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*

- *Volume 1*, ACL '98, pages 483–490, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jirka Hana and Anna Feldman. 2010. A positional tagset for russian. *Proceedings of LREC-10. Malta*.

Nancy Ide and Catherine Macleod. 2001. The american national corpus: A standardized resource of american english. In Paul Rayson, Andrew Wilson, Tony McEnery, Andrew Hardie, , and Shereen Khoja, editors, *Proceedings of the Corpus Linguistics 2001 Conference*, pages 274–280. Lancaster University (UK).

N. Ide, P. Bonhomme, and L. Romary. 2000. Xces: An xml-based standard for linguistic corpora. In *Proceedings of the Second Annual Conference on Language Resources and Evaluation*, pages 825–830, Athens.

Nancy Ide. 1998. Corpus encoding standard: Sgml guidelines for encoding linguistic corpora. In *Proceedings of the First International Language Resources and Evaluation Conference*, pages 463–70. Citeseer.

H. Kim. 2006. Korean national corpus in the 21st century sejong project. language corpora:their compilation and application. In *Proceedings of the 13th NIJL International Symposium*, pages 49–54, Tokyo, March.

Geoffrey Leech, Roger Garside, and Michael Bryant. 1994. Claws4: the tagging of the british national corpus. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 622–628. Association for Computational Linguistics.

R. Gary Leonard and George Doddington. 1993. Tidigits.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330, June.

Akmaral Mukan. 2012. *A Learner's Dictionary of Kazakh Idioms*. Georgetown University Press.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a turkish treebank. In *Treebanks*, pages 261–277. Springer.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *Arxiv preprint ArXiv:1104.2086*.

John F. Pitrelli, Cynthia Fong, Suk H. Wong, Judith R. Spitz, and Hong C. Leung. 1995. Phonebook: A phonetically-rich isolated-word telephone-speech database. In *ICASSP*, volume 1, pages 101–104.

Ruscorpora. 2003. Russian national corpus. [visited 29/08/2012].

WaveSurfer. 2013. `http://www.speech.kth.se/wavesurfer/`. Accessed: 2013-03-30.

Wikipedia. 2012. Kazakh alphabet. [visited 29/08/2012].

Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora

Ramy Eskander, Nizar Habash, Owen Rambow
Center for Computational Learning Systems
Columbia University
{reskander,habash,rambow}@ccls.columbia.edu

Abstract

We present a method for automatically learning inflectional classes and associated lemmas from morphologically annotated corpora. The method consists of a core language-independent algorithm, which can be optimized for specific languages. The method is demonstrated on Egyptian Arabic and German, two morphologically rich languages. Our best method for Egyptian Arabic provides an error reduction of 55.6% over a simple baseline; our best method for German achieves a 66.7% error reduction.

1 Introduction

Morphological lexicons specify all inflected forms for each lexeme; in a language with rich morphology, such a resource can be important for natural language processing (NLP) tasks in order to limit data sparseness. For example, a morphological lexicon is an important component of a morphological tagger or of a part-of-speech (POS) tagger for languages with rich morphology.[1] Traditionally, a morphological lexicon has been created through painstaking lexicographic and morphological analysis of the language, drawing on unannotated corpora. Recently, new approaches have emerged. Fully or largely unsupervised approaches cannot link surface forms to morphosyntactic features and are thus not suited for building morphological lexicons. This problem is overcome by approaches that use explicit linguistic knowledge. In this paper, we investigate using existing morphologically annotated corpora. In a morphologically annotated corpus, the words in naturally occurring texts or transcribed speech are annotated for the correct morphological analysis (including, of course, core POS) in context. While there has been much work on computational morphology, to our knowledge this is the first paper to study the question of how to extract morphological lexicons from morphologically annotated corpora, and how to determine how much annotation is needed. In this paper, we assume a corpus with each word annotated with morphosyntactic features and with a lemma which tells us what lexeme the word form is part of. The task is to predict the correspondence between a word form and its lemma and morphological features.

This paper makes two contributions. First, we introduce an algorithm that learns unseen forms by analogy. This algorithm is language-independent. It incrementally merges complementary paradigm information about different lexemes into more abstract and more informative inflectional classes. Second, we explore how to model stems, and we propose a generalization of the Semitic root-and-template modeling. We use Egyptian Arabic (EGY), and German (GER) as our test languages. We test on corpus data, in order to simulate a standard real-world application. The baseline just uses the word forms seen in training and does not predict any unseen forms. Our language-independent algorithm improves the performance for both EGY and GER, with error reductions over the baseline of 44.4% for EGY and of 66.7% for GER. By adding language-specific modeling of the stem using templates, we obtain further error reductions for EGY (up to 55.6%) but not for GER.

Next, we review related work (Section 2) and introduce the key linguistic concepts we use (Section 3). We present our basic language-independent method in Section 4, and our language-specific modeling of stem variation in Section 5.

[1] Note that a full-form morphological lexicon is functionally equivalent to a morphological analyzer-generator, since one can be used to create the other.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1032–1043,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Approaches to Morphological Modeling Much work has been done in the area of computational morphology ranging from systems painstakingly designed by hand (Koskenniemi, 1983; Buckwalter, 2004; Habash and Rambow, 2006; Détrez and Ranta, 2012) to unsupervised methods that learn morphology models from unannotated data (Creutz and Lagus, 2007; Monson et al., 2008; Hammarström and Borin, 2011; Dreyer and Eisner, 2011). There is a large continuum between these two approaches. Closer to one end, we find work on minimally supervised methods for morphology learning that make use of available resources such as parallel data, dictionaries or some additional morphological annotations (Yarowsky and Wicentowski, 2000; Cucerzan and Yarowsky, 2002; Neuvel and Fulop, 2002; Snyder and Barzilay, 2008). Closer to the other end, we find work that focuses on defining morphological models with limited lexicons that are then extended using raw text (Clément et al., 2004; Forsberg et al., 2006). The work presented in this paper falls in the middle of this continuum: we are interested in learning complete morphological models using rich morphological annotations and, optionally, limited linguistic knowledge. We compare the value of different amounts of annotation and how they relate to additional linguistic knowledge.

Morphological Paradigms Many traditional and modern theories of inflectional morphology organize natural language morphology by paradigms (Stump, 2001; Walther, 2011; Camilleri, 2011). Within the continuum we discussed above, we find hierarchical representations of paradigm knowledge that have been used in manually constructed morphological models (Finkel and Stump, 2002; Habash et al., 2005). Furthermore, Détrez and Ranta (2012) introduce an implementation of Smart Paradigms – heuristically organized paradigms minimizing the number of forms needed to predict the full paradigm of a particular lexeme.

Both Forsberg et al. (2006) and Clément et al. (2004) describe methods for automatically populating a lexicon from raw data given a set of morphological inflectional classes in a language. Our work differs in that we use annotated data, but do not start with a complete set of inflectional classes; thus, our work is exactly complementary to this work.

The concept of a paradigm is also used in many published efforts on unsupervised learning of morphology, although not always in a way consistent with its use in linguistics. For instance, Snover et al. (2002) (and later on Can and Manandhar (2012)) define a paradigm as "a set of suffixes and the stems that attach to those suffixes and no others". This definition is quite limited since it is not modeling the notion of lexeme. Chan (2006) defines a simpler concept of paradigms in his *probabilistic paradigm* model, which has many limitations, such as not handling syncretism or irregular morphology, nor distinguishing inflection and derivation.

Dreyer and Eisner (2011) learn complete German verb paradigms from a small set of complete seed paradigms (50 or 100), which they choose randomly from all verbs in the language. They model stem changes using letter-based models, and use a large unannotated corpus in addition to the seed paradigms. Durrett and DeNero (2013) attack the same problem as Dreyer and Eisner (2011). Instead of using unannotated text, they model explicit rules for affixes and stem changes. The major difference between these two efforts and our work is that the problem is defined differently: we assume that the training and test data is defined by a corpus, not by complete paradigms. Our methods therefore are more sensitive to frequency effects of tokens. We believe that our way of stating the problem is more relevant to actual computational challenges for languages with limited morphological resources. We empirically compare our approach to that of Durrett and DeNero (2013) in Section 5.

None of the unsupervised approaches mentioned model inflectional classes, i.e., meta-paradigmatic representation that cluster the various paradigms of different lexemes into a set of general classes of paradigms. There is no explicit notion of morphosyntactic features. In this paper we target the learning and completion of inflectional classes from morphologically annotated data. Our approach does not sacrifice details of what paradigms should include: we handle syncretism and stem changes, and allow for the prediction of new word forms from morphosyntactic features and lemmas, unlike the largely unsupervised work. Our work also differs from most previous work in that we investigate how to model stem change explicitly. Whereas other approaches model stem syncretism through letter-

based models (Yarowsky and Wicentowski, 2000; Neuvel and Fulop, 2002; Dreyer and Eisner, 2011), we explore the use of abstract stems.

In our previous work on the EGY morphological analyzer CALIMA, we similarly used a lexicon of annotated morphological forms and extended it automatically using a simpler approach to paradigm completion (Habash et al., 2012).

3 Linguistic Terminology

In this section, we review key concepts from morphology, and introduce the terminology we will use in this paper.[2] We then introduce our own formalization of stems using vocalic templates.

Morphology is the study of word forms and their decomposition into elementary morphemes, which are the smallest meaning-bearing units of a language. There are two types of morphological processes: inflectional and derivational morphology. In inflectional morphology, a core meaning is retained and different word forms reflect different types of morphosyntactic features such as person, number, or tense. In derivational morphology, the core meaning of a word is changed, and perhaps even its part-of-speech (POS). In this paper, we restrict our interest to inflectional morphology. Furthermore, we take the written form of the word to be primary, and base all morphological analyses on the written form.

We will refer to the set of all word forms that are related through inflectional morphology alone as **lexeme**. We can refer to a lexeme with a **lemma**, which we take to be a language-specific and conventionalized choice of one of the inflected forms. For example, in English the verb is conventionally cited in the infinitive (often with *to*, which can be omitted), while in Arabic it is conventionally cited in perfective third person masculine singular. The lemma is sometimes referred to as a "citation form". A **paradigm** of a lexeme is a list of **cells**, where a cell is a combination of a complete set of morphosyntactic features (properties) and the corresponding inflected form of the lexeme. A paradigm is **complete** if there are cells for all possible morphosyntactic features (the list of possible morphosyntactic features is of course language-dependent).

We can divide the word forms into **affixes** (i.e., prefixes and suffixes) and the **stem**. There is no single correct way to do this for the words of a language. Each lexeme has its own paradigm. We can abstract from paradigms by grouping together paradigms which share the same affixes in corresponding cells, and where stems in corresponding cells differ in some restricted manner. We can define the **inflectional class** (IC) more formally as a set of **abstract cells**, where an abstract cell is a combination of a complete set of morphosyntactic features (properties) and an abstract representation of a stem (an **abstract stem**) along with fully specified affixes. The abstract stems (and thus the abstract cells) must have the property that, given a single instantiated word form of a lexeme along with its associated IC, we can derive the *complete* paradigm of the lexeme deterministically. In the first results we present, we simply assume that the stem is either shared entirely with other abstract cells, or it is entirely lexically instantiated. We explore language-specific approaches to defining an abstract stem in Section 5, where we also discuss relevant morphological facts of EGY and GER.

Prefixes, suffixes, and stems can be the same for different cells of a single paradigm or IC. This is called **syncretism**. We will refer to the cells which share a stem as a **stem syncretism zone** or "zone" for short.

4 Language-Independent Inflectional Class Construction Algorithm

In this section, we present our language-independent IC construction algorithm (LICA). LICA consists of a core algorithm for building ICs from seen data and models of soft-stem syncretism and affix prediction.

4.1 Problem Definition

Starting with a corpus of words annotated as triples of ⟨*prefix*+*stem*+*suffix*, *lemma*, *features*⟩, we want to create a lexicon of complete ICs, with each IC having an associated set of lemmas. The following is an input example specifying the inflected form of the 3rd person plural imperfective inflection for the EGY lemma **katab**[3] 'write': ⟨y+iktib+uwA, **katab**, I3UP⟩.

[2] We (roughly) base our terminology and our conceptualization on the inferential-realizational theory of Stump (2001).

[3] Arabic transliteration throughout the paper is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

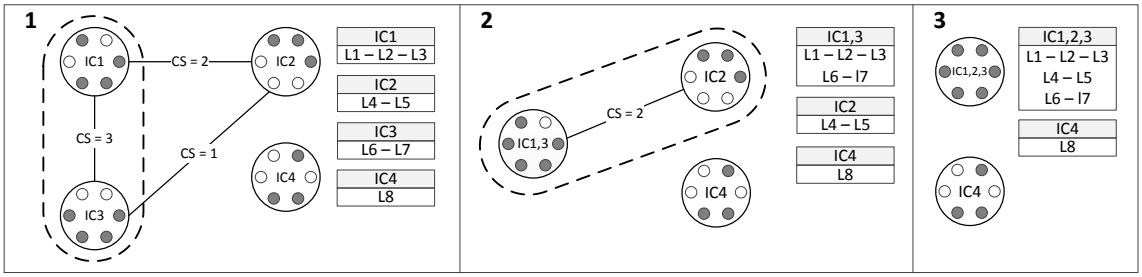


Figure 1: This graph illustrates two merges that result in combining three ICs in two steps. The IC cells are represented as solid (filled cells) or blank (empty cells) circles. CS is the compatibility score marking the number of matching filled cells in an IC. The boxes to the right of the graph represent the lexicon which associates lemmas (L*) with ICs. IC4 is not connected with any of the other ICs, because it has cell values that are incompatible with them.

4.2 Core Algorithm

Building the Initial Inflectional Classes An initial IC is constructed for each lemma found in the input corpus using all the triplets involving said lemma. Since the lemma itself is an inflected form with an *a priori* fixed feature combination, we also use the lemma to construct the initial IC, even if it did not occur as a word form in the corpus. If all inflected forms of a lemma appear in the training data, the IC will be complete. However, typically there are many unseen forms. If all seen forms related to one lexeme have the same stem, the abstract stem chosen for the IC is a single stem variable; the abstract cells of the new IC can of course differ in terms of affixes (which are always fully specified rather than represented as variables). In cases when the seen forms of one lemma have more than one stem, the IC created will simply be the same as the paradigm, i.e., we have fully instantiated stems as our abstract stems. We call such ICs **suppletive ICs**. At this point, we have a repository of numerous incomplete ICs and a lexicon consisting of a one-to-one mapping between lemmas and these new ICs. We then identify all ICs that are exactly the same (by definition, these are not suppletive ICs), and merge the associated sets of lemmas. We now have a new lexicon in which some ICs are associated with more than one lemma.

Constructing the Inflectional Class Graph We construct an IC graph that connects all *mergeable* ICs. Two ICs are mergeable if neither of the ICs is suppletive, if they share at least one abstract cell for some morphological feature, and if no morphological features are associated with different abstract cells, i.e., there are no incompatibilities. Each point in the graph represents a specific IC, while the edges carry the following four scores that we use to determine mergeability order:

- The **compatibility score** is the number of non-empty intersections between the two ICs.
- The **originality score** is the larger number of previous merges of the two ICs. At the beginning, this number is 0 for all edges.
- The **completeness score** is the size of the union of non-empty rows from the two ICs
- The **lexical size score** is the sum of the number of lemmas associated with the two ICs.

The edges of the graph are ranked according to the compatibility score first (more is better). Any ties are broken using the originality score (lower is better), any remaining ties by the completeness score (more is better) and then by the lexical size score (more is better). We explored all possible orders of tie breaking using EGY data, and determined the order of the listing above to be best. We use it in all experiments reported in this paper.

Merging the Inflectional Classes While the IC graph is still connected, we repeat the following merging procedure. Starting with the highest ranked edge in the graph, we merge the two ICs connected by the edge. In case of multiple edges that are equally highly ranked (after tie breaking), we select randomly among them. The merge creates a new IC that has the union of the cells in the two ICs. The lexicon is adjusted accordingly by associating with the new IC the union of all the lemmas originally associated with the two ICs. The new IC inherits the union of all the IC graph connections of its predecessors. The IC graph edge scores between the new

IC and other ICs are recalculated. Graph edges that become incompatible after the merge are removed. The two original ICs are removed from the graph and lexicon. See Figure 1 for an illustration of the merge process.

4.3 Completing Stems and Affixes

Soft Stem Syncretism Zones We extend the concept of stem-syncretism zones into a statistical model that computes the probability that the abstract stems in two abstract cells are the same given their feature combinations, i.e., that they belong to the same syncretism zone. We refer to this approach as "soft" stem-syncretism zones or "soft zones" for short. The probabilities are computed for each feature-combination pair as the ratio of the times the abstract stem for the feature-combination pair are equal divided by the times the abstract stem for the feature-combination pair are not empty. The probabilities can be computed after the initial IC graph construction or after the merging process has concluded; they can also be based on IC type counts or weighted by the number of associated lemmas. Applying soft zones to fully complete the ICs can only be done after merging is completed. We try all the possible alternatives for learning the SZs on the EGY data, including experiments with small training data sizes. The best accuracy is obtained using IC type weights learned after the merge. We use this setting for all experiments.

When applying the soft zones to determine the abstract stems of empty cells, we consider all filled cells in the same IC and select the abstract stem from the cell of the feature combination that has the highest soft-zone probability with the feature combination of the empty cell. Note that the copied abstract stem can be either a stem variable, or, for a suppletive IC, a lexical form.

Predicting Affixes In building the ICs mentioned above, some feature combinations and their corresponding affixes (prefixes and suffixes) are missing since they were not in the training data. We fill the missing affixes in a particular IC i by copying them from other ICs that we rank by overall seen affix similarity to IC i. ICs with conflicting affixes for any feature combinations are excluded completely. For any remaining missing affix-feature combination, we use the most common affix for that feature combination over all ICs.

4.4 Model Application

The complete ICs produced by the completion algorithm, associated with their lemmas in the lexicon, can now be used to predict the surface form of a given lemma and morphological features. We use the following procedure: If the given features are the same as those used to define the lemma, then the surface form is the same as the lemma form. If the lemma was seen in training, we select its IC from the model and generate the inflection associated with the features. This may require instantiating a stem variable (in case the IC is not suppletive), but this can be done deterministically from the lemma.

If the lemma has not been seen, then we build an IC on the fly using the lemma, i.e., an IC with a single cell. We then pick an IC from the model that does not conflict with that IC. The priority is given to the IC with the largest lexicon size, which is likely to be a result of several merges. If such an IC does not exist, a backup mode returns the stem of the lemma associated with the most frequent affixes of the queried features.

4.5 Results on Egyptian Arabic

Data and Metrics We use a morphologically annotated EGY corpus based on the CALLHOME EGY (CHE) corpus (Gadalla et al., 1997).[4] We divide the corpus into three parts: training, development and test, of about 75K, 36K and 41K words, respectively. We conduct our experiments on verbs only since they have a large number of possible morphosyntactic feature combinations. The verbs in these experiments are uncliticized. Clitics are easily handled using a few orthographic rules (El Kholy and Habash, 2010). On average, uncliticized verbs are about 12% of all words in our corpus. The data is represented in triplets as described in Section 4.1. There is a total of 19 feature combinations of aspect/mood (perfective,imperfective and imperative), person (first, second and third), gender (masculine, feminine and neutral) and number (singular and plural). Some combinations are invalid such as the first and third persons with the imperative form. The lemma we use is the Arabic citation form for verbs, which is the perfective third person masculine sin-

[4]The corpus was automatically annotated using information from the CHE transcripts (Gadalla et al., 1997) and the Egyptian Colloquial Arabic Lexicon (Kilany et al., 2002). For more details, see Habash et al. (2012) and Eskander et al. (2013).

gular (P3MS) inflection. We use this fact to fill the P3MS cells when building the initial ICs.

We evaluate the accuracy of the automatically generated bidirectional lexicon by generating surface forms from lemmas and morphosyntactic features. We use the model application method described in Section 4.4.

Baseline Our baseline system consists of two steps. First, we check the features. All cases of citation form features (in the case of EGY, P3MS) return the lemma as the inflected form. Otherwise, we look up the lemma and feature pair among all triplets in the training data and return the inflected form if such a triplet was found. If not, the baseline does not return an answer.

Results The results on tokens are summarized in Table 2 under the column heading BL (baseline) and NOTMP (LICA with no stem template). Our system consistently improves over the baseline for all training data sizes explored.

Error Analysis We performed an error analysis using the best settings found on the development set. We found that in 46% of the error types the lemma is unseen. Additional 35% of the cases are due to stem templates that are unseen in the complete ICs although their corresponding lemmas are seen. About one tenth of the cases are because of the existence of multiple forms of the same lemma and feature combinations, where our system assigns a form that is different from the gold form. Finally, gold errors contribute to about 9% of all errors.

4.6 Results on German

Data and Metrics For our experiments, we use the TIGER corpus (Brants and Hansen, 2002). We divide the corpus into three parts: training, development and test, of about 709K, 143K and 37K words, respectively. However, we use the first 75K words in training and the first 36K words in development to have the results comparable to EGY. We conduct our experiments on verbs only. We disregard any verbs with separable prefixes (as is common in work in morphology learning). On average, verbs without separable prefixes are about 9% of all words in our corpus. The data is represented in triplets as described in Section 4.1. There are 28 feature combinations of tense (present, past), person (first, second and third), number (singular, plural), and mood (indicative, subjunctive, imperative, past participle, infinitive). Some combinations are invalid such as any tense with the non-tensed participle or infinitive. The infinitive is the lemma. We use the same metrics as for EGY (Section 4.5).

Results The results are summarized in Table 5, with the relevant results in the column NOTMP. We see that our algorithm performs substantially better than the baseline at all training set sizes, with greater relative error reductions at smaller training sizes.

Error Analysis We inspected all error types in the development set and found that nearly 8% of all error types are errors in the gold annotation of the development set, and in 4% of cases, our predicted form is correct because a lemma is shared by two verbal paradigms (for example, *werden* has different past participles depending on whether it is the passive auxiliary or the verb for 'become').

5 Language-Specific Modeling of Stems

5.1 General Approach

We model the abstract stems using the notions of **orthographic template** and **orthographic root**. To meet our definition of IC given above, we define these two notions so that the root and template can always be extracted deterministically. We define them simply in terms of sets of letters: the set of letters of the alphabet used to write the language we are modeling is partitioned into the **root letters** and the **pattern letters**. The orthographic template is a string that specifies the template letters and that has placeholders for the root letters, which we write as '□'. The orthographic root is a sequence of strings that specifies the root letters that can fill a vocalic template's placeholders, in the order specified. Note that an IC along with a root is equivalent to a paradigm, since the root can be inserted deterministically into the abstract stem. This gives us another way to specify a lexeme: since a complete paradigm enumerates all inflected forms of a lexeme, and since a complete IC along with a root defines a complete paradigm, we can specify a lexeme to be a pair consisting of an IC along with a root. This pair determines a complete mapping from morphosyntactic features to surface word forms for the lexeme.

The basic algorithm discussed earlier is modified as follows: when we read in the training data, the

EGY Inflectional Class (IC) Repository			
	IC-1	IC-2	IC-3
P1US	□a□a□+t	□a□∼+ayt	□u□+t
P1UP	□a□a□+nA	□a□∼+aynA	□u□+nA
P2MS	□a□a□+t	□a□∼+ayt	□u□+t
P2FS	□a□a□+tiy	□a□∼+aytiy	□u□+tiy
P2UP	□a□a□+tuwA	□a□∼+aytuwA	□u□+tuwA
P3MS	□a□a□	□a□∼	□A□
P3FS	□a□a□+it	□a□∼+it	□A□+it
P3UP	□a□a□+uwA	□a□∼+uwA	□A□+uwA
I1US	Aa+□□i□	Aa+□i□∼	Aa+□uw□
I1UP	ni+□□i□	ni+□i□∼	ni+□uw□
I2MS	ti+□□i□	ti+□i□∼	ti+□uw□
I2FS	ti+□□i□+iy	ti+□i□∼+iy	ti+□uw□+iy
I2UP	ti+□□i□+uwA	ti+□i□∼+uwA	ti+□uw□+uwA
I3MS	yi+□□i□	yi+□i□∼	yi+□uw□
I3FS	ti+□□i□	ti+□i□∼	ti+□uw□
I3UP	yi+□□i□+uwA	yi+□i□∼+uwA	yi+□uw□+uwA
C2MS	A i□□i□	□i□∼	□uw□
C2FS	A i□□i□+iy	□i□∼+iy	□uw□+iy
C2UP	A i□□i□+uwA	□i□∼+uwA	□uw□+uwA

EGY Lexicon					
IC-1		IC-2		IC-3	
katab	**faraD**	**mad∼**	**Hal∼**	**šAf**	**qAl**
‖k,t,b‖	‖f,r,D‖	‖m,d‖	‖H,l‖	‖š,f‖	‖q,l‖
‘write’	‘suppose’	‘extend’	‘solve’	‘see’	‘say’

Table 1: Example of three inflectional classes and associated lemmas with their roots in EGY. The various blocks in the IC table specify different syncretism zones.

root of the lemma is used to decompose the stem of the inflected form into a root and a template.

5.2 Choosing Root and Pattern Letters

Determining which letters count as root letters and which count as pattern letters is language-specific. In this paper, we use three approaches to choosing root and pattern letters.

No Template (NOTMP) In this approach, we do not model the stem at all, i.e., there are no pattern letters. As a result, the ICs cannot express stem changes; any verb with a stem change will be an irregular verb and it will be modeled with a suppletive IC. Note that in EGY, *every* verb manifests a stem change between the perfective and imperfective forms, while in GER many verbs in fact do have the same stem for all inflected forms. The results of the NOTMP approach are the same basic approach results already presented in Section 4.

Scholar (SCHLR) In this approach, we use linguistic scholarship and intuition to choose a set of letters to be the pattern letters. These are letters which we know can change in stems within the same IC. We discuss this approach for EGY in Section 5.3.1 and for GER Section 5.4.1.

Empirical (EMPR) In this approach, we obtain the set of pattern letters empirically. We align the letters of all the stems in the development corpus with the stem letters of their corresponding lemmas. This allows us to identify which letters change between lemma stem and inflected stem. We order the letters by the probability that a letter changes given all occurrences of that letter, and by the probability that a letter changes given all changes that occur. This gives us two rankings. For both rankings and for each letter that changes, we construct a set of letters by including all letters from the most highly ranked letter down to the letter under consideration. This gives us per ranking as many sets as there are letters that change. We then choose the best performing set among all sets generated by both rankings. We present the results of this approach for EGY in Section 5.3.1 and for GER in Section 5.4.1.

5.3 Egyptian Arabic

5.3.1 Choice of Pattern Letters

Scholar-based Pattern Letters For the EGY SCHLR approach, we selected the following pattern letters: اأإآوؤيىئءـُـِـّ *AÂĂĀwŵyýŷ’aui∼*. These letters cover all the so-called weak root radicals, Hamzated forms and diacritics that are often used to discuss different verbal paradigms in Arabic (Gadalla, 2000). Table 1 shows three EGY inflectional classes. We follow the standard practice in modeling Arabic morphology of assuming that each placeholder in an orthographic template corresponds to exactly one letter from the orthographic root (i.e., the root is a sequence of strings each of length one). We would like to stress that our orthographic template and root differ from the notions of pattern and root in Semitic morphology (for an overview, see (Habash, 2010)): while for **katab** ‘write’ the orthographic root as well as the “real” root is ‖k,t,b‖, the orthographic root for **xal∼aý** ‘let’ is ‖x,l‖ (as opposed to the real root ‖x,l,y‖). Henceforth, we will omit the adjective “orthographic”, since we will not talk about the “real” root. To inflect a lemma for a particular set of features, we combine the root with the □ slots in the corresponding inflectional class feature row, e.g., Hal∼ + P2FS ⇒ ‖H,l‖ +

□a□∼+aytiy ⇒ Hal∼aytiy. Note that in this paper, we do not use any rules; all regular phonological and orthographic variation is "compiled" into the ICs. We also see examples of stem syncretism zones in Table 1; they are marked with horizontal lines. The stems associated with P3MS, P3FS and P3UP happen to be the same within each IC (although different across ICs). Different ICs have different zones: e.g., IC-1 has one zone for the P*** features, while IC-2 and IC-3 have two identical zones for P3** and P[12]**.

Empirically Determined Pattern Letters For EGY, the EMPR approach, using the algorithm described in Section 5.2, yields the following optimal set of pattern letters: اأإويىيئءثـّـُـِ *AÂĂwyýŷ'θaui*. This set is very similar to the SCHLR set except in that it omits the lexically constant (per lexeme) Shadda diacrtic ّ ∼ (orthographic gemination marker), and some very infrequent Hamzated forms; and it also adds one letter ث θ as a result of orthographic inconsistency in the training data.

Corpus Size	Verb Count	BL	NOTMP	SCHLR	EMPR	EMPR IIC+HZ
0K	0	19.3	20.0	20.0	20.0	58.8
1K	95	51.8	64.2	68.3	68.1	83.3
5K	630	64.2	77.0	83.9	83.9	91.1
10K	1,201	70.4	84.3	89.4	89.1	92.4
25K	2,994	79.6	91.5	94.4	94.8	95.6
50K	5,966	84.5	94.2	96.9	96.7	97.2
75K	8,690	85.6	94.9	97.5	97.6	97.2

Table 2: Learning curve comparing system performance for EGY on **tokens** (development set).

Results The template approaches SCHLR and EMPR consistently beat NOTMP which does not make use of any templatic stem modeling (see Table 2). However, SCHLR and EMPR perform about equally. This is not surprising since the two sets of pattern letters are quite similar.

Error Analysis We conducted an error analysis of EMPR using the best settings found on the development set. About 86% of the error types in NOTMP where the lemma is unseen are solved after introducing EMPR. Additionally, 71% of the cases in NOTMP where the stem template is unseen and the lemma is seen are solved. However, 7% of the error types in EMPR are not present in NOTMP, and they are all cases where the stem template is unseen while the lemma is seen. Errors due to multiple stem forms and gold errors remain the same in both NOTMP and EMPR, contributing to 26% and 23%, respectively, of all the error types in EMPR.

5.3.2 Other Enhancements with Linguistic Knowledge

Other linguistic knowledge can also help enrich the process of IC learning, especially under limited annotation conditions. We use the following two types of linguistic knowledge, which seamlessly integrate into the core merging algorithm described above.

Iconic Inflectional Classes (IICs) are ICs that are manually fully annotated, i.e., they have all the template cells for all morphosyntactic features specified. IICs are treated like any other ICs when constructing the initial IC graph. They are different from other ICs in that they initially have no lemmas associated with them in the lexicon.

Hard Stem-Syncretism Zones (HZ) are stem-syncretism zones determined manually by linguists to hold for all ICs. As such they can be more fine-grained than is needed to describe individual ICs. A hard zone is not applied in case of any partial disagreement within it. Unlike soft zones, they could be applied before or after the merge process, and they do not guarantee that the ICs will be completely filled.

We conducted experiments where we added external linguistic knowledge to our training data. We added 112 IICs which are extracted from all EGY verb inflections listed in a reference grammar of EGY (Gadalla, 2000). Also we added the following eight HZs for EGY (with reference to features $<tense, person, gender, number>$): (P1US-P1UP-P2MS-P2FS-P2UP), (I1UP-I2MS-I3MS-I3FS), (I2FS-I2UP-I3UP), (P3FS-P3UP), (C2FS-C2UP), (P3MS), (I1US), and (C2MS). Applying the HZs on the completed ICs (before soft zone application) gives higher results than applying them on initial ICs. We only report below on the setting of applying HZs after IC merge completion. We present the accuracies of IC learning for the baseline, and for the best setup with and without IICs and HZs, for different training sizes in Table 2. The baseline with no training data is at 19.3%

because of all the cases with verbs appearing in the citation form. As expected, IICs always help improve accuracy, especially under limited (and no) data conditions. However the benefits diminish rapidly for larger training sets. When evaluating on types only (results not presented in this paper), we find that using IICs in our system with no data is better than using the baseline with 75K words.

5.3.3 Blind Test Set

Table 3 shows the accuracies the different systems on our blind test set. The results for the test set are lower than those of the development set, but the trends are the same. We also compare our results to those obtained using the system of Durrett and DeNero (2013) on the same test data. Note that we apply their system to our problem – predicting unseen forms from annotated corpora (i.e., incomplete paradigms), not to the problem for which they created their system – predicting unseen forms from complete paradigms. Our best system outperforms theirs by 2.8% absolute in accuracy.

System	Accuracy	Error Reduction
Baseline	84.7	
NOTMP	91.5	44.4
SCHLR	93.2	55.6
EMPR	93.2	55.6
Durrett & DeNero	90.4	37.3

Table 3: Results for EGY on **tokens** using a blind test set.

5.4 German

5.4.1 Choice of Pattern Letters

Scholar-based Pattern Letters We now discuss our scholarship-based choice of pattern letters for German. Like Arabic, German verb paradigms can show stem changes which are typically vowel changes. Furthermore, like Arabic, German has prefixes, suffixes, and circumfixes. However, unlike Arabic, German has many verbs (called "weak verbs") which are regular in the sense that they show no stem change at all. The irregular verbs, or "strong verbs", show many different patterns of stem changes. Another difference to Arabic is that the affixes are not the same for all verb paradigms. In particular, the weak verbs form several inflectional classes (which, of course, differ only in affixes). Finally, unlike Arabic, in the strong verbs the orthographic root does not necessarily consist

GER Inflectional Class (IC) Repository						
	IC-1		IC-2		IC-3	
PI1S		□o□ +e		□e□ +e		□e□ +e
PI2S		□o□ +st		□ie□ +st		□i□ +st
PI3S		□o□ +t		□ie□ +t		□i□ +t
PI1P		□o□ +en		□e□ +en		□e□ +en
PI2P		□o□ +t		□e□ +t		□e□ +t
PI3P		□o□ +en		□e□ +en		□e□ +en
PS1S		□o□ +e		□e□ +e		□e□ +e
PS2S		□o□ +est		□e□ +est		□e□ +est
XI1S		□o□ +te		□a□ +		□a□ +
XI2S		□o□ +est		□a□ +st		□a□ +st
XS1S		□o□ +te		□ä□ +e		□ä□ +e
XS2S		□o□ +test		□ä□ +est		□ä□ +est
PP	ge+	□o□ +t	ge+	□e□ +en		□e□ +en
INF		□o□ +en		□e□ +en		□e□ +en

GER Lexicon					
IC-1		**IC-2**		**IC-3**	
holen	**sohlen**	**sehen**	**lesen**	**vergeben**	**begeben**
‖h,l‖	‖s,hl‖	‖s,h‖	‖l,s‖	‖verg,b‖	‖beg,b‖
'fetch'	'sole'	'see'	'read'	'forgive'	'occur'

Table 4: Example of three inflectional classes (some cells omitted in the interest of space economy) and associated lemmas with their roots in German ("X" stands for past tense). The different blocks in the IC table specify different stem syncretism zones.

of sequences of single letters: the strong verbs have monosyllabic stems (plus perhaps derivational morphology), with the vowel in this stem potentially undergoing changes. However, the onset and coda of the stem syllable can be any consonant cluster allowed by German phonology. Thus, in German, we model roots as pairs of strings of any length (which represent the onset and coda of the stem syllable). Table 4 shows a weak IC and two strong ICs.

Since German strong verbs can have any stem vowel, we assume that all eight vowel letters of German (*aeiouäöü*) are pattern letters, and all other consonant letters are root letters. German weak verbs show no stem changes at all; if we tailored our templates to them, we would define all letters to be root letters, and there would be no pattern letters at all. This is in fact the experiment we reported on in Section 4.6, and whose results are shown as NOTMP in Table 5. However, since we do not know during training time whether a verb is weak or strong, all verbs will be modeled with an orthographic template, even though there is no stem change at all.

Empirically Determined Pattern Letters For GER, the EMPR approach, using the algorithm described in Section 5.2, yields *oaäß* as the optimal

set of pattern letters. The EMPR pattern letters differ from the SCHLR pattern letters by omitting five vowels, but including the *ß* variant of the *s*.

Results In table 5 we see that using all eight vowels as pattern letters (SCHLR column) in fact decreases performance at every training size (and relatively more at smaller training sizes). However, if we use the empirically obtained pattern letter set *oaäß*, we see that we perform better than SCHLR at almost all training sizes, and slightly better than NOTMP at larger training sizes.

Error Analysis A manual inspection of all development error types again revealed 8% development set annotation errors and 4% acceptable variations. To investigate why NOTMP outperforms SCHLR for German on the development set, we performed an oracle experiment: we assumed we knew for each seen verb in training whether it is a weak or a strong verb. If it is weak, we model it using NOTMP, and if it is strong, using SCHLR. We observe as expected that the performance on weak verbs is very similar to that obtained using NOTMP on all verbs. However, for the strong verbs, it is only when we have more than 75,000 words of training data that the oracle outperforms NOTMP. We assume that the reason is that the highly frequent verbs are strong, but occur frequently enough so that their IC can be learned directly from the training data. Using SCHLR simply adds noise for these very frequent verbs. It is only for the less frequent strong verbs that SCHLR can contribute, and then only when a large amount of training data is available.

5.4.2 Blind Test Set

Table 6 shows the accuracies the different systems on our blind test set. The results for the test set are lower than those of the development set, and NOTMP, SCHLR,and EMPR produce very similar accuracy results. We also compare our results to those obtained by running the system of Durrett and DeNero (2013) on the same training and test data. Our system outperforms Durrett and DeNero (2013)'s system reducing the error of by 5%.[5]

[5]We also tested our system on Durrett and DeNero (2013)'s problem definition and data, training on 200 GER paradigms, and testing on 200 unseen paradigms. This is the case of testing for unseen lemmas in our system. Our system gives an accuracy of 88.4% as opposed to 91.8% as reported by Durrett and DeNero (2013). Our system was not designed for this task.

Corpus Size	Verb Count	BL	NOTMP	SCHLR	EMPR
0K	0	13.7	25.2	25.2	25.2
1K	81	43.3	72.0	67.0	69.4
5K	461	64.8	83.5	83.0	83.1
10K	929	72.1	89.0	87.4	88.6
25K	2,362.0	79.5	93.6	93.0	92.4
50K	4,527	86.2	95.6	95.1	95.6
75K	6,728	88.9	96.8	96.2	97.0

Table 5: Learning curve comparing system performance for GER on **tokens** on DEV corpus. BL=Baseline

System	Accuracy	Error Reduction
Baseline	89.5	
NOTMP	96.5	66.7
SCHLR	96.5	66.7
EMPR	96.5	66.7
Durrett	96.3	64.8

Table 6: Results for GER on **tokens** using a blind test set.

6 Conclusion and Future Work

We presented a method for automatically learning inflectional classes and associated lemmas from morphologically annotated corpora. In the future, we plan to improve several aspects of our models, in particular, using more powerful language-independent template transformations to automatically optimize for stem and affix modeling. We plan to take the insights from this paper and apply them to new dialects and languages with limited resources. We are interested in extending our approach to languages with different morphological systems, e.g., agglutinative or reduplicative. We will explore ideas from unsupervised morphology learning to minimize the need for morphological annotations.

Acknowledgment

This paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0014. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

Sabine Brants and Silvia Hansen. 2002. Developments in the tiger annotation scheme and their realization in the corpus. In *In Proceedings of the Third Conference on Language Resources and Evaluation LREC-02. Las Palmas de Gran Canaria*, pages 1643–1649.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Maris Camilleri. 2011. Island morphology: Morphology's interactions in the study of stem patterns. *Linguistica*, 51:65–84. Internal and External Boundaries of Morphology.

Burcu Can and Suresh Manandhar. 2012. Probabilistic hierarchical clustering of morphological paradigms. *EACL 2012*, page 654.

Erwin Chan. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL*, pages 69–78.

Lionel Clément, Benoît Sagot, and Bernard Lang. 2004. Morphology based automatic acquisition of large-coverage lexica. In *LREC 04*, pages 1841–1844.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1).

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *Proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7.

Grégoire Détrez and Aarne Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. *EACL 2012*, page 645.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627.

Greg Durrett and John DeNero. 2013. Supervised Learning of Complete Morphological Paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Ahmed El Kholy and Nizar Habash. 2010. Techniques for Arabic morphological detokenization and orthographic denormalization. In *Proceedings of LREC-2010*, May.

Ramy Eskander, Nizar Habash, Ann Bies, Seth Kulick, and Mohamed Maamouri. 2013. Automatic correction and extension of morphological annotations. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 1–10, Sofia, Bulgaria, August. Association for Computational Linguistics.

Raphael Finkel and Gregory Stump. 2002. Generating Hebrew Verb Morphology by Default Inheritance Hierarchies. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 9–18.

Markus Forsberg, Harald Hammarström, and Aarne Ranta. 2006. Morphological lexicon extraction from raw text data. *Advances in Natural Language Processing*, pages 488–499.

Hassan Gadalla, Hanaa Kilany, Howaida Arram, Ashraf Yacoub, Alaa El-Habashi, Amr Shalaby, Krisjanis Karins, Everett Rowson, Robert MacIntyre, Paul Kingsbury, David Graff, and Cynthia McLemore. 1997. CALLHOME Egyptian Arabic Transcripts. In *Linguistic Data Consortium, Philadelphia*.

Hassan Gadalla. 2000. *Comparative Morphology of Standard and Egyptian Arabic*. LINCOM EUROPA.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688, Sydney, Australia.

Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological Analysis and Generation for Arabic Dialects. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 17–24, Ann Arbor, Michigan.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

H. Kilany, H. Gadalla, H. Arram, A. Yacoub, A. El-Habashi, and C. McLemore. 2002. Egyptian Colloquial Arabic Lexicon. LDC catalog number LDC99L22.

Kimmo Koskenniemi. 1983. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th In-*

ternational Joint Conference on Artificial Intelligence, pages 683–685.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor: Finding paradigms across morphology. *Advances in Multilingual and Multimodal Information Retrieval*, pages 900–907.

Sylvain Neuvel and Sean A Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 31–40. Association for Computational Linguistics.

Matthew G Snover, Gaja E Jarosz, and Michael R Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 11–20.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June.

Gregory T. Stump. 2001. *Inflectional Morphology. A Theory of Paradigm Structure*. Cambridge Studies in Linguistics. Cambridge University Press.

Géraldine Walther. 2011. Measuring morphological canonicity. *Linguistica*, 51:157–180. Internal and External Boundaries of Morphology.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216.

Joint Language and Translation Modeling with Recurrent Neural Networks

Michael Auli, Michel Galley, Chris Quirk, Geoffrey Zweig
Microsoft Research
Redmond, WA, USA
{michael.auli,mgalley,chrisq,gzweig}@microsoft.com

Abstract

We present a joint language and translation model based on a recurrent neural network which predicts target words based on an unbounded history of both source and target words. The weaker independence assumptions of this model result in a vastly larger search space compared to related feed-forward-based language or translation models. We tackle this issue with a new lattice rescoring algorithm and demonstrate its effectiveness empirically. Our joint model builds on a well known recurrent neural network language model (Mikolov, 2012) augmented by a layer of additional inputs from the source language. We show competitive accuracy compared to the traditional channel model features. Our best results improve the output of a system trained on WMT 2012 French-English data by up to 1.5 BLEU, and by 1.1 BLEU on average across several test sets.

1 Introduction

Recently, several feed-forward neural network-based language and translation models have achieved impressive accuracy improvements on statistical machine translation tasks (Allauzen et al., 2011; Le et al., 2012b; Schwenk et al., 2012). In this paper we focus on recurrent neural network architectures, which have recently advanced the state of the art in language modeling (Mikolov et al., 2010; Mikolov et al., 2011a; Mikolov, 2012), outperforming multi-layer feed-forward based networks in both perplexity and word error rate in speech recognition (Arisoy et al., 2012; Sundermeyer et al., 2013). The major attraction of recurrent architectures is their potential to capture long-span dependencies since predictions are based on an *unbounded history* of previous words. This is in contrast to feed-forward networks as well as conventional n-gram models, both of which are limited to fixed-length contexts. Building on the success of recurrent architectures, we base our joint language and translation model on an extension of the recurrent neural network language model (Mikolov and Zweig, 2012) that introduces a layer of additional inputs (§2).

Most previous work on neural networks for speech recognition or machine translation used a rescoring setup based on n-best lists (Arisoy et al., 2012; Mikolov, 2012) for evaluation, thereby side stepping the algorithmic and engineering challenges of direct decoder-integration.[1] Instead, we exploit *lattices*, which offer a much richer representation of the decoder output, since they compactly encode an exponential number of translation hypotheses in polynomial space. In contrast, n-best lists are typically very redundant, representing only a few combinations of top scoring arcs in the lattice. A major challenge in lattice rescoring with a recurrent neural network model is the effect of the unbounded history on search since the usual dynamic programming assumptions which are exploited for efficiency do not hold up anymore. We apply a novel algorithm to the task of rescoring with an unbounded language model and empirically demonstrate its effectiveness (§3).

The algorithm proves robust, leading to significant improvements with the recurrent neural network language model over a competitive n-gram baseline across several language pairs. We even observe consistent gains when pairing the model with a large n-gram model trained on up to 575 times more

[1] One notable exception is Le et al. (2012a) who rescore reordering lattices with a feed-forward network-based model.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1044–1054,
Seattle, Washington, USA, 18-21 October 2013.

data, demonstrating that the model provides complementary information (§4).

Our joint modeling approach is based on adding a *continuous space representation* of the foreign sentence as an additional input to the recurrent neural network language model. With this extension, the language model can measure the consistency between the source and target words in a context-sensitive way. The model effectively combines the functionality of both the traditional channel and language model features. We test the power of this new model by using it as the only source of traditional channel information. Overall, we find that the model achieves accuracy competitive with the older channel model features and that it can improve over the gains observed with the recurrent neural network language model (§5).

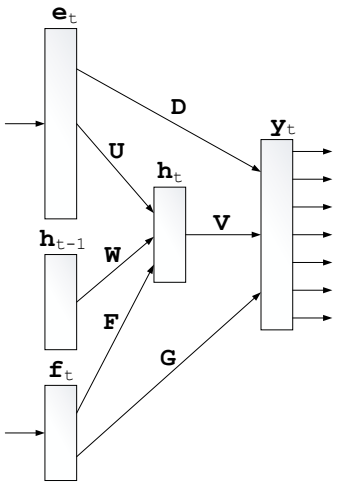


Figure 1: Structure of the recurrent neural network model, including the auxiliary input layer $\mathbf{f}_t$.

2 Model Structure

We base our model on the recurrent neural network language model of Mikolov et al. (2010) which is factored into an input layer, a hidden layer with recurrent connections, and an output layer (Figure 1). The input layer encodes the target language word at time t as a 1-of-N vector $\mathbf{e}_t$, where $|V|$ is the size of the vocabulary, and the output layer $\mathbf{y}_t$ represents a probability distribution over target words; both of size $|V|$. The hidden layer state $\mathbf{h}_t$ encodes the history of all words observed in the sequence up to time step t. This model is extended by an *auxiliary input layer* $\mathbf{f}_t$ which provides complementary information to the input layer (Mikolov and Zweig, 2012). While the auxiliary input layer can be used to feed in arbitrary additional information, we focus on encodings of the foreign sentence (§5).

The state of the hidden layer is determined by the input layer, the auxiliary input layer and the hidden layer configuration of the previous time step $\mathbf{h}_{t-1}$. The weights of the connections between the layers are summarized in a number of matrices: $\mathbf{U}$, $\mathbf{F}$ and $\mathbf{W}$, represent weights from the input layer to the hidden layer, from the auxiliary input layer to the hidden layer, and from the previous hidden layer to the current hidden layer, respectively. Matrix $\mathbf{V}$ represents connections between the current hidden layer and the output layer; $\mathbf{G}$ represents direct weights between the auxiliary input and output layers.

The hidden and output layers are computed via a series of matrix-vector products and non-linearities:

$$\mathbf{h}_t = s(\mathbf{U}\mathbf{e}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{F}\mathbf{f}_t)$$
$$\mathbf{y}_t = g(\mathbf{V}\mathbf{h}_t + \mathbf{G}\mathbf{f}_t)$$

where

$$s(z) = \frac{1}{1+\exp\{-z\}}, \quad g(z_m) = \frac{\exp\{z_m\}}{\sum_k \exp\{z_k\}}$$

are sigmoid and softmax functions, respectively. Additionally, the network is interpolated with a maximum entropy model of sparse n-gram features over input words (Mikolov et al., 2011a).[2] The maximum entropy weights are added to the output activations before computing the softmax.

The model is optimized via a maximum likelihood objective function using stochastic gradient descent. Training is based on the back propagation through time algorithm, which unrolls the network and then computes error gradients over multiple time steps (Rumelhart et al., 1986). After training, the output layer represents posteriors $p(e_{t+1}|e_{t-n+1}^t, \mathbf{h}_t, \mathbf{f}_t)$; the probabilities of words in the output vocabulary given the n previous input words e_{t-n+1}^t, the hidden layer configuration $\mathbf{h}_t$ as well as the auxiliary input layer configuration $\mathbf{f}_t$.

[2]While these features depend on multiple input words, we depicted them for simplicity as a connection between the current input word vector $\mathbf{e}_t$ and the output layer ($\mathbf{D}$).

Naïve computation of the probability distribution over the next word is very expensive for large vocabularies. A well established efficiency trick uses word-classing to create a more efficient two-step process (Goodman, 2001; Emami and Jelinek, 2005; Mikolov et al., 2011b) where each word is assigned a unique class. To compute the probability of a word, we first compute the probability of its class, and then multiply it by the probability of the word conditioned on the class:

$$p(e_{t+1}|e_{t-n+1}^{t}, \mathbf{h}_t, \mathbf{f}_t) = \\ p(c_i|e_{t-n+1}^{t}, \mathbf{h}_t, \mathbf{f}_t) \times p(e_{t+1}|c_i, e_{t-n+1}^{t}, \mathbf{h}_t, \mathbf{f}_t)$$

This factorization reduces the complexity of computing the output probabilities from $\mathcal{O}(|V|)$ to $\mathcal{O}(|C| + \max_i |c_i|)$ where $|C|$ is the number of classes and $|c_i|$ is the number of words in class c_i. The best case complexity $\mathcal{O}(\sqrt{|V|})$ requires the number of classes and words to be evenly balanced, i.e., each class contains exactly as many words as there are classes.

3 Lattice Rescoring with an Unbounded Language Model

We evaluate our joint language and translation model in a lattice rescoring setup, allowing us to search over a much larger space of translations than would be possible with n-best lists. While very space efficient, lattices also impose restrictions on the context available to features, a particularly challenging setting for our model which depends on the entire prefix of a translation. In the ensuing description we introduce a new algorithm to efficiently tackle this issue.

Phrase-based decoders operate by maintaining a set of *states* representing competing translations, either partial or complete. Each state is scored by a number of features including the n-gram language model. The independence assumptions of the features determine the amount of *context* each state needs to maintain in order for it to be possible to assign a score to it. For example, a trigram language model is indifferent to any context other than the two immediately preceding words. Assuming the trigram model dominates the Markov assumptions of all other features, which is typically the case, then we have to maintain at least two words at each state, also known as the *n-gram context*.

```
1: function RescoreLattice(k, V, E, s, T)
2:   Q ← topologically-sort(V)
3:   for all v in V do                    ▷ Heaps of split-states
4:     H_v ← minHeap()
5:   end for
6:   h_0 ← 0⃗                              ▷ Initialize start-state
7:   H_s.add(h_0)
8:   for all v in Q do                    ▷ Examine outgoing arcs
9:     for ⟨v, x⟩ in E do
10:      for h in H_v do                  ▷ Extend LM states
11:        h' ← scoreRNN(h, phrase(h))
12:        parent(h') ← h                 ▷ Backpointers
13:        if H_x.size() ≥ k∧             ▷ Beam width
14:          H_x.min()<score(h') then
15:            H_x.removeMin()
16:        if H_x.size()<k then
17:            H_x.add(h')
18:      end for
19:    end for
20:  end for
21:  I = maxHeap()
22:  for all t in T do                    ▷ Find best final split-state
23:    I.merge(H_t)
24:  end for
25:  return I.max()
26: end function
```

Figure 2: Push-forward rescoring with a recurrent neural network language model given a beam-width for language model split-states k, decoder states V, edges E, a start state s and final states T.

However, a recurrent neural network language model makes much weaker independence assumptions. In fact, the predictions of such a model depend on *all* previous words in the sentence, which would imply a potentially very large context. But storing all words is an inefficient solution from a dynamic programming point of view. Fortunately, we do not need to maintain entire translations as context in the states: the recurrent model compactly encodes the entire history of previous words in the hidden layer configuration $\mathbf{h}_i$. It is therefore sufficient to add $\mathbf{h}_i$ as context, instead of the entire translation. The language model can then simply score any new words

based on $\mathbf{h}_i$ from the previous state when a new state is created.

A much larger problem is that items, that were previously equivalent from a dynamic programming perspective, may now be different. Standard phrase-based decoders (Koehn et al., 2007) *recombine* decoder states with the same context into a single state because they are equivalent to the model features; usually recombination retains only the highest scoring candidate.[3] However, if the context is large, then the amount of recombination will decrease significantly, leading to less variety in the decoder beam. This was confirmed in preliminary experiments where we simulated context sizes of up to 100 words but found that accuracy dropped by between 0.5-1.0 BLEU.

Integrating a long-span language model naïvely requires to keep context equivalent to the entire left prefix of the translation, a setting which would permit very little recombination. Instead of using inefficient long-span contexts, we propose to maintain the usual n-gram context and to keep a fixed number of hidden layer configurations k at each decoder state. This leads to a new split-state dynamic program which splits each decoder state into at most k new items, each with a separate hidden layer configuration representing an unbounded history (Figure 2). This maintains diversity in the explored translation hypothesis space and preserves high-scoring hidden layer configurations.

What is the effect of this strategy? To answer this question we measured translation accuracy for various settings of k on our lattice rescoring setup (see §4 for details). In the same experiment, we compare lattices to n-best lists in terms of accuracy, model score and wall time impact.[4] The results (Table 1 and Figure 3) show that reranking accuracy on lattices is not significantly better, however, rescoring lattices with $k = 1$ is much faster than n-best lists. Similar observations have been made in previous work on minimum error-rate training (Macherey et al., 2008). The recurrent language model adds an overhead of about 54% at $k = 1$ on top of the time to produce the baseline 1-best output, a considerable but not necessarily prohibitive overhead. Larger values of k return higher probability solutions, but there is little impact on accuracy: the BLEU score is nearly identical when retaining up to 100 histories compared to keeping only the highest scoring.

While surprising at first, we believe that this effect is due to the high similarity of the translations represented by the histories in the beam. Each history represents a different translation but all translation hypothesis share the same n-gram context, and, more importantly, they are translations of the *same foreign words*, since they have exactly the same coverage vector. These commonalities are likely to result in similar recurrent histories, which in turn reduces the effect of aggressive pruning.

	BLEU	oracle	sec/sent
Baseline	28.25	-	0.173
100-best	28.90	37.22	0.470
1000-best	28.99	40.06	3.920
lattice ($k = 1$)	29.00	43.50	0.093
lattice ($k = 10$)	29.04	43.50	0.599
lattice ($k = 100$)	29.03	43.50	4.531

Table 1: Rescoring n-best lists and lattices with various language model beam widths k. Accuracy is based on the news2011 French-English task. Timing results are in *addition* to the baseline.

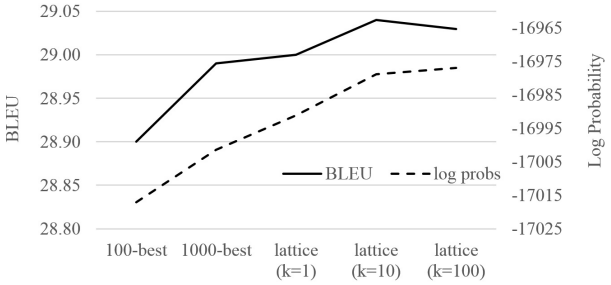


Figure 3: BLEU vs. log probabilities of 1-best translations when rescoring n-best lists and lattices (cf. Table 1).

4 Language Model Experiments

Recurrent neural network language models have previously only been used in n-best rescoring

[3] Assuming a max-translation decision rule. In a minimum-risk setting, we may assign the sum of the scores of all candidates to the retained item.

[4] We measured running times on an HP z800 workstation equipped with 24 GB main memory and two Xeon E5640 CPUs with four cores each, clocked at 2.66 GHz. All experiments were run single-threaded.

settings and on small-scale tasks with baseline language models trained on only 17.5m words (Mikolov, 2012). We extend this work by experimenting on lattices using strong baselines with n-gram models trained on over one billion words and by evaluating on a number of language pairs.

4.1 Experimental Setup

Baseline. We experiment with an in-house phrase-based system similar to Moses (Koehn et al., 2003), scoring translations by a set of common features including maximum likelihood estimates of source given target mappings $p_{MLE}(e|f)$ and vice versa $p_{MLE}(f|e)$, as well as lexical weighting estimates $p_{LW}(e|f)$ and $p_{LW}(f|e)$, word and phrase-penalties, a linear distortion feature and a lexicalized reordering feature. Log-linear weights are estimated with minimum error rate training (Och, 2003).

Evaluation. We use training and test data from the WMT 2012 campaign and report results on French-English, German-English and English-German. Translation models are estimated on 102m words of parallel data for French-English, 91m words for German-English and English-German; between 3.5-5m words are newswire, depending on the language pair, and the remainder are parliamentary proceedings. The baseline systems use two 5-gram modified Kneser-Ney language models; the first is estimated on the target-side of the parallel data, while the second is based on a large newswire corpus released as part of the WMT campaign. For French-English and German-English we use a language model based on 1.15bn words, and for English-German we train a model on 327m words. We evaluate on the newswire test sets from 2010-2011 containing between 2034-3003 sentences. Log-linear weights are estimated on the 2009 data set comprising 2525 sentences. We rescore the lattices produced by the baseline systems with an aggressive but effective context beam of $k = 1$ that did not harm accuracy in preliminary experiments (§3).

Neural Network Language Model. The vocabularies of the language models are comprised of the words in the training set after removing singletons. We obtain word-classes using a version of Brown-Clustering with an additional regularization term to optimize the runtime of the language model (Brown et al., 1992; Zweig and Makarychev, 2013). Direct connections use maximum entropy features over unigrams, bigrams and trigrams (Mikolov et al., 2011a). We use the standard settings for the model with the default learning rate $\alpha = 0.1$ that decays exponentially if the validation set entropy does not increase after each epoch. Back propagation through time computes error gradients over the past twenty time steps. Training is stopped after 20 epochs or when the validation entropy does not decrease over two epochs. We experiment with varying training data sizes and randomly draw the data from the same corpora used for the baseline systems. Throughout, we use a hidden layer size of 100 which provided a good trade-off between time and accuracy in initial experiments.

4.2 Results

Training times for neural networks can be a major bottleneck. Recurrent architectures are particularly hard to parallelize due to their inherent dependence on the previous hidden layer configuration. One straightforward way to influence training time is to change the size of the training corpus.

Our results (Table 2, Table 3 and Table 4) show that even small models trained on only two million words significantly improve over the 1-best decoder output (Baseline); this represents only 0.6 percent of the data available to the n-gram model used by the baseline. Models of this size can be trained in only about 3.5 hours. A model trained on 50m words took 63 hours to train. When paired with an n-gram model trained on 25 times more data, accuracy improved by up to 0.7 BLEU on French-English.

5 Joint Model Experiments

In the next set of experiments, we turn to the joint language and translation model, an extension of the recurrent neural network language model with additional inputs for the foreign sentence. We first introduce two continuous space representations of the foreign sentence (§5.1). Using these representations we evaluate the accuracy of the joint model in the lattice rescoring setup and compare against the traditional translation channel model features (§5.2). Next, we establish an upper bound on accuracy for the joint model via an oracle experiment (§5.3). Inspired by the results of the oracle experiment we

	dev	news2010	news2011	newssyscomb2011	Avg(test)
Baseline	26.6	27.6	28.3	27.5	27.8
+RNNLM (2m)	27.5	28.1	28.6	28.1	28.3
+RNNLM (50m)	27.7	28.2	29.0	28.1	28.5

Table 2: French-English results when rescoring with the recurrent neural network language model; the baseline relies on an n-gram model trained on 1.15bn words.

	dev	news2010	news2011	newssyscomb2011	Avg(test)
Baseline	21.2	20.7	19.2	20.6	20.0
+RNNLM (2m)	21.8	20.9	19.4	20.9	20.3
+RNNLM (50m)	22.1	21.1	19.7	21.0	20.5

Table 3: German-English results when rescoring with the recurrent neural network language model.

	dev	news2010	news2011	newssyscomb2011	Avg(test)
Baseline	15.2	15.6	14.3	15.7	15.1
+RNNLM (2m)	15.7	15.9	14.6	16.0	15.4
+RNNLM (50m)	15.8	15.9	14.7	16.1	15.5

Table 4: English-German results when rescoring with the recurrent neural network language model; the baseline relies on an n-gram model trained on 327m words.

train a transform between the source words and the reference representations. This leads to the best results improving 1.5 BLEU over the 1-best decoder output and adding 0.2 BLEU on average to the gains achieved by the recurrent language model (§5.4).

Setup. Conventional language models can be trained on monolingual or bilingual data; however, the joint model can only be trained on the latter. In order to control for data size effects, we restrict training of all models, including the baseline n-gram model, to the target side of the parallel corpus, about 102m words for French-English. Furthermore we train recurrent models only on the newswire portion (about 3.5m words for training and 250k words for validation) since initial experiments showed comparable results to using the full parallel corpus, available to the baseline. This is reasonable since the test data is newswire. Also, it allows for more rapid experimentation.

5.1 Foreign Sentence Representations

We represent foreign sentences either by latent semantic analysis (LSA; Deerwester et al. 1990) or by word encodings produced as a by-product of training the recurrent neural network language model on the source words.

LSA is widely used for representing words and documents in low-dimensional vector space. The method applies reduced singular value decomposition (SVD) to a matrix M of word counts; in our setting, rows represent sentences and columns represent foreign words. SVD reduces the number of columns while preserving similarity among the rows, effectively mapping from a high-dimensional representation of a sentence, as a set of words, to a low-dimensional set of *concepts*. The output of SVD is an approximation of M by three matrices: T contains single word representations, R represents full sentences, and S is a diagonal scaling matrix:

$$M \approx TSR^T$$

Given vocabulary V and n sentences, we construct M as a matrix of size $|V \times n|$. The ij-th entry is the number of times word i occurs in sentence j, also known as the term frequency value; the entry is also weighted by the inverse document frequency, the relative importance of word i among all sentences, expressed as the negative logarithm of the fraction of sentences in which word i occurs.

As a second representation we use single *word*

embeddings implicitly learned by the input layer weights $\mathbf{U}$ of the recurrent neural network language model (§2), denoted as **RNN**. Each word is represented by a vector of size $|\mathbf{h}_i|$, the number of neurons in the hidden layer; in our experiments, we consider concatenations of individual word vectors to represent foreign word contexts. These encodings have previously been found to capture syntactic and semantic regularities (Mikolov et al., 2013) and are readily available in our experimental framework via training a recurrent neural network language model on the source-side of the parallel corpus.

5.2 Results

We first experiment with the two previously introduced representations of the source-side sentence. Table 5 shows the results compared to the 1-best decoder output and an RNN language model (target-only). We first try LSA encodings of the entire foreign sentence as 80 or 240 dimensional vectors (sent-lsa-dim80, sent-lsa-dim240). Next, we experiment with single-word RNN representations of sliding word-windows in the hope of representing relevant context more precisely. Word-windows are constructed relative to the source words aligned to the current target word, and individual word vectors are concatenated into a single vector. We first try contexts which do not include the aligned source words, in the hope of capturing information not already modeled by the channel models, starting with the next five words (ww-rnn-dim50.n5), the five previous and the next five words (ww-rnn-dim50.p5n5) as well as the previous three words (ww-rnn-dim50.p3). Next, we experiment with word-windows of up to five aligned source words (ww-rnn-dim50.c5). Finally, we try contexts based on LSA word vectors (ww-lsa-dim50.n5, ww-lsa-dim50.p3).[5]

While all models improve over the baseline, none significantly outperforms the recurrent neural network language model in terms of BLEU. However, the perplexity results suggest that the models utilize the foreign representations since all joint models improve vastly over the target-only language model. The lowest perplexity is achieved by the context covering the aligned source words (ww-rnn-dim50.c5) since the source words are a better predictor of the target words than outside context.

[5]We ignore the coverage vector when determining word-windows which risks including already translated words. Building word-windows based on the coverage vector requires additional state in a rescoring setting meant to be light-weight.

	$-p(e\|f)$	$-p(f\|e)$ $-p(e\|f)$
Baseline without CM	24.0	22.5
+ target-only	24.5	22.6
+ sent-lsa-dim240	24.9	23.3
+ ww-rnn-dim50.n5	24.9	24.0
+ ww-rnn-dim50.p5n5	24.6	23.7
+ ww-rnn-dim50.p3	24.6	22.3
+ ww-rnn-dim50.c5	24.9	24.0
+ ww-lsa-dim50.n5	24.8	23.9
+ ww-lsa-dim50.p3	23.8	23.2

Table 6: Comparison of the joint model and the channel model features (CM) by removing channel features corresponding to $-p(e|f)$ from the lattices, or both directions $-p(e|f), -p(f|e)$ and replacing them by various joint models. We re-tuned the log-linear weights for different feature-sets. Accuracy is based on the average BLEU over news2010, newssyscomb2010, news2011.

The experiments so far measured if the joint model can improve *in addition* to the four channel model features used by the baseline, that is, the maximum likelihood and lexical translation features in both translation directions. The joint model clearly overlaps with these features, but how well does the recurrent model perform compared *against* the channel model features? To answer this question, we removed channel model features corresponding to the same translation direction as the joint model, specifically $p_{MLE}(e|f)$ and $p_{LW}(e|f)$, from the lattices and measured the effect of adding the joint models.

The results (Table 6, column $-p(e|f)$) clearly show that our joint models are competitive with the channel model features by outperforming the original baseline with all channel model features (24.7 BLEU) by 0.2 BLEU (ww-rnn-dim50.n5, ww-rnn-dim50.c5). As a second experiment, we removed all channel model features (column $-p(e|f), p(f|e)$), diminishing baseline accuracy to 22.5 BLEU. In this setting, the best joint model is able to make up 1.5 of the 2.2 BLEU lost due to removal of the channel

	dev	news2010	news2011	newssyscomb2010	Avg(test)	PPL
Baseline	24.3	24.4	25.1	24.3	24.7	341
target-only	25.1	25.1	26.4	25.0	25.6	218
sent-lsa-dim80	25.2	25.2	26.3	25.1	25.6	147
sent-lsa-dim240	25.1	25.0	26.2	24.9	25.4	126
ww-rnn-dim50.n5	24.9	25.0	26.3	24.8	25.4	61
ww-rnn-dim50.p5n5	25.0	24.8	26.2	24.7	25.3	59
ww-rnn-dim50.p3	25.1	25.1	26.5	24.9	25.6	143
ww-rnn-dim50.c5	24.8	24.9	26.0	24.8	25.3	16
ww-lsa-dim50.n5	25.0	25.0	26.2	24.8	25.4	76
ww-lsa-dim50.p3	25.1	25.1	26.5	24.9	25.6	151

Table 5: Translation accuracy of the joint model with various encodings of the foreign sentence measured on the French-English task. Perplexity (PPL) is based on news2011.

model features, while modeling only a single translation direction. This setup also shows the negligible effect of the target-only language model in the absence of translation scores, whereas the joint models are much more effective since they do model translation. Overall, the best joint models prove very competitive to the traditional channel features.

5.3 Oracle Experiment

The previous section examined the effect of a set of basic foreign sentence representations. Although we find some benefit from these representations, the differences are not large. One might naturally ask whether there is greater potential upside from this channel model. Therefore we turn to measuring the upper bound on accuracy for the joint approach as a whole.

Specifically, we would like to find a bound on accuracy given an *ideal representation* of the source sentence. To answer this question, we conducted an experiment where the joint model has access to an LSA representation of the reference translation.

Table 7 shows that the joint approach has an oracle accuracy of up to 4.3 BLEU above the baseline. This clearly confirms that the joint approach can exploit the additional information to improve BLEU, given a good enough representation of the foreign sentence. In terms of perplexity, we see an improvement of up to 65% over the target-only model. It should be noted that since LSA representations are computed on reference words, perplexity no longer has its standard meaning.

	BLEU	PPL
Baseline	25.2	341
target-only	26.4	218
oracle (sent-lsa-dim40)	27.7	124
oracle (sent-lsa-dim80)	28.5	103
oracle (sent-lsa-dim160)	29.0	86
oracle (sent-lsa-dim240)	29.5	76

Table 7: Oracle accuracy of the joint model when using an LSA encoding of the references, measured on the news2011 French-English task.

5.4 Target Language Projections

Our experiments so far showed that joint models based on direct representations of the source words are very competitive to the traditional channel models (§5.2). However, these experiments have not shown any improvements over the normal recurrent neural network language model. The previous section demonstrated that good representations can lead to substantial gains (§5.3). In order to bridge the gap, we propose to learn a separate transform from the foreign words to an encoding of the reference target words, thus making the source-side representations look more like the target-side encodings used in the oracle experiment.

Specifically, we learn a linear transform $d_\theta : \mathbf{x} \rightarrow \mathbf{r}$ mapping directly from a vector encoding of the foreign sentence $\mathbf{x}$ to an l-dimensional LSA representation $\mathbf{r}$ of the reference sentence. At test and training time we apply d_θ to the foreign words and use the transformation instead of a direct

	dev	news2010	news2011	newssyscomb2010	Avg(test)	PPL
Baseline	24.3	24.4	25.1	24.3	24.7	341
target-only	25.1	25.1	26.4	25.0	25.6	218
proj-lsa-dim40	25.1	25.3	26.5	25.2	25.8	145
proj-lsa-dim80	25.1	25.3	26.6	25.2	25.8	134

Table 8: Translation accuracy of the joint model with a source-target transform, measured on the French-English task. Perplexity (PPL) is based on news2011; differences to target-only are significant at the $p < 0.001$ level.

source-side representation.

The transform models all foreign words in the parallel corpus except singletons, which are collapsed into a unique class, similar to the recurrent neural network language model. We train the transform to minimize the *squared error* with respect to the reference LSA vector using an SGD online learner:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{n} \Big(\mathbf{r}_i - d_\theta(\mathbf{x}_i) \Big)^2 \qquad (1)$$

We found a simple constant learning rate, tuned on the validation data, to be as effective as schedules based on constant decay, or reducing the learning rate when the validation error increased. Our feature-set includes unigram and bigram word features. The value of unigram features is simply the unigram count in that sentence; bigram features receive a weight of the bigram count divided by two to help prevent overfitting. Then the vector for each sentence was divided by its L2 norm. Both weighting and normalization led to substantial improvements in test set error. More complex features such as skip-bigrams, trigrams and character n-grams did not yield any significant improvements. Even this representation of sentences is composed of a large number of instances, and so we resorted to feature hashing by computing feature ids as the least significant 20 bits of each feature name. Our best transform achieved a cosine similarity of 0.816 on the training data, 0.757 on the validation data, and 0.749 on news2011.

The results (Table 8) show that the transform improves over the recurrent neural network language model on all test sets and by 0.2 BLEU on average. We verified significance over the target-only model using paired bootstrap resampling (Koehn, 2004) over all test sets (7526 sentences) at the $p < 0.001$ level. Overall, we improve accuracy by up to 1.5 BLEU and by 1.1 BLEU on average across all test sets over the decoder 1-best with our joint language and translation model.

6 Related Work

Our approach of combining language and translation modeling is very much in line with recent work on n-gram-based translation models (Crego and Yvon, 2010), and more recently continuous space-based translation models (Le et al., 2012a; Gao et al., 2013). The joint model presented in this paper differs in a number of key aspects: we use a recurrent architecture representing an unbounded history of both source and target words, rather than a feed-forward style network. Feed-forward networks and n-gram models have a finite history which makes predictions independent of anything but a small history of words. Furthermore, we only model the target-side which is different to previous work modeling both sides.

We introduced a new algorithm to tackle lattice rescoring with an unbounded model. The automatic speech recognition community has previously addressed this issue by either approximating long-span language models via simpler but more tractable models (Deoras et al., 2011b), or by identifying confusable subsets of the lattice from which n-best lists are constructed and rescored (Deoras et al., 2011a). We extend their work by directly mapping a recurrent neural network model onto the structure of the lattice, rescoring all states instead of focusing only on subsets.

7 Conclusion and Future Work

Joint language and translation modeling with recurrent neural networks leads to substantial gains over the 1-best decoder output, raising accuracy by up to 1.5 BLEU and by 1.1 BLEU on average across

several test sets. The joint approach also improves over the gains of the recurrent neural network language model, adding 0.2 BLEU on average across several test sets. Our models are competitive to the traditional channel models, outperforming them in a head-to-head comparison.

Furthermore, we tackled the issue of lattice rescoring with an unbounded recurrent model by means of a novel algorithm that keeps a beam of recurrent histories. Finally, we have shown that the recurrent neural network language model can significantly improve over n-gram baselines across a range of language-pairs, even when the baselines were trained on 575 times more data.

In future work we plan to directly learn representations of the source-side during training of the joint model. Thus, the model itself can decide which encoding is best for the task. We also plan to change the cross entropy objective to a BLEU-inspired objective in a discriminative training regime, which we hope to be more effective. We would also like to apply recent advances in tackling the vanishing gradient problem (Pascanu et al., 2013) using a regularization term to maintain the magnitude of the gradients during back propagation through time. Finally, we would like to integrate the recurrent model directly into first-pass decoding, a straightforward extension of lattice rescoring using the algorithm we developed.

Acknowledgments

We would like to thank Anthony Aue, Hany Hassan Awadalla, Jon Clark, Li Deng, Sauleh Eetemadi, Jianfeng Gao, Qin Gao, Xiaodong He, Will Lewis, Arul Menezes, and Kristina Toutanova for helpful discussions related to this work as well as for comments on previous drafts. We would also like to thank the anonymous reviewers for their comments.

References

Alexandre Allauzen, Hélène Bonneau-Maynard, Hai-Son Le, Aurélien Max, Guillaume Wisniewski, François Yvon, Gilles Adda, Josep Maria Crego, Adrien Lardilleux, Thomas Lavergne, and Artem Sokolov. 2011. LIMSI @ WMT11. In *Proc. of WMT*, pages 309–315, Edinburgh, Scotland, July. Association for Computational Linguistics.

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep Neural Network Language Models. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 20–28, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Josep Crego and Franois Yvon. 2010. Factored bilingual n-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Anoop Deoras, Tomáš Mikolov, and Kenneth Church. 2011a. A Fast Re-scoring Strategy to Capture Long-Distance Dependencies. In *Proc. of EMNLP*, pages 1116–1127, Stroudsburg, PA, USA, July. Association for Computational Linguistics.

Anoop Deoras, Tomáš Mikolov, Stefan Kombrink, M. Karafiat, and Sanjeev Khudanpur. 2011b. Variational Approximation of Long-Span Language Models for LVCSR. In *Proc. of ICASSP*, pages 5532–5535.

Ahmad Emami and Frederick Jelinek. 2005. A Neural Syntactic Language Model. *Machine Learning*, 60(1-3):195–227, September.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2013. Learning Semantic Representations for the Phrase Translation Model. Technical Report MSR-TR-2013-88, Microsoft Research, September.

Joshua Goodman. 2001. Classes for Fast Maximum Entropy Training. In *Proc. of ICASSP*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of HLT-NAACL*, pages 127–133, Edmonton, Canada, May.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, Jun.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of EMNLP*, pages 388–395, Barcelona, Spain, Jul.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012a. Continuous Space Translation Models with

Neural Networks. In *Proc. of HLT-NAACL*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Hai-Son Le, Thomas Lavergne, Alexandre Allauzen, Marianna Apidianaki, Li Gong, Aurélien Max, Artem Sokolov, Guillaume Wisniewski, and François Yvon. 2012b. LIMSI @ WMT12. In *Proc. of WMT*, pages 330–337, Montréal, Canada, June. Association for Computational Linguistics.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *Proc. of EMNLP*, pages 725–734, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomáš Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proc. of Spoken Language Technologies (SLT)*, pages 234–239, Dec.

Tomáš Mikolov, Karafiát Martin, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proc. of INTERSPEECH*, pages 1045–1048.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. 2011a. Strategies for Training Large Scale Neural Network Language Models. In *Proc. of ASRU*, pages 196–201.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proc. of ICASSP*, pages 5528–5531.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space-Word Representations. In *Proc. of NAACL*, pages 746–751, Stroudsburg, PA, USA, June. Association for Computational Linguistics.

Tomáš Mikolov. 2012. *Statistical Language Models based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL*, pages 160–167, Sapporo, Japan, July.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training Recurrent Neural Networks. *Proc. of ICML*, abs/1211.5063.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning Internal Representations by Error Propagation. In *Symposium on Parallel and Distributed Processing*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics.

Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schlüter, and Hermann Ney. 2013. Comparison of Feedforward and Recurrent Neural Network Language Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8430–8434, Vancouver, Canada, May.

Geoff Zweig and Konstantin Makarychev. 2013. Speed Regularization and Optimality in Word Classing. In *Proc. of ICASSP*.

Multi-domain Adaptation for SMT Using Multi-task Learning*

Lei Cui[1], **Xilun Chen**[2], **Dongdong Zhang**[3], **Shujie Liu**[3], **Mu Li**[3], and **Ming Zhou**[3]

[1]Harbin Institute of Technology, Harbin, P.R. China
leicui@hit.edu.cn
[2]Cornell University, Ithaca, NY, U.S.
xlchen@cs.cornell.edu
[3]Microsoft Research Asia, Beijing, P.R. China
{dozhang,shujliu,muli,mingzhou}@microsoft.com

Abstract

Domain adaptation for SMT usually adapts models to an individual specific domain. However, it often lacks some correlation among different domains where common knowledge could be shared to improve the overall translation quality. In this paper, we propose a novel multi-domain adaptation approach for SMT using Multi-Task Learning (MTL), with in-domain models tailored for each specific domain and a general-domain model shared by different domains. The parameters of these models are tuned jointly via MTL so that they can learn general knowledge more accurately and exploit domain knowledge better. Our experiments on a large-scale English-to-Chinese translation task validate that the MTL-based adaptation approach significantly and consistently improves the translation quality compared to a non-adapted baseline. Furthermore, it also outperforms the individual adaptation of each specific domain.

1 Introduction

Domain adaptation is an active topic in statistical machine learning and aims to alleviate the domain mismatch between training and testing data. Like many machine learning tasks, Statistical Machine Translation (SMT) assumes that the data distributions of training and testing domains are similar. However, this assumption does not hold for real world SMT systems since training data for SMT models may come from a variety of domains. The translation quality is often unsatisfactory when translating texts from a specific domain using a general model that is trained over a hotchpotch of bilingual corpora. Therefore, domain adaptation is crucial for SMT systems to achieve better performance.

Previous research on domain adaptation for SMT includes data selection and weighting (Eck et al., 2004; Lü et al., 2007; Foster et al., 2010; Moore and Lewis, 2010; Axelrod et al., 2011), mixture models (Foster and Kuhn, 2007; Koehn and Schroeder, 2007; Sennrich, 2012; Razmara et al., 2012), and semi-supervised transductive learning (Ueffing et al., 2007), etc. Most of these methods adapt SMT models to a specific domain according to testing data and have achieved good performance. It is natural that real world SMT systems should adapt the models to multiple domains because the input may be heterogeneous, so that the overall translation quality can be improved. Although we can easily apply these methods to multiple domains individually, it is difficult to use the common knowledge across different domains. To leverage the common knowledge, we need to devise a multi-domain adaptation approach that jointly adapts the SMT models.

Multi-domain adaptation has been proved quite effective in sentiment analysis (Dredze and Crammer, 2008) and web ranking (Chapelle et al., 2011), where the commonalities and differences across multiple domains are explicitly addressed by Multi-task Learning (MTL). MTL is an approach that learns one target problem with other related problems at the same time, using a shared feature representation. The key advantage of MTL is to enable implicit data sharing and regularization. Therefore, it often leads to a better model for each task. Analogously, we expect that the overall translation quality can be further improved by using an MTL-based

*This work was done while the first and second authors were visiting Microsoft Research Asia.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1055–1065,
Seattle, Washington, USA, 18-21 October 2013.

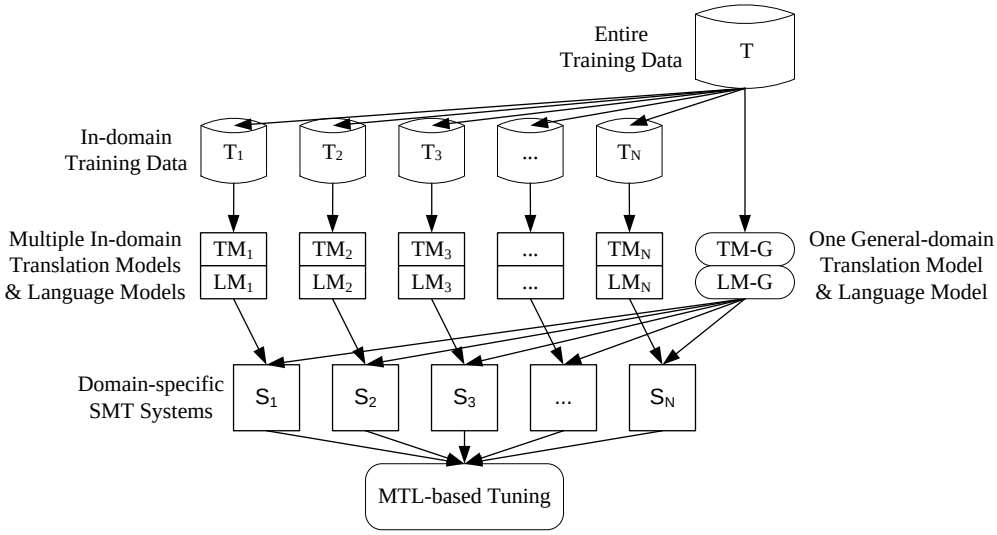


Figure 1: An example with N pre-defined domains, where T is the entire training corpus. T_i is the in-domain training data for the i-th domain selected from T using the bilingual cross-entropy based method (Axelrod et al., 2011). The in-domain TM$_i$ and LM$_i$ are trained using the in-domain training data T_i. The general-domain models TM-G and LM-G are trained using the entire training corpus T. S_i is the domain-specific SMT system for the i-th domain, leveraging the in-domain models and the general-domain models as features.

multi-domain adaptation approach.

In this paper, we use MTL to jointly adapt SMT models to multiple domains. Specifically, we develop multiple SMT systems based on mixture models, where each system is tailored for one specific domain with an in-domain Translation Model (TM) and an in-domain Language Model (LM). Meanwhile, all the systems share a same general-domain TM and LM. These SMT systems are considered as several related tasks with a shared feature representation, which fits well into a unified MTL framework. With the MTL-based joint tuning, general knowledge can be better learned by the general-domain models, while domain knowledge can be better exploited by the in-domain models as well. By using a distributed stochastic learning approach (Simianer et al., 2012), we can estimate the feature weights of multiple SMT systems at the same time. Furthermore, we modify the algorithm to treat in-domain and general-domain features separately, which brings regularization to multiple SMT systems in an efficient way. Experimental results have shown that our method can significantly improve the translation quality on multiple domains over a non-adapted baseline. Moreover, the MTL-based adaptation also outperforms the conventional individual adaptation approach towards each domain.

The rest of the paper is organized as follows: The proposed approach is explained in Section 2. Experimental results are presented in Section 3. Section 4 introduces some related work. Section 5 concludes the paper and suggests future research directions.

2 The Proposed Approach

Figure 1 gives an example with N pre-defined domains to illustrate the main idea. There are three steps in the training phase. First, in-domain training data is selected according to the pre-defined domains (Section 2.1). Second, in-domain models and general-domain models are trained to develop the domain-specific SMT systems (Section 2.2). Third, multiple domain-specific SMT systems are tuned jointly by using an MTL-based approach (Section 2.3).

2.1 In-domain Data Selection

In the first step, in-domain bilingual data is selected from all the bilingual data to train in-domain TMs. We use the bilingual cross-entropy based approach (Axelrod et al., 2011) to obtain the in-domain data:

$$[H_{I-src}(s)-H_{G-src}(s)]+[H_{I-tgt}(t)-H_{G-tgt}(t)] \quad (1)$$

where $\{s,t\}$ is a bilingual sentence pair in the entire bilingual corpus. $H_{I-\text{xxx}}(\cdot)$ and $H_{G-\text{xxx}}(\cdot)$ represent the cross-entropy of a string according to an in-domain LM and a general-domain LM, respectively. "xxx" denotes either the source language (src) or the target language (tgt). $H_{I-src}(s) - H_{G-src}(s)$ is the cross-entropy difference of string s between the in-domain and general-domain source-side LMs, and $H_{I-tgt}(t) - H_{G-tgt}(t)$ is the cross-entropy difference of string t between the in-domain and general-domain target-side LMs. This criterion biases towards sentence pairs that are like the in-domain corpus but unlike the general-domain corpus. Therefore, the sentence pairs with lower scores (larger differences) are presumed to be better.

Now, the question is how to find sufficient monolingual data to train in-domain LMs. A straightforward solution is to collect the data from the internet. There are a large number of monolingual webpages with domain information from web portal sites[1], which can be collected to train in-domain LMs. In large-scale real world SMT systems, practical domain adaptation techniques should target more domains rather than just one due to heterogeneous input. Therefore, we use a web crawler to collect monolingual webpages of N domains from web portal sites, for both the source language and the target language. The statistics of web-crawled data is given in Section 3.1. We use the web-crawled monolingual documents to train N in-domain source-side LMs and N in-domain target-side LMs. Additionally, we also train the source-side and target-side general-domain LMs with all the web-crawled documents from different domains. Finally, these in-domain and general-domain LMs are used to select in-domain bilingual data for different domains according to Formula (1).

2.2 SMT Systems with Mixture Models

In the second step, with the selected in-domain training data, we develop SMT systems based on mixture models. In particular, we use the mixture model based approach proposed by Koehn and Schroeder (2007). Specifically, we have developed N SMT systems for N domains respectively, where each system is a typical log-linear model. For each system, the best translation candidate $\hat{f}$ is given by:

$$\hat{f} = \arg\max_{f} \{P(f|e)\} \tag{2}$$

where the translation probability $P(f|e)$ is given by:

$$\begin{aligned} P(f|e) &\propto \sum_{i} w_i \cdot \log \phi_i(f,e) \\ &= \underbrace{\sum_{j \in \mathbf{I}} w_j \cdot \log \phi_j(f,e)}_{\textbf{I}\text{n-domain}} + \underbrace{\sum_{k \in \mathbf{G}} w_k \cdot \log \phi_k(f,e)}_{\textbf{G}\text{eneral domain}} \end{aligned} \tag{3}$$

where $\phi_j(f,e)$ is the in-domain feature function and w_j is the corresponding feature weight. $\phi_k(f,e)$ is the general-domain feature function and w_k is the feature weight. The detailed feature description is as follows:

In-domain features

- An in-domain TM, including phrase translation probabilities and lexical weights for both directions (4 features)
- An in-domain target-side LM (1 feature)
- word count (1 feature)
- phrase count (1 feature)
- NULL penalty (1 feature)
- Number of hierarchical rules used (1 feature)

General-domain features

- A general-domain TM, including phrase translation probabilities and lexical weights for both directions (4 features)
- A general-domain target-side LM (1 feature)

The feature description indicates that each SMT system contains two TMs and two LMs. The in-domain TMs are trained using the selected bilingual training data according to Formula (1), and the general-domain TM is trained using the entire bilingual training data. For the LMs, we re-use the target-side in-domain LMs and general-domain LM trained

[1]Many web portal sites contain domain information for webpages, such as "www.yahoo.com" in English and "www.sina.com.cn" in Chinese and etc. The webpages are often categorized by human editors into different domains, such as politics, sports, business, etc.

for data selection (Section 2.1). Compared with a normal single-model system, the system with mixture models can balance the contributions from the general-domain and in-domain knowledge. Hence it potentially benefits from both.

2.3 MTL-based Tuning

In the third step, the feature weights in multiple domain-specific SMT systems are estimated. Instead of tuning each domain-specific system separately, we treat different systems as related tasks and tune them jointly in an MTL framework. There are two main reasons for MTL-based tuning:

1. Domain-specific translation tasks share the same general-domain LM and TM. MTL often leads to better performance by leveraging commonalities among different tasks.

2. By enforcing that the general-domain LM and TM perform equally across different domains, MTL provides a kind of regularization to prevent over-fitting.

Formally, the objective function of the proposed MTL-based approach is described as follows:

$$\min_{\mathbf{W}} \left\{ \sum_{i=1}^{N} \textbf{Loss}(\mathbf{E}_i, \hat{\mathbf{e}}(\mathbf{F}_i, \mathbf{w}_i)) \right\} \quad (4)$$

where N is the number of pre-defined domains. $\{\mathbf{F}_i,\mathbf{E}_i\}$ is the in-domain development dataset for the i-th domain. $\mathbf{F}_i$ denotes the source sentences and $\mathbf{E}_i$ denotes the reference translations. $\mathbf{w}_i$ is a D-length feature weight column vector for the i-th domain, where D is the dimension of the feature space. $\mathbf{W}$ is a N-by-D matrix, representing $[\mathbf{w}_1|\mathbf{w}_2|\ldots|\mathbf{w}_N]^T$. $\hat{\mathbf{e}}(\mathbf{F}_i, \mathbf{w}_i)$ are the best translations obtained for $\mathbf{F}_i$ with parameters $\mathbf{w}_i$. $\textbf{Loss}(\cdot,\cdot)$ denotes the loss between the system's output and the reference translations. The basic idea of the objective function is to minimize the sum of loss functions for all the domains, rather than one domain at a time. Therefore, by adjusting the in-domain and general-domain feature weights, the translation quality is expected to be good across different domains.

To effectively tune SMT systems jointly, we modify the asynchronous Stochastic Gradient Descend (SGD) Algorithm (Simianer et al., 2012) to optimize objective function (4). We follow the pairwise ranking approach with the perceptron algorithm (Shen and Joshi, 2005) to update feature weights. Let a translation candidate be denoted by its feature vector $\mathbf{v} \in \mathbb{R}^D$, the pairwise preference for training is constructed by ranking two candidates according to the smoothed sentence-level BLEU (Liang et al., 2006). For a preference pair $\mathbf{v}_{[j]}$=($\mathbf{v}^{(1)}, \mathbf{v}^{(2)}$) where $\mathbf{v}^{(1)}$ is preferred, a hinge loss is used:

$$L(\mathbf{w}_i) = (-\langle \mathbf{w}_i, \mathbf{v}^{(1)} - \mathbf{v}^{(2)} \rangle)_+ \quad (5)$$

where $(x)_+ = \max(0, x)$ and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. With the perceptron algorithm (Shen and Joshi, 2005), the gradient of the hinge loss is:

$$\nabla L(\mathbf{w}_i) = \begin{cases} \mathbf{v}^{(2)} - \mathbf{v}^{(1)} & \text{if} \langle \mathbf{w}_i, \mathbf{v}^{(1)} - \mathbf{v}^{(2)} \rangle \leq 0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The training instances for the discriminative learning in pairwise ranking are made by comparing the N-best list of the translation candidates scored by the smoothed sentence-level BLEU (Liang et al., 2006). Following Simianer et al. (2012), the N-best list is divided into three bins: the top 10% (High), the middle 80% (Middle), and the last 10% (Low). These bins are used for pairwise ranking where the translation preference pairs are built between the candidates in High-Middle, Middle-Low, and High-Low, but not the candidates within the same bin, which is shown in Figure 2. The idea is to guarantee that the ranker is more discriminative to prefer the good translations to the bad ones.

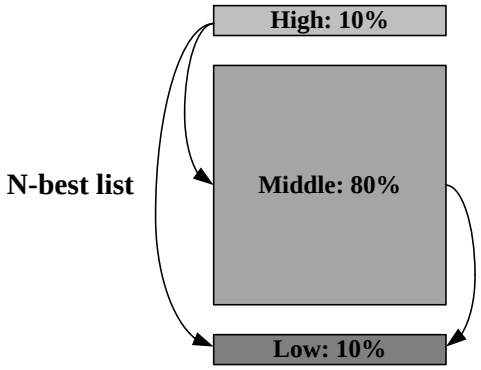


Figure 2: Training instances for pairwise ranking.

Algorithm 1 Modified Asynchronous SGD

1: Distribute N domain-specific decoders to N machines
2: Initialize $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N \leftarrow 0$
3: **for epochs** $t \leftarrow 0 \ldots T-1$ **do**
4: **for all domains** $d \in \{1 \ldots N\}$: **parallel do**
5: $\mathbf{u}_{d,t,0,0} = \mathbf{w}_d$
6: $S = |\mathbf{F}_d|$
7: **for all** $i \in \{0 \ldots S-1\}$ **do**
8: Decode i-th sentence with $\mathbf{u}_{d,t,i,0}$
9: P = No. of pairs built from the N-best list
10: **for all pairs** $\mathbf{v}_{[j]}, j \in \{0 \ldots P-1\}$ **do**
11: $\mathbf{u}_{d,t,i,j+1} \leftarrow \mathbf{u}_{d,t,i,j} - \eta \nabla L(\mathbf{u}_{d,t,i,j})$
12: **end for**
13: $\mathbf{u}_{d,t,i+1,0} \leftarrow \mathbf{u}_{d,t,i,P}$
14: **end for**
15: **end for**
16: **for all domains** $d \in \{1 \ldots N\}$ **do**
17: $\mathbf{w}_d = \mathbf{u}_{d,t,S,0}$
18: **end for**
19: $\mathbf{W}^G \leftarrow [\mathbf{w}_1^G | \ldots | \mathbf{w}_N^G]^T$
20: **for all domains** $d \in \{1 \ldots N\}$ **do**
21: **for** $k \leftarrow 1 \ldots |\mathbf{w}_d^G|$ **do**
22: $\mathbf{w}_d^G[k] = \frac{1}{N} \sum_{n=1}^{N} \mathbf{W}^G[n][k]$
23: **end for**
24: $\mathbf{w}_d \leftarrow \begin{bmatrix} \mathbf{w}_d^I \\ \mathbf{w}_d^G \end{bmatrix}$
25: **end for**
26: **end for**
27: **return** $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N$

Our modified algorithm is illustrated in Algorithm 1. Each column vector $\mathbf{w}_i$ is further split into two parts $\mathbf{w}_i^I$ and $\mathbf{w}_i^G$, representing the **I**n-domain and **G**eneral-domain feature weights respectively. In Algorithm 1, we first distribute the domain-specific SMT decoders to different machines and initialize the feature weights (line 1-2). Typically, the SGD algorithm runs in several iterations (In this study, we set the number of epochs T to 20) (line 3). Multiple SMT decoders run in parallel and each decoder updates its feature weights individually using its in-domain development data (line 4-15). For each domain, the domain-specific decoder translates each in-domain development sentence and determines the N-best translations (line 4-8). The preference pairs are built and used to update the parameters by gradient descent with $\eta = 0.0001$ (line 9-13). Each domain-specific decoder translates its in-domain development data multiple times. After each iteration, feature weights from all decoders are collected (line 16-19). In contrast to the original algorithm (Simianer et al., 2012), we only average the general-domain feature weights $\mathbf{w}_1^G, \ldots, \mathbf{w}_N^G$, but do not average the in-domain feature weights (line 20-25). The reason is we hope to leverage the commonalities among these systems. Meanwhile, general knowledge is enforced to be conveyed equally across different domains. Finally, the algorithm returns all the domain-specific feature weights $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N$ that are used for testing (line 27).

After the joint MTL-based tuning, the feature weights tailored for domain-specific SMT systems are used to translate the testing data. We collect in-domain testing data for each domain to evaluate the domain-specific systems. Although this is not always the case in real applications where the testing domain is known, this study mainly focuses on the effectiveness of the MTL-based tuning approach.

3 Experiments

3.1 Data

We evaluated our MTL-based domain adaptation approach on a large-scale English-to-Chinese machine translation task. The training data consisted of two parts: monolingual data and bilingual data. The monolingual data was used to train the source-side and target-side LMs, both of which were used for data selection in Section 2.1. In addition, the target-side LMs were re-used in the SMT systems as features. As mentioned in Section 2.1, we built a web crawler to collect a large number of webpages from web portal sites in English and Chinese respectively. In the experiments, we mainly focused on six popular domains, namely Business, Entertainment, Health, Science & Technology, Sports, and Politics. For both English and Chinese webpages, the HTML tags were removed and the main content was extracted. The data statistics are shown in Table 1.

The bilingual data we used was mainly mined from the web using the method proposed by Jiang et al. (2009), with a post-processing step using our bilingual data cleaning method (Cui et al., 2013). Therefore, the data quality is pretty good. In addition, we also used the English-Chinese parallel corpus released by LDC[2]. In total, the bilingual data

[2]LDC2003E07, LDC2003E14, LDC2004E12, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26,

Domain	English		Chinese	
	Docs	Words	Docs	Words
Business	21M	10.4B	7.91M	2.73B
Ent.	18.3M	8.29B	4.16M	1.31B
Health	8.7M	4.73B	0.9M	0.42B
Sci&Tech	10.9M	5.33B	5.28M	1.6B
Sports	18.9M	9.58B	2.49M	0.59B
Politics	10.3M	5.56B	1.67M	0.39B

Table 1: Statistics of web-crawled monolingual data, in numbers of documents and words (main content). "M" refers to million and "B" refers to billion.

contained around 30 million sentence pairs, with 404M words in English and 329M words in Chinese. For each domain, we used the cross-entropy based method in Section 2.1 to rank the entire bilingual data, and the top 10% sentence pairs from the ranked bilingual data were selected as the in-domain data to train the in-domain TM. Moreover, we prepared 2,000 in-domain sentences for development and 1,000 in-domain sentences for testing in each domain. The details are shown in Table 2.

Domain	Train		Dev		Test	
	En	Ch	En	Ch	En	Ch
Business	30M	28M	36K	35K	19K	19K
Ent.	25M	22M	21K	18K	13K	12K
Health	23M	20M	33K	33K	21K	22K
Sci&Tech	28M	26M	46K	45K	27K	27K
Sports	19M	16M	18K	14K	10K	9K
Politics	28M	24M	19K	17K	13K	12K

Table 2: Statistics of in-domain training, development and testing data, in number of words.

3.2 Setup

An in-house hierarchical phrase-based SMT decoder was implemented for our experiments. The CKY decoding algorithm was used and cube pruning was performed with the same default parameter settings as in Chiang (2007). We used a 100-best list from the decoder for the pairwise ranking algorithm. Translation models were trained over the bilingual data that was automatically word-aligned using GIZA++ (Och and Ney, 2003) in both directions, and the diag-grow-final heuristic was used to refine the symmetric word alignment. The phrase tables were filtered to retain top-20 translation candidates for each source phrase for efficiency. An in-house language modeling toolkit was used to train the 4-gram language models with modified Kneser-Ney smoothing (Kneser and Ney, 1995) over the web-crawled data. The evaluation metric for the overall translation quality was case-insensitive BLEU4 (Papineni et al., 2002). A statistical significance test was performed using the bootstrap resampling method (Koehn, 2004).

LDC2006E34, LDC2006E85, LDC2006E92.

3.3 Baseline

We have two baselines. The first baseline is a non-adapted Hiero using our implementation. It contained the general-domain TM and LM, as well as other standard features. In addition, the fix-discount method (Foster et al., 2006) for phrase table smoothing was also used. The system was general-domain oriented and it was tuned by using MERT (Och, 2003) with a combination of six in-domain development datasets. The second baseline is Google Online Translation Service[3]. We obtained the English-to-Chinese translations of the testing data from Google Translation to have a more solid comparison.

Moreover, we also compared our method with the adapted systems towards each domain individually (Koehn and Schroeder, 2007). This is to demonstrate the superiority of our MTL-based tuning approach across different domains.

3.4 Results

The end-to-end translation performance is shown in Table 3. We found that the baseline has a similar performance to Google Translation, with certain domains performed even better (Business, Sci&Tech, Sports, Politics). This demonstrates that the translation quality of our baseline is state-of-the-art. Moreover, we can answer three questions according to the experimental results as follow:

First, is domain mismatch a significant problem for a real world SMT system? We used the same system only with general-domain TM and LM, but tuned towards each domain individually using in-domain dev data. Table 3 shows that the setting "[A] *G*-TM + *G*-LM" performs much better than

[3]http://translate.google.com

	Business	Ent.	Health	Sci&Tech	Sports	Politics
[N] Baseline (*G*-TM + *G*-LM)	27.19	17.87	25.79	25.34	25.53	23.01
Google Translation	26.01	18.44	27.71	25.07	24.08	22.97
[A] *G*-TM + *G*-LM	29.58	19.08	28.80	26.84	30.28	25.64
[A] *I*-TM + *I*-LM	28.20	17.25	27.20	25.41	30.12	22.97
[A] (*G*+*I*)-TM + *G*-LM	29.45	19.22	28.93	27.01	31.01	25.40
[A] (*G*+*I*)-TM + *I*-LM	29.60	19.43	28.94	27.05	**34.36**	25.98
[A] (*G*+*I*)-LM + *G*-TM	29.66	19.50	29.00	27.10	33.60	26.03
[A] (*G*+*I*)-LM + *I*-TM	28.50	17.66	27.58	25.99	30.44	23.30
[A] (*G*+*I*)-TM + (*G*+*I*)-LM	29.82	19.53	29.03	26.94	33.77	26.09
[A,MTL] (*G*+*I*)-TM + (*G*+*I*)-LM	**30.26**	**19.94**	**29.08**	**27.17**	34.11	**26.50**

Table 3: End-to-end experimental results (BLEU4%) with large-scale training data ($p < 0.05$). "[N]" means the system is non-adapted and tuned using MERT on general-domain dev data. "[A]" denotes that the system is adapted towards each domain individually using MERT on in-domain dev data. "[A,MTL]" indicates that the system was tuned using our MTL-based approach on in-domain dev data. "*I*-TM" and "*G*-TM" denote the in-domain and general-domain translation model. "*I*-LM" and "*G*-LM" denote the in-domain and general-domain language model. We also obtained translations of the testing data using Google Translation for comparison.

the non-adapted baseline across all domains with at least 1.2 BLEU points. In addition, the setting "[A] *G*-TM + *G*-LM" also outperforms Google Translation on all domains. Analogous to previous research, this confirms that the domain mismatch indeed exists and the parameter estimation using in-domain dev data is quite useful.

Second, does the mixture models based adaptation work for a variety of domains? We experimented with different settings with multiple TMs or LMs, or both. It is interesting to note that for large-scale SMT systems, using in-domain models alone is inferior to using the general models alone. The setting "[A] *G*-TM + *G*-LM" is better than the setting "[A] *I*-TM + *I*-LM" across different domains. The reason is the data for general models has already included the in-domain data and the data coverage is much larger, thus the probability estimation is more reliable and the translation quality is much better.

For the LM, the in-domain LM performs better than the general-domain LM because our monolingual data (Table 1) for each domain is already sufficient for training an in-domain LM with good performance. From Table 3, we observed that the setting "[A] (*G*+*I*)-TM + *I*-LM" outperforms "[A] (*G*+*I*)-TM + *G*-LM", with the "Sports" domain being the most significant. For the TM, the performance of the in-domain TM is inferior to the general-domain TM. The results show that the setting "[A] (*G*+*I*)-LM + *G*-TM" is significantly better than "[A] (*G*+*I*)-LM + *I*-TM". The main reason is the data coverage for in-domain TM is much smaller than the general model. When each system uses two TMs and two LMs, it consistently results in better performance, indicating that mixture models are crucial for domain adaptation in SMT.

Third, can MTL further improve the translation quality? We used the MTL-based approach to jointly tune multiple domain-specific systems, leveraging the commonalities among different but related tasks. From Table 3, the MTL-based approach significantly improve the translation quality over the non-adapted baseline, and also outperforms conventional mixture models based methods. In particular, the "Sports" domain benefits the most from the in-domain knowledge, which confirms that domain discrepancy should be addressed and may bring large improvements on certain domains.

3.5 Discussion

According to our experiments, only averaging over the out-of-domain feature weights returned robust and converged results. We do not have theoretically grounded guarantee. However, we observed that the BLEU score of our method on DEV data was slightly lower than that in the baseline system, which indicates the out-of-domain features are less over-fitting on the domain-specific DEV data since

SOURSE	A point begins with a **player** serving the ball. This means one **player** hits the ball towards the other **player**. (The **serve** must be played from behind the baseline and must **land** in the **service box**. Players get two attempts to make a good **serve**.)
REF	得分由一个**球员**发球开始，这是指一个**球员**向另一个**球员**击球。(**发球**时选手必须站在底线之外，球必须要**落在**对方的**发球区**内，每次**发球**允许有一次失误。)
[N] Baseline (*G*-TM + *G*-LM)	舞会始于**玩家**服务的一个点。这意味着**玩家**对其他**玩家**的击球。(该**服务**必须从背后打的基线和必须**降落在**服务框。球员两次试图成为一个好的**服务**。)
[A] (*G+I*)-TM + (*G+I*)-LM	一开始球的**球员**，这意味着一名**球员** 球打向其他**球员**。(必须从底线**发球**，必须在**发球区**的**区域**。球员只有两次尝试去做一个好的**发球**。)
[A,MTL](*G+I*)-TM + (*G+I*)-LM	第一球的**球员**，这意味着一名 **球员**对另一个**球员**击球。(必须在底线后面**发球**，并且必须**降落在**发球区。球员两次试图成为一个好的**发球**。)

Table 4: Examples illustrating some different translations, where the Chinese phrases are translated from the English phrases with the same symbols (e.g., underline, wavy-line, and box). The details are explained in Section 3.5.

we enforced them to play the same role across different domains. It seems that averaging the out-of-domain feature weights can be considered as a kind of regularization.

An example sentence from the Sports domain with translations from different methods is shown in Table 4. In this sentence, the baseline always translates "player" to "玩家" (game player), which should be "球员" (ball player). And, the baseline translates "serve" to "服务" (work for), which should be "发球" (put the ball into play). The phrase "service box" here means "发球区", which denotes the zone where the ball is to be served. However, the baseline incorrectly splits them into two words, then translates "service" to "服务" and "box" to "框". In contrast, the approaches with adapted models are able to translate these words very well.

Both our MTL-based approach and the conventional adaptation methods leverage the mixture models. A natural question is why our MTL-based approach performs better than the individual adaptation. To answer this question, we looked into the details of the tuning and decoding procedures in the MTL-based approach. We observed that the BLEU score on the development data for each system was lower than the score when conducting individual adaptation. Considering that the algorithm enforcing the general features play the same role across different domains, we suspect that MTL-based approach introduces a kind of regularization for each domain-specific system. The regularization prevents the general features from biasing towards certain domains to the extreme. This property is quite important for real world SMT systems. Usually, a sentence is composed of some domain-specific words and some general words, so it is often improper to translate every word in the sentence using the in-domain knowledge. For the example in Table 4, the individual adaptation method "[A] (*G+I*)-TM + (*G+I*)-LM" translates "land" to "区域" (zone) improperly, because "区域" appears more often in the Sports text than the general-domain text. This shows that the individual adaptation methods tend to overfit the in-domain development data. In contrast, the MTL-based approach "[A,MTL](*G+I*)-TM + (*G+I*)-LM" just translates "land" to "降落在" (fall on), which is more appropriate.

4 Related Work

4.1 Domain Adaptation

One direction of domain adaptation explored the data selection and weighting approach to improve the performance of SMT on specific domains. Eck

et al. (2004) first decoded the testing data with a general TM, and then used the translation results to train an adapted LM, which was in turn used to re-decode the testing data. Lü et al. (2007) tried to weight the training data according to the similarity with test data using information retrieval models, while Foster et al. (2010) trained a discriminative model to estimate a weight for each sentence in the training corpus. Other methods conducted data selection based on cross-entropy (Moore and Lewis, 2010), and Axelrod et al. (2011) further extended their cross-entropy based method to the selection of bilingual corpus in the hope that more relevant corpus to the target domain could yield smaller models with better performance. Other methods included using semi-supervised transductive learning techniques to exploit the monolingual in-domain data (Ueffing et al., 2007).

Adaptation methods also involved the utilization of mixture models. Foster and Kuhn (2007) explored a number of variants of utilizing multiple TMs and LMs by interpolation. Koehn and Schroeder (2007) used MERT to simultaneously tune two TMs or LMs. Sennrich (2012) investigated the TM perplexity minimization as a method to set model weights in mixture modeling. In addition, inspired by system combination approaches, Razmara et al. (2012) used the ensemble decoding method to mix multiple translation models, which outperformed a variety of strong baselines.

Generally, most previous methods merely conducted domain adaption for a single domain, rather than multiple domains at the same time. One could also simply build multiple SMT systems that were adapted to multiple domains, but they were often separated and not tuned together. So far, there has been little research into the multi-domain adaptation problem over mixture models for SMT systems, as proposed in this paper.

4.2 Multi-task Learning

In machine learning, MTL is an approach to learn one target problem with other related problems at the same time. This often leads to a better model for the main task because it allows the learner to use the commonality among the tasks. MTL is performed by learning tasks in parallel while using a shared representation. Therefore, what is learned for each task can help other tasks be learned better.

MTL was successfully applied in some Natural Language Processing (NLP) tasks. For example, Blitzer et al. (2006) extended the MTL approach (Ando and Zhang, 2005) to domain adaptation tasks in part-of-speech tagging. Collobert and Weston (2008) proposed using deep neural networks to train a set of tasks, including part-of-speech tagging, chunking, named entity recognition, and semantic roles labeling. They reported that jointly learning these tasks led to superior performance. MTL was also applied in sentiment analysis (Dredze and Crammer, 2008) and web ranking (Chapelle et al., 2011) to address the multi-domain learning and adaptation. In SMT, Duh et al. (2010) proposed using MTL for N-best re-ranking on sparse feature sets, where each N-best list corresponded to a distinct task. Simianer et al. (2012) proposed distributed stochastic learning with feature selection inspired by MTL. The distributed learning approach outperformed several other training methods including MIRA and SGD.

Inspired by these methods, we used MTL to tune multiple SMT systems at the same time, where each system was composed of in-domain and general-domain models. Through a shared feature representation, the commonalities among the SMT systems were better learned by the general models. In addition, domain-specific translation knowledge was also better characterized by the in-domain models.

5 Conclusion and Future Work

In this paper, we propose an MTL-based approach to address multi-domain adaptation for SMT. We first use the cross-entropy based data selection method to obtain in-domain bilingual data. After that, in-domain TMs and LMs are trained for each domain-specific SMT system. In addition, the general-domain TM and LM are also trained and shared across different systems. Finally, MTL is leveraged to tune multiple systems jointly. Experimental results have shown that our approach is quite promising for the multi-domain adaptation problem, and it brings significant improvement over both the non-adapted baselines and the conventional domain adaptation methods with mixture models.

We assume the domain information for testing

data is known beforehand in this study. However, this is not always the case for real world SMT systems. Therefore, to apply our approach in real applications, the domain information needs to be identified automatically. In the future, we will pre-define more popular domains and develop automatic domain classifiers. For those domains that are identified with high confidence, we use the domain-specific system to translate the texts. For other texts, we use the general system to translate them. Furthermore, since our approach is a general training method, we may also combine this approach with other domain adaptation methods to get more performance improvement.

Acknowledgments

We are especially grateful to Nan Yang, Yajuan Duan, Hong Sun and Danran Chen for the helpful discussions. We also thank the anonymous reviewers for their insightful comments.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. 2011. Boosted multi-task learning. *Machine learning*, 85(1-2):149–173.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Lei Cui, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Bilingual data cleaning for smt using graph-based random walk. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 340–345, Sofia, Bulgaria, August. Association for Computational Linguistics.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 689–697, Honolulu, Hawaii, October. Association for Computational Linguistics.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 375–383, Uppsala, Sweden, July. Association for Computational Linguistics.

Matthias Eck, Stephan Vogel, and Alex Waibel. 2004. Language model adaptation for statistical machine translation based on information retrieval. In *In Proc. of LREC*.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic, June. Association for Computational Linguistics.

George Foster, Roland Kuhn, and Howard Johnson. 2006. Phrasetable smoothing for statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 53–61, Sydney, Australia, July. Association for Computational Linguistics.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA, October. Association for Computational Linguistics.

Long Jiang, Shiquan Yang, Ming Zhou, Xiaohua Liu, and Qingsheng Zhu. 2009. Mining bilingual data from the web with adaptively learnt patterns. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 870–878, Suntec, Singapore, August. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July. Association for Computational Linguistics.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350, Prague, Czech Republic, June. Association for Computational Linguistics.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden, July. Association for Computational Linguistics.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Majid Razmara, George Foster, Baskaran Sankaran, and Anoop Sarkar. 2012. Mixing multiple translation models in statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 940–949, Jeju Island, Korea, July. Association for Computational Linguistics.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France, April. Association for Computational Linguistics.

Libin Shen and Aravind K Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning*, 60(1-3):73–96.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–21, Jeju Island, Korea, July. Association for Computational Linguistics.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Translation with Source Constituency and Dependency Trees

Fandong Meng[†§] **Jun Xie**[†] **Linfeng Song**[†§] **Yajuan Lü**[†] **Qun Liu**[‡†]
†Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
§University of Chinese Academy of Sciences
{mengfandong,xiejun,songlinfeng,lvyajuan}@ict.ac.cn
‡Centre for Next Generation Localisation
Faculty of Engineering and Computing, Dublin City University
qliu@computing.dcu.ie

Abstract

We present a novel translation model, which simultaneously exploits the constituency and dependency trees on the source side, to combine the advantages of two types of trees. We take head-dependents relations of dependency trees as backbone and incorporate phrasal nodes of constituency trees as the source side of our translation rules, and the target side as strings. Our rules hold the property of long distance reorderings and the compatibility with phrases. Large-scale experimental results show that our model achieves significantly improvements over the constituency-to-string (+2.45 BLEU on average) and dependency-to-string (+0.91 BLEU on average) models, which only employ single type of trees, and significantly outperforms the state-of-the-art hierarchical phrase-based model (+1.12 BLEU on average), on three Chinese-English NIST test sets.

1 Introduction

In recent years, syntax-based models have become a hot topic in statistical machine translation. According to the linguistic structures, these models can be broadly divided into two categories: constituency-based models (Yamada and Knight, 2001; Graehl and Knight, 2004; Liu et al., 2006; Huang et al., 2006), and dependency-based models (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Xiong et al., 2007; Shen et al., 2008; Xie et al., 2011). These two kinds of models have their own advantages, as they capture different linguistic phenomena. Constituency trees describe how words and sequences of words combine to form constituents, and constituency-based models show better compatibility with phrases. However, dependency trees describe the grammatical relation between words of the sentence, and represent long distance dependencies in a concise manner. Dependency-based models, such as dependency-to-string model (Xie et al., 2011), exhibit better capability of long distance reorderings.

In this paper, we propose to combine the advantages of source side constituency and dependency trees. Since the dependency tree is structurally simpler and directly represents long distance dependencies, we take dependency trees as the backbone and incorporate constituents to them. Our model employs rules that represent the source side as head-dependents relations which are incorporated with constituency phrasal nodes, and the target side as strings. A head-dependents relation (Xie et al., 2011) is composed of a head and all its dependents in dependency trees, and it encodes phrase pattern and sentence pattern (typically long distance reordering relations). With the advantages of head-dependents relations, the translation rules of our model hold the property of long distance reorderings and the compatibility with phrases.

Our new model (Section 2) extracts rules from word-aligned pairs of source trees (constituency and dependency) and target strings (Section 3), and translate source trees into target strings by employing a bottom-up chart-based algorithm (Section 4). Compared with the constituency-to-string (Liu et al., 2006) and dependency-to-string (Xie et al., 2011) models that only employ a single type of trees, our

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1066–1076,
Seattle, Washington, USA, 18-21 October 2013.

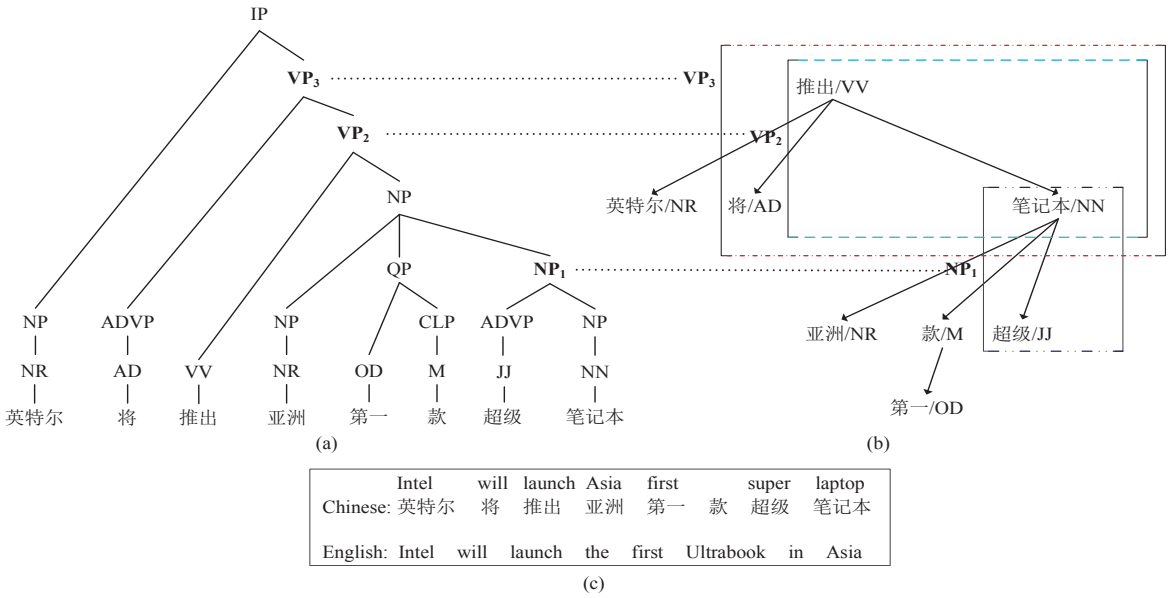


Figure 1: Illustration of phrases that can not be captured by a dependency tree (b) while captured by a constituency tree (a), where the bold phrasal nodes $\mathrm{NP}_1, \mathrm{VP}_2, \mathrm{VP}_3$ indicate the phrases which can not be captured by dependency syntactic phrases. (c) is the corresponding bilingual sentences. The subscripts of phrasal nodes are used for distinguishing the nodes with same phrasal categories.

approach yields encouraging results by exploiting two types of trees. Large-scale experiments (Section 5) on Chinese-English translation show that our model significantly outperforms the state-of-the-art single constituency-to-string model by averaged +2.45 BLEU points, dependency-to-string model by averaged +0.91 BLEU points, and hierarchical phrase-based model (Chiang, 2005) by averaged +1.12 BLEU points, on three Chinese-English NIST test sets.

2 Grammar

We take head-dependents relations of dependency trees as backbone and incorporate phrasal nodes of constituency trees as the source side of our translation rules, and the target side as strings. A head-dependents relation consists of a head and all its dependents in dependency trees, and it can represent long distance dependencies. Incorporating phrasal nodes of constituency trees into head-dependents relations further enhances the compatibility with phrases of our rules. Figure 1 shows an example of phrases which can not be captured by a dependency tree while captured by a constituency tree, such as the bold phrasal nodes NP_1,VP_2 and VP_3. The phrasal node NP_1 in the constituency tree indicates that "超级 笔记本" is a noun phrase and it should be translated as a basic unit, while in the dependency tree it is a non-syntactic phrase. The head-dependents relation in the top level of the dependency tree presents long distance dependencies of the words "英特尔", "将", "推出", and "笔记本" in a concise manner, which is useful for long distance reordering. We adopt this kind of rule representation to hold the property of long distance reorderings and the compatibility with phrases.

Figure 2 shows two examples of our translation rules corresponding to the top level of Figure 1-(b). We can see that r_1 captures a head-dependents relation, while r_2 extends r_1 by incorporating a phrasal node VP_2 to replace the two nodes "推出/VV" and "笔记本/NN". As shown in Figure 1-(b), VP_2 consists of two parts, a head node "推出/VV" and a subtree rooted at the dependent node "笔记本/NN". Therefore, we use VP_2 and the POS tags of the two nodes VV and NN to denote the part covered by VP_2 in r_2, to indicate that the source sequence covered by VP_2 can be translated by a bilingual phrase. Since VP_2 covers a head node "推出/VV", we represent r_2 by constructing a new head node

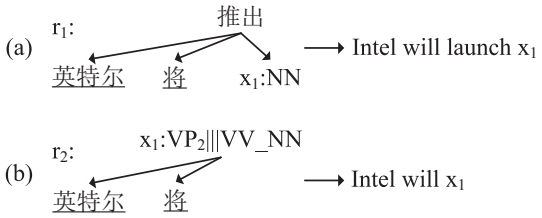


Figure 2: Two examples of our translation rules corresponding to the top level of Figure 1-(b). r_1 captures a head-dependents relation, and r_2 extends r_1 by incorporating a phrasal node VP_2. "x_1:NN" indicates a substitution site which can be replaced by a subtree whose root has POS tag "NN". "x_1:VP_2|||VV_NN" indicates a substitution site which can be replaced by a source phrase covered by a phrasal node VP (the phrasal node consists of two dependency nodes with POS tag VV and NN, respectively). The underline denotes a leaf node.

VP_2|||VV_NN. For simplicity, we use a shorten form CHDR to represent the head-dependents relations with/without constituency phrasal nodes.

Formally, our grammar G is defined as a 5-tuple $G = \langle \Sigma, N_c, N_d, \Delta, R \rangle$, where Σ is a set of source language terminals, N_c is a set of constituency phrasal categories, N_d is a set of categories (POS tags) for the terminals in Σ, Δ is a set of target language terminals, and R is a set of translation rules that include bilingual phrases for translating source language terminals and CHDR rules for translation and reordering. A CHDR rule is represented as a triple $\langle t, s, \sim \rangle$, where:

- t is CHDR with each node labeled by a terminal from Σ or a variable from a set $X = \{x_1, x_2, \cdots\}$ constrained by a terminal from Σ or a category from N_d or a joint category (constructed by the categories from N_c and N_d);
- $s \in (X \cup \Delta)$ denotes the target side string;
- $\sim$ denotes one-to-one links between nonterminals in t and variables in s.

We use the lexicon dependency grammar (Hellwig, 2006) which adopts a bracket representation to express the head-dependents relation and CHDR. For example, the left-hand sides of r_1 and r_2 in Figure 2 can be respectively represented as follows:

(英特尔) (将) 推出 (x_1:NN)

(英特尔) (将) x_1:VP_2|||VV_NN

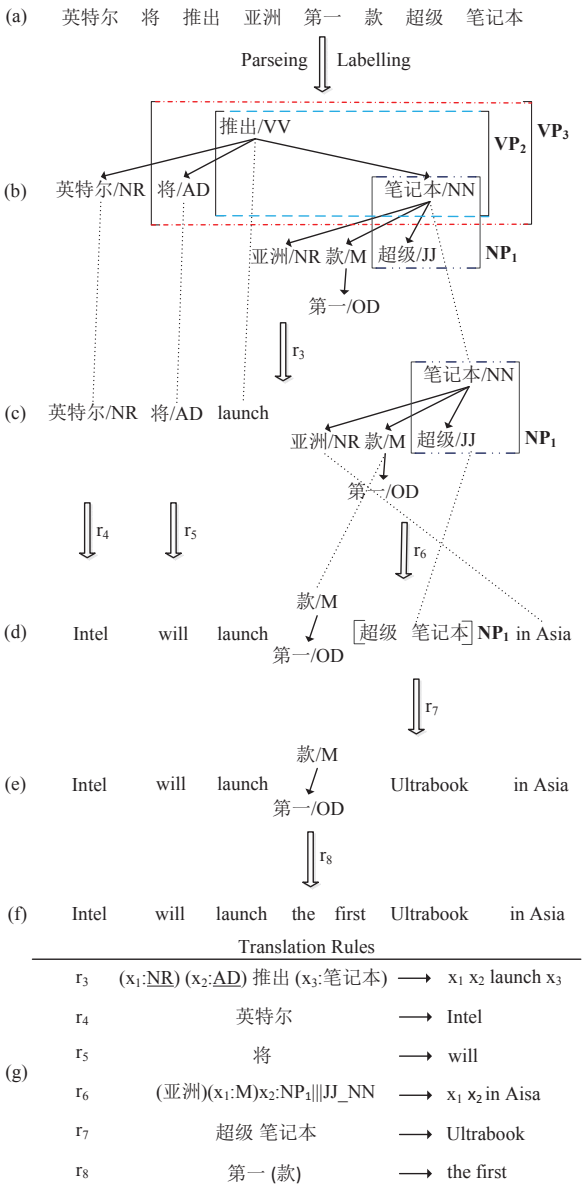


Figure 3: An example derivation of translation. (g) lists all the translation rules. r_3, r_6 and r_8 are CHDR rules, while r_4, r_5 and r_7 are bilingual phrases, which are used for translating source terminals. The dash lines indicate the reordering when employing a translation rule.

The formalized presentation of r_2 in Figure 2-(b):

t = (英特尔) (将) x_1:VP_2|||VV_NN

s = Intel will x_1

$\sim$= x_1:VP_2|||VV_NN $\leftrightarrow$ x_1

where the underline indicates a leaf node.

Figure 3 gives an example of the translation derivation in our model, with the translation rules

listed in (g). r_3, r_6 and r_8 are CHDR rules, while r_4, r_5 and r_7 are bilingual phrases, which are used for translating source language terminals. Given a sentence to translate in (a), we first parse it into a constituency tree and a dependency tree, then label the phrasal nodes from the constituency tree to the dependency tree, and yield (b). Then, we translate it into a target string by the following steps. At the root node, we apply rule r_3 to translate the top level head-dependents relation and results in four unfinished substructures and target strings in (c). From (c) to (d), there are three steps (one rule for one step). We use r_4 to translate "英特尔" to "Intel", r_5 to translate "将" to "will", and r_6 to translate the rightmost unfinished part. Then, we apply r_7 to translate the phrase "超级 笔记本" to "Ultrabook", and yield (e). Finally, we apply r_8 to translate the last fragment to "the first", and get the final result (f).

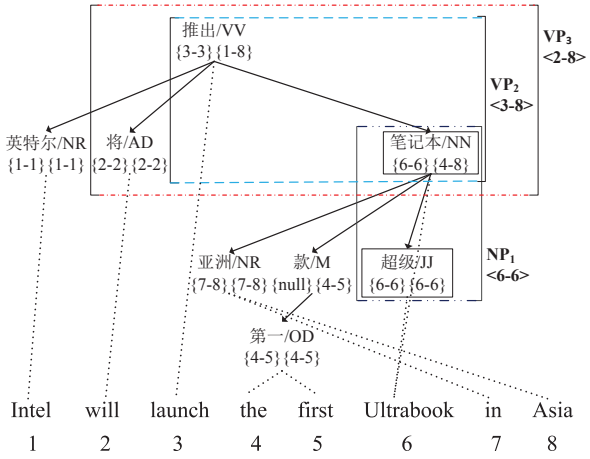


Figure 4: An annotated dependency tree. Each node is annotated with two spans, the former is node span and the latter subtree span. The fragments covered by phrasal nodes are annotated with phrasal spans. The nodes denoted by the solid line box are not nsp consistent.

3 Rule Extraction

In this section, we describe how to extract rules from a set of 4-tuples $\langle C, T, S, A\rangle$, where C is a source constituency tree, T is a source dependency tree, S is a target side sentence, and A is an word alignment relation between T/C and S. We extract CHDR rules from each 4-tuple $\langle C, T, S, A\rangle$ based on GHKM algorithm (Galley et al., 2004) with three steps:

1. Label the dependency tree with phrasal nodes from the constituency tree, and annotate alignment information to the phrasal nodes labeled dependency tree (Section 3.1).

2. Identify acceptable CHDR fragments from the annotated dependency tree for rule induction (Section 3.2).

3. Induce a set of lexicalized and generalized CHDR rules from the acceptable fragments (Section 3.3).

3.1 Annotation

Given a 4-tuple $\langle C, T, S, A\rangle$, we first label phrasal nodes from the constituency tree C to the dependency tree T, which can be easily accomplished by phrases mapping according to the common covered source sequences. As dependency trees can capture some phrasal information by dependency syntactic phrases, in order to complement the information that dependency trees can not capture, we only label the phrasal nodes that cover dependency non-syntactic phrases.

Then, we annotate alignment information to the phrasal nodes labeled dependency tree T, as shown in Figure 4. For description convenience, we make use of the notion of spans (Fox, 2002; Lin, 2004). Given a node n in the source phrasal nodes labeled T with word alignment information, the spans of n induced by the word alignment are consecutive sequences of words in the target sentence. As shown in Figure 4, we annotate each node n of phrasal nodes labeled T with two attributes: *node span* and *subtree span*; besides, we annotate *phrasal span* to the parts covered by phrasal nodes in each subtree rooted at n. The three types of spans are defined as follows:

Definition 1 *Given a node n, its **node span** $nsp(n)$ is the consecutive target word sequence aligned with the node n.*

Take the node "亚洲/NR" in Figure 4 for example, nsp(亚洲/NR)={7-8}, which corresponds to the target words "in" and "Asia".

Definition 2 *Given a subtree T' rooted at n, the **subtree span** $tsp(n)$ of n is the consecutive target word sequence from the lower bound of the nsp of*

all nodes in T' to the upper bound of the same set of spans.

For instance, tsp(笔记本/NN)={4-8}, which corresponds to the target words "the first Ultrabook in Asia", whose indexes are from 4 to 8.

Definition 3 *Given a fragment f covered by a phrasal node, the* ***phrasal span*** $psp(f)$ *of f is the consecutive target word sequence aligned with source string covered by f.*

For example, $psp(\text{VP}_2)$=⟨3-8⟩, which corresponds to the target word sequence "launch the first Ultrabook in Asia".

We say nsp, tsp and psp are *consistent* according to the notion in the phrase-based model (Koehn et al., 2003). For example, nsp(亚洲/NR), tsp(笔记本/NN) and $psp(\text{NP}_1)$ are consistent while nsp(超级/JJ) and nsp(笔记本/NN) are not consistent.

The annotation can be achieved by a single postorder transversal of the phrasal nodes labeled dependency tree. For simplicity, we call the annotated phrasal nodes labeled dependency tree *annotated dependency tree*. The extraction of bilingual phrases (including the translation of head node, dependency syntactic phrases and the fragment covered by a phrasal node) can be readily achieved by the algorithm described in Koehn et al., (2003). In the following, we focus on CHDR rules extraction.

3.2 Acceptable Fragments Identification

Before present the method of acceptable fragments identification, we give a brief description of CHDR fragments. A CHDR fragment is an annotated fragment that consists of a source head-dependents relation with/without constituency phrasal nodes, a target string and the word alignment information between the source and target side. We identify the acceptable CHDR fragments that are suitable for rule induction from the annotated dependency tree. We divide the acceptable CHDR fragments into two categories depending on whether the fragments contain phrasal nodes. If an acceptable CHDR fragment does not contain phrasal nodes, we call it *CHDR-normal fragment*, otherwise *CHDR-phrasal fragment*. Given a CHDR fragment F rooted at n, we say F is acceptable if it satisfies any one of the following properties:

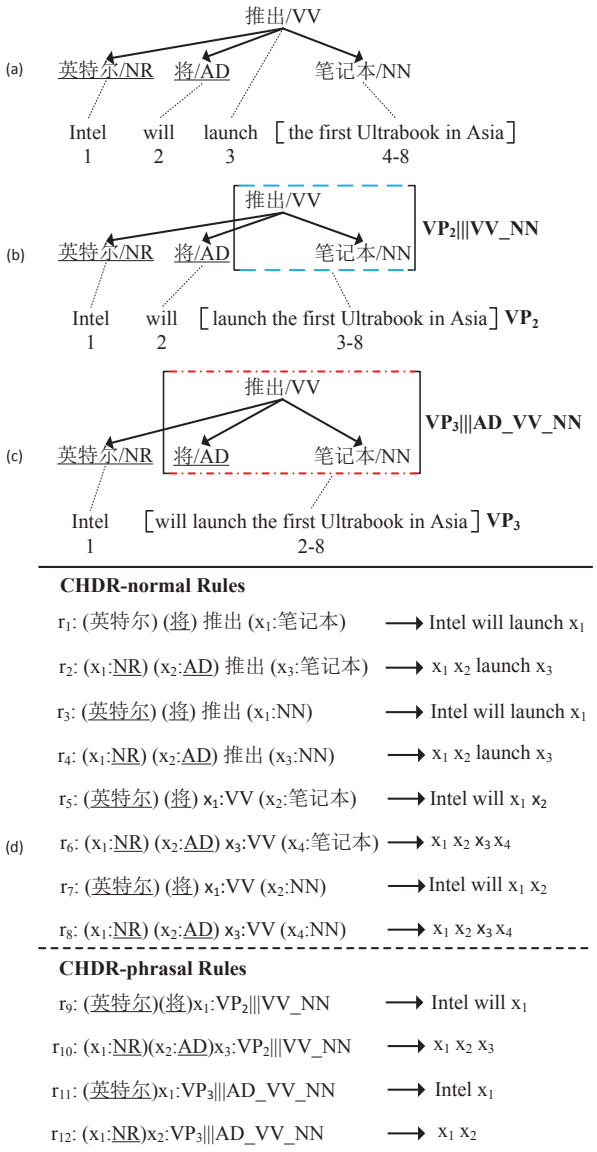


Figure 5: Examples of a CHDR-normal fragment (a), two CHDR-phrasal fragments (b) and (c) that are identified from the top level of the annotated dependency tree in Figure 4, and the corresponding CHDR rules (d) induced from (a), (b) and (c). The underline denotes a leaf node.

1. Without phrasal nodes, the node span of the root n is consistent and the subtree spans of n's all dependents are consistent. For example, Figure 5-(a) shows a CHDR-normal fragment that identified from the top level of the annotated dependency tree in Figure 4, since the nsp(推出/VV), tsp(英特尔/NR), tsp(将/AD) and tsp(笔记本/NN) are consistent.

2. With phrasal nodes, the phrasal spans of phrasal nodes are consistent; and for the other nodes, the node span of head (if it is not covered by any phrasal node) is consistent, and the subtree spans of dependents are consistent. For instance, Figure 5-(b) and (c) show two CHDR-phrasal fragments identified from the top level of Figure 4. In Figure 5-(b), $psp(\text{VP}_2)$, tsp(英特尔/NR) and tsp(将/AD) are consistent. In Figure 5-(c), $psp(\text{VP}_3)$ and tsp(英特尔/NR) are consistent.

The identification of acceptable fragments can be achieved by a single postorder transversal of the annotated dependency tree. Typically, each acceptable fragment contains at most three types of nodes: head node, head of the related CHDR; internal nodes, internal nodes of the related CHDR except head node; leaf nodes, leaf nodes of the related CHDR.

3.3 Rule Induction

From each acceptable CHDR fragment, we induce a set of lexicalized and generalized CHDR rules. We induce CHDR-normal rules and CHDR-phrasal rules from CHDR-normal fragments and CHDR-phrasal fragments, respectively.

We first induce a lexicalized form of CHDR rule from an acceptable CHDR fragment:

1. For a CHDR-normal fragment, we first mark the internal nodes as substitution sites. This forms the input of a CHDR-normal rule. Then we generate the target string according to the node span of the head and the subtree spans of the dependents, and turn the word sequences covered by the internal nodes into variables. This forms the output of a lexicalized CHDR-normal rule.

2. For a CHDR-phrasal fragment, we first mark the internal nodes and the phrasal nodes as substitution sites. This forms the input of a CHDR-phrasal rule. Then we construct the output of the CHDR-phrasal rule in almost the same way with constructing CHDR-normal rules, except that we replace the target sequences covered by the internal nodes and the phrasal nodes with variables.

For example, rule r_1 in Figure 5-(d) is a lexicalized CHDR-normal rule induced from the CHDR-normal fragment in Figure 5-(a). r_9 and r_{11} are CHDR-phrasal rules induced from the CHDR-phrasal fragment in Figure 5-(b) and Figure 5-(c) respectively. As we can see, these CHDR-phrasal rules are partially unlexicalized.

To alleviate the sparseness problem, we generalize the lexicalized CHDR-normal rules and partially unlexicalized CHDR-phrasal rules with unlexicalized nodes by the method proposed in Xie et al., (2011). As the modification relations between head and dependents are determined by the edges, we can replace the lexical word of each node with its category (POS tag) and obtain new head-dependents relations with unlexicalized nodes keeping the same modification relations. We generalize the rule by simultaneously turn the nodes of the same type (head, internal, leaf) into their categories. For example, CHDR-normal rules $r_2 \sim r_7$ are generalized from r_1 in Figure 5-(d). Besides, r_{10} and r_{12} are the corresponding generalized CHDR-phrasal rules. Actually, our CHDR rules are the superset of head-dependents relation rules in Xie et al., (2011). CHDR-normal rules are equivalent with the head-dependents relation rules and the CHDR-phrasal rules are the extension of these rules. For convenience of description, we use the subscript to distinguish the phrasal nodes with the same category, such as VP_2 and VP_3. In actual operation, we use VP instead of VP_2 and VP_3.

We handle the unaligned words of the target side by extending the node spans of the lexicalized head and leaf nodes, and the subtree spans of the lexicalized dependents, on both left and right directions. This procedure is similar with the method of Och and Ney, (2004). During this process, we might obtain $m(m \geq 1)$ CHDR rules from an acceptable fragment. Each of these rules is assigned with a fractional count $1/m$. We take the extracted rule set as observed data and make use of relative frequency estimator to obtain the translation probabilities $P(t|s)$ and $P(s|t)$.

4 Decoding and the Model

Following Och and Ney, (2002), we adopt a general loglinear model. Let d be a derivation that convert a

source phrasal nodes labeled dependency tree into a target string e. The probability of d is defined as:

$$P(d) \propto \prod_i \phi_i(d)^{\lambda_i} \quad (1)$$

where ϕ_i are features defined on derivations and λ_i are feature weights. In our experiments of this paper, the features are used as follows:

- CHDR rules translation probabilities $P(t|s)$ and $P(s|t)$, and CHDR rules lexical translation probabilities $P_{lex}(t|s)$ and $P_{lex}(s|t)$;
- bilingual phrases translation probabilities $P_{bp}(t|s)$ and $P_{bp}(s|t)$, and bilingual phrases lexical translation probabilities $P_{bplex}(t|s)$ and $P_{bplex}(s|t)$;
- rule penalty $exp(-1)$;
- pseudo translation rule penalty $exp(-1)$;
- target word penalty $exp(|e|)$;
- language model $P_{lm}(e)$.

We have twelve features in our model. The values of the first four features are accumulated on the CHDR rules and the next four features are accumulated on the bilingual phrases. We also use a pseudo translation rule (constructed according to the word order of head-dependents relation) as a feature to guarantee the complete translation when no matched rules can be found during decoding.

Our decoder is based on bottom-up chart-based algorithm. It finds the best derivation that convert the input phrasal nodes labeled dependency tree into a target string among all possible derivations. Given the source constituency tree and dependency tree, we first generate phrasal nodes labeled dependency tree T as described in Section 3.1, then the decoder transverses each node in T by postorder. For each node n, it enumerates all instances of CHDR rooted at n, and checks the rule set for matched translation rules. A larger translation is generated by substituting the variables in the target side of a translation rule with the translations of the corresponding dependents. Cube pruning (Chiang, 2007; Huang and Chiang, 2007) is used to find the k-best items with integrated language model for each node.

To balance the performance and speed of the decoder, we limit the search space by reducing the number of translation rules used for each node. There are two ways to limit the rule table size: by a fixed limit (rule-limit) of how many rules are retrieved for each input node, and by a threshold (rule-threshold) to specify that the rule with a score lower than β times of the best score should be discarded. On the other hand, instead of keeping the full list of candidates for a given node, we keep a top-scoring subset of the candidates. This can also be done by a fixed limit (stack-limit) and a threshold (stack-threshold).

5 Experiments

We evaluated the performance of our model by comparing with hierarchical phrase-based model (Chiang, 2007), constituency-to-string model (Liu et al., 2006) and dependency-to-string model (Xie et al., 2011) on Chinese-English translation. First, we describe data preparation (Section 5.1) and systems (Section 5.2). Then, we validate that our model significantly outperforms all the other baseline models (Section 5.3). Finally, we give detail analysis (Section 5.4).

5.1 Data Preparation

Our training data consists of 1.25M sentence pairs extracted from LDC [1] data. We choose NIST MT Evaluation test set 2002 as our development set, NIST MT Evaluation test sets 2003 (MT03), 2004 (MT04) and 2005 (MT05) as our test sets. The quality of translations is evaluated by the case insensitive NIST BLEU-4 metric [2].

We parse the source sentences to constituency trees (without binarization) and projective dependency trees with Stanford Parser (Klein and Manning, 2002). The word alignments are obtained by running GIZA++ (Och and Ney, 2003) on the corpus in both directions and using the "grow-diag-final-and" balance strategy (Koehn et al., 2003). We get bilingual phrases from word-aligned data with algorithm described in Koehn et al. (2003) by running Moses Toolkit [3]. We apply SRI Language Modeling Toolkit (Stolcke and others, 2002) to train a 4-gram

[1] Including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[2] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

[3] http://www.statmt.org/moses/

System	Rule #	MT03	MT04	MT05	Average
Moses-chart	116.4M	34.65	36.47	34.39	35.17
cons2str	25.4M+32.5M	33.14	35.12	33.27	33.84
dep2str	19.6M+32.5M	34.85	36.57	34.72	35.38
consdep2str	23.3M+32.5M	**35.57***	**37.68***	**35.62***	**36.29**

Table 1: Statistics of the extracted rules on training data and the BLEU scores (%) on the test sets of different systems. The "+" denotes that the rules are composed of syntactic translation rules and bilingual phrases (32.5M). The "*" denotes that the results are significantly better than all the other systems ($p<0.01$).

language model with modified Kneser-Ney smoothing on the Xinhua portion of the English Gigaword corpus. We make use of the standard MERT (Och, 2003) to tune the feature weights in order to maximize the system's BLEU score on the development set. The statistical significance test is performed by *sign-test* (Collins et al., 2005).

5.2 Systems

We take the open source hierarchical phrase-based system *Moses-chart* (with default configuration), our in-house constituency-to-string system *cons2str* and dependency-to-string system *dep2str* as our baseline systems.

For *cons2str*, we follow Liu et al., (Liu et al., 2006) to strict that the height of a rule tree is no greater than 3 and phrase length is no greater than 7. To keep consistent with our proposed model, we implement the dependency-to-string model (Xie et al., 2011) with GHKM (Galley et al., 2004) rule extraction algorithm and utilize bilingual phrases to translate source head node and dependency syntactic phrases. Our *dep2str* shows comparable performance with Xie et al., (2011), which can be seen by comparing with the results of hierarchical phrase-based model in our experiments. For *dep2str* and our proposed model *consdep2str*, we set rule-threshold and stack-threshold to 10^{-3}, rule-limit to 100, stack-limit to 300, and phrase length limit to 7.

5.3 Experimental Results

Table 1 illustrates the translation results of our experiments. As we can see, our *consdep2str* system has gained the best results on all test sets, with +1.12 BLEU points higher than *Moses-chart*, +2.45 BLEU points higher than *cons2str*, and +0.91 BLEU points higher than *dep2str*, averagely on MT03, MT04 and MT05. Our model significantly outperforms all the other baseline models, with $p<0.01$ on statistical significance test *sign-test* (Collins et al., 2005). By exploiting two types of trees on source side, our model gains significant improvements over constituency-to-string and dependency-to-string models, which employ single type of trees.

Table 1 also lists the statistical results of rules extracted from training data by different systems. According to our statistics, the number of rules extracted by our *consdep2str* system is about 18.88% larger than *dep2str*, without regard to the 32.5M bilingual phrases. The extra rules are CHDR-phrasal rules, which can bring in BLEU improvements by enhancing the compatibility with phrases. We will conduct a deep analysis in the next sub-section.

5.4 Analysis

In this section, we first illustrate the influence of CHDR-phrasal rules in our *consdep2str* model. We calculate the proportion of 1-best translations in test sets that employ CHDR-phrasal rules, and we call this proportion "*CHDR-phrasal Sent.*". Besides, the proportion of CHDR-phrasal rules in all CHDR rules is calculated in these translations, and we call this proportion "*CHDR-phrasal Rule*". Table 2 lists the using of CHDR-phrasal rules on test sets, showing that *CHDR-phrasal Sent.* on all test sets are higher than 50%, and *CHDR-phrasal Rule* on all three test sets are higher than 10%. These results indicate that CHDR-phrasal rules do play a role in decoding.

Furthermore, we compare some actual translations of our test sets generated by *cons2str*, *dep2str* and *consdep2str* systems, as shown in Figure 6. In the first example, the Chinese input holds long distance dependencies "联合国 已经 对 ... 加诸于 ... 表示 关切", which correspond to the sentence pattern "noun+adverb+prepositional

System	MT03	MT04	MT05
CHDR-phrasal Sent.	50.71	61.80	56.19
CHDR-phrasal Rule	10.53	13.55	10.83

Table 2: The proportion (%) of 1-best translations that employs CHDR-phrasal rules (CHDR-phrasal Sent.) and the proportion (%) of CHDR-phrasal rules in all CHDR rules in these translations (CHDR-phrasal Rule).

phrase+verb+noun". *Cons2str* gives a bad result with wrong global reordering, while our *consdep2str* system gains an almost correct result since we capture this pattern by CHDR-normal rules. In the second example, we can see that the Chinese phrase "再次 出现" is a non-syntactic phrase in the dependency tree, and this phrase can not be captured by head-dependents relation rules in Xie et al., (2011), thus can not be translated as one unit. Since we encode constituency phrasal nodes to the dependency tree, "再次 出现" is labeled by a phrasal node "VP" (means verb phrase), which can be captured by our CHDR-phrasal rules and translated into the correct result "reemergence" with bilingual phrases.

By combining the merits of constituency and dependency trees, our *consdep2str* model learns CHDR-normal rules to acquire the property of long distance reorderings and CHDR-phrasal rules to obtain good compatibility with phrases.

6 Related Work

In recent years, syntax-based models have witnessed promising improvements. Some researchers make efforts on constituency-based models (Graehl and Knight, 2004; Liu et al., 2006; Huang et al., 2006; Zhang et al., 2007; Mi et al., 2008; Liu et al., 2009; Liu et al., 2011; Zhai et al., 2012). Some works pay attention to dependency-based models (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Xiong et al., 2007; Shen et al., 2008; Xie et al., 2011). These models are based on single type of trees.

There are also some approaches combining merits of different structures. Marton and Resnik (2008) took the source constituency tree into account and added soft constraints to the hierarchical phrase-based model (Chiang, 2005). Cherry (2008) utilized dependency tree to add syntactic cohesion to the phrased-based model. Mi and Liu, (2010) proposed a constituency-to-dependency translation model, which utilizes constituency forests on the source side to direct the translation, and dependency trees on the target side to ensure grammaticality. Feng et al. (2012) presented a hierarchical chunk-to-string translation model, which is a compromise between the hierarchical phrase-based model and the constituency-to-string model. Most works make effort to introduce linguistic knowledge into the phrase-based model and hierarchical phrase-based model with constituency trees. Only the work proposed by Mi and Liu, (2010) utilized constituency and dependency trees, while their work applied two types of trees on two sides.

Instead, our model simultaneously utilizes constituency and dependency trees on the source side to direct the translation, which is concerned with combining the advantages of two types of trees in translation rules to advance the state-of-the-art machine translation.

7 Conclusion

In this paper, we present a novel model that simultaneously utilizes constituency and dependency trees on the source side to direct the translation. To combine the merits of constituency and dependency trees, our model employs head-dependents relations incorporating with constituency phrasal nodes. Experimental results show that our model exhibits good performance and significantly outperforms the state-of-the-art constituency-to-string, dependency-to-string and hierarchical phrase-based models. For the first time, source side constituency and dependency trees are simultaneously utilized to direct the translation, and the model surpasses the state-of-the-art translation models.

Since constituency tree binarization can lead to more constituency-to-string rules and syntactic phrases in rule extraction and decoding, which improve the performance of constituency-to-string systems, for future work, we would like to do research on encoding binarized constituency trees to dependency trees to improve translation performance.

Acknowledgments

The authors were supported by National Natural Science Foundation of China (Contracts 61202216),

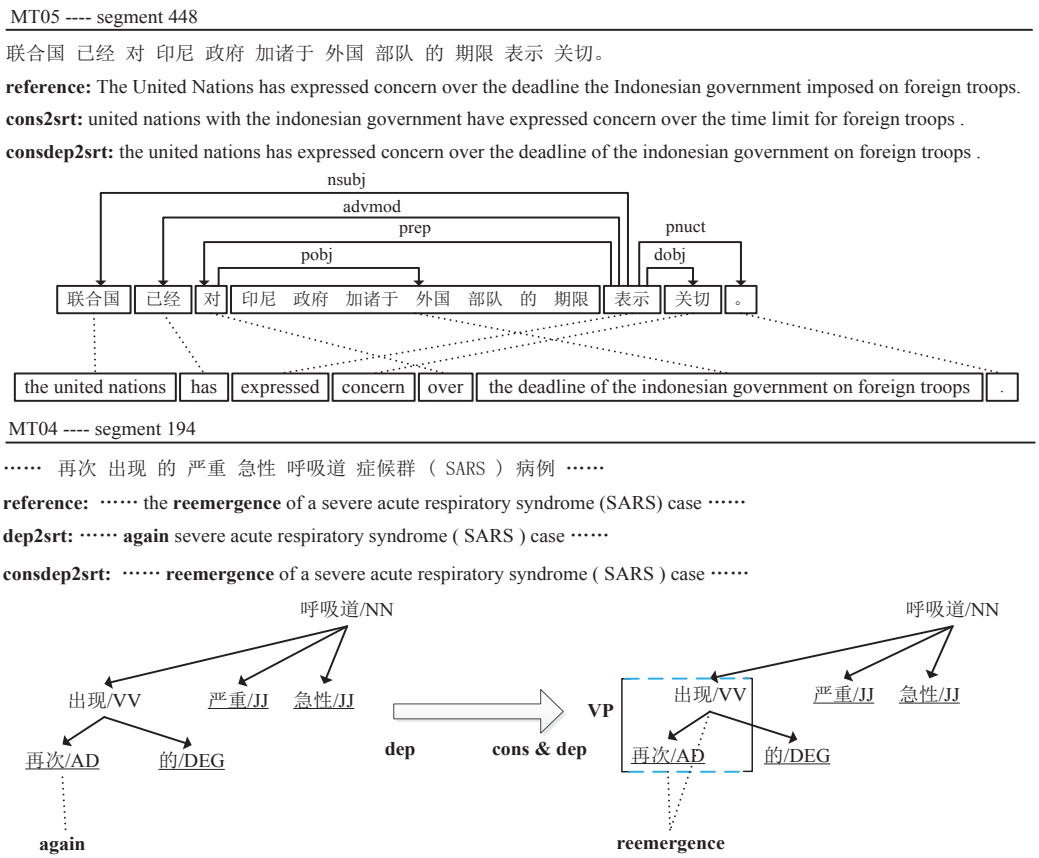


Figure 6: Actual examples translated by the *cons2str*, *dep2str* and *consdep2str* systems.

863 State Key Project (No. 2011AA01A207), and National Key Technology R&D Program (No. 2012BAH39B03), Key Project of Knowledge Innovation Program of Chinese Academy of Sciences (No. KGZD-EW-501). Qun Liu's work was partially supported by Science Foundation Ireland (Grant No. 07/CE/I1142) as part of the CNGL at Dublin City University. Sincere thanks to the anonymous reviewers for their thorough reviewing and valuable suggestions. We appreciate Haitao Mi, Zhaopeng Tu and Anbang Zhao for insightful advices in writing.

References

Colin Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *ACL*, pages 72–80.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270.

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

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 531–540.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 541–548.

Yang Feng, Dongdong Zhang, Mu Li, Ming Zhou, and Qun Liu. 2012. Hierarchical chunk-to-string translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 950–958.

Heidi J Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 304–3111.

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

ceedings of HLT/NAACL, volume 4, pages 273–280. Boston.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112.

Peter Hellwig. 2006. Parsing with dependency grammars. *An International Handbook of Contemporary Research*, 2:1081–1109.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Annual Meeting-Association For Computational Linguistics*, volume 45, pages 144–151.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, pages 66–73.

Dan Klein and Christopher D Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, volume 15, pages 3–10.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 625–630.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 609–616.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 558–566.

Yang Liu, Qun Liu, and Yajuan Lü. 2011. Adjoining tree-to-string translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1278–1287.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011.

Haitao Mi and Qun Liu. 2010. Constituency to dependency translation with forests. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1433–1442.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

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

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics*, 30(4):417–449.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 271–279.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–226.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2012. Tree-based translation without using parse trees. In *Proceedings of COLING 2012*, pages 3037–3054.

Min Zhang, Hongfei Jiang, AiTi Aw, Jun Sun, Sheng Li, and Chew Lim Tan. 2007. A tree-to-tree alignment-based model for statistical machine translation. *MT-Summit-07*, pages 535–542.

Monolingual Marginal Matching for Translation Model Adaptation

Ann Irvine
Johns Hopkins University
anni@jhu.edu

Chris Quirk
Microsoft Research
chrisq@microsoft.com

Hal Daumé III
University of Maryland
me@hal3.name

Abstract

When using a machine translation (MT) model trained on OLD-domain parallel data to translate NEW-domain text, one major challenge is the large number of out-of-vocabulary (OOV) and new-translation-sense words. We present a method to identify new translations of both known and unknown source language words that uses NEW-domain comparable document pairs. Starting with a joint distribution of source-target word pairs derived from the OLD-domain parallel corpus, our method recovers a new joint distribution that matches the marginal distributions of the NEW-domain comparable document pairs, while minimizing the divergence from the OLD-domain distribution. Adding learned translations to our French-English MT model results in gains of about 2 BLEU points over strong baselines.

1 Introduction

When a statistical machine translation (SMT) model trained on OLD-domain (e.g. parliamentary proceedings) parallel text is used to translate text in a NEW-domain (e.g. medical or scientific), performance degrades drastically. One of the major causes is the large number of NEW-domain words that are out-of-vocabulary (OOV) with respect to the OLD-domain text. Figure 1 shows the OOV rate for text in several NEW-domains, with respect to OLD-domain parliamentary proceedings. Even more challenging are the difficult-to-detect new-translation-sense (NTS) words: French words that are present in both the OLD and NEW domains but that are translated differently in each domain. For example, the French word *enceinte* is mostly translated in parliamentary proceedings as *place*, *house*, or *chamber*; in medical text, the translation is mostly *pregnant*; in scientific text, *enclosures*.

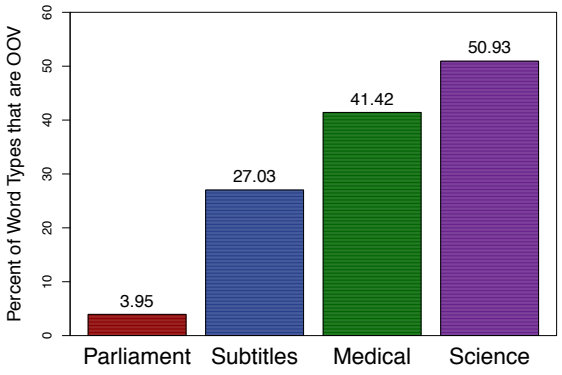


Figure 1: Percent of test set word types by domain that are OOV with respect to five million tokens of OLD-domain French parliamentary proceedings data.

One potential remedy is to collect parallel data in the NEW-domain, from which we can train a new SMT model. Smith et al. (2010), for example, mine parallel text from comparable corpora. Parallel sentences are informative but also rare: in the data released by Smith et al. (2010), only 21% of the foreign sentences have a near-parallel counterpart in the English article.[1] Furthermore, these sentences do not capture all terms. In that same dataset, we find that on average only 20% of foreign and 28% of English word types in a given article are represented in the parallel sentence pairs.

In this work, we seek to learn a joint distribu-

[1] Only 12% of sentences from generally longer English articles have a near-parallel counterpart in the foreign language.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1077–1088,
Seattle, Washington, USA, 18-21 October 2013.

tion of translation probabilities over all source and target word pairs in the NEW-domain. We begin with a maximum likelihood estimate of the joint based on a word aligned OLD-domain corpus and update this distribution using NEW-domain comparable data. We define a model based on a single comparable corpus and then extend it to learn from document aligned comparable corpora with any number of comparable *document pairs*. This approach allows us to identify translations for OOV words in the OLD-domain (e.g. French *cisaillement* and *perçage*, which translate as *shear* and *drilling*, in the scientific domain) as well as new translations for previously observed NTS words (e.g. *enceinte* translates as *enclosures*, not *place*, in the scientific domain). In our MT experiments, we use the learned NEW-domain joint distribution to update our SMT model with translations of OOV and low frequency words; we leave the integration of new translations for NTS words to future work.

Our approach crucially depends on finding comparable document pairs relevant to the NEW-domain. Such pairs could be derived from a number of sources, with document pairings inferred from timestamps (e.g. news articles) or topics (inferred or manually labeled). We use Wikipedia[2] as a source of comparable pairs. So-called "interwiki links" (which link Wikipedia articles written on the same topic but in different languages) act as rough guidance that pages may contain similar information. Our approach does not exploit any Wikipedia structure beyond this signal, and thus is portable to alternate sources of comparable articles, such as multilingual news articles covering the same event.

Our model also relies on the assumption that each comparable document pair describes generally the same concepts, though the order and structure of presentation may differ significantly. The efficacy of this method likely depends on the degree of comparability of the data; exploring the correlation between comparability and MT performance is an interesting question for future work.

2 Previous Work

In prior work (Irvine et al., 2013), we presented a systematic analysis of errors that occur when shifting domains in machine translation. That work concludes that errors resulting from unseen (OOV) and new translation sense words cause the majority of the degradation in translation performance that occurs when an MT model trained on OLD-domain data is used to translate data in a NEW-domain. Here, we target OOV errors, though our marginal matching method is also applicable to learning translations for NTS words.

A plethora of prior work learns bilingual lexicons from monolingual and comparable corpora with many signals including distributional, temporal, and topic similarity (Rapp, 1995; Fung and Yee, 1998; Rapp, 1999; Schafer and Yarowsky, 2002; Schafer, 2006; Klementiev and Roth, 2006; Koehn and Knight, 2002; Haghighi et al., 2008; Mimno et al., 2009; Mausam et al., 2010; Prochasson and Fung, 2011; Irvine and Callison-Burch, 2013). However, this prior work stops short of using these lexicons in translation. We augment a baseline MT system with learned translations.

Our approach bears some similarity to Ravi and Knight (2011), Dou and Knight (2012), and Nuhn et al. (2012); we learn a translation distribution despite a lack of parallel data. However, we focus on the domain adaptation setting. Parallel data in an OLD-domain acts as a starting point (prior) for this translation distribution. It is reasonable to assume an initial bilingual dictionary can be obtained even in low resource settings, for example by crowdsourcing (Callison-Burch and Dredze, 2010) or pivoting through related languages (Schafer and Yarowsky, 2002; Nakov and Ng, 2009).

Daumé III and Jagarlamudi (2011) mine translations for high frequency OOV words in NEW-domain text in order to do domain adaptation. Although that work shows significant MT improvements, it is based primarily on distributional similarity, thus making it difficult to learn translations for low frequency source words with sparse word context counts. Additionally, that work reports results using artificially created monolingual corpora taken from separate source and target halves of a NEW-domain parallel corpus, which may have more lexical overlap with the corresponding test set than we could expect from true monolingual corpora. Our work mines NEW-domain-like document pairs from Wikipedia. In this work, we show that, keeping

[2] www.wikipedia.org

data resources constant, our model drastically outperforms this previous approach. Razmara et al. (2013) take a fundamentally different approach and construct a graph using source language monolingual text and identify translations for source language OOV words by pivoting through paraphrases.

Della Pietra et al. (1992) and Federico (1999) explore models for combining foreground and background distributions for the purpose of language modeling, and their approaches are somewhat similar to ours. However, our focus is on translation.

3 Model

Our goal is to recover a probabilistic translation dictionary in a NEW-domain, represented as a joint probability distribution $p^{\text{new}}(s,t)$ over source/target word pairs. At our disposal, we have access to a joint distribution $p^{\text{old}}(s,t)$ from the OLD-domain (computed from word alignments), plus comparable document pairs in the NEW-domain. From these comparable documents, we can extract raw word frequencies on both the source and target side, represented as marginal distributions $q(s)$ and $q(t)$. The key idea is to estimate this NEW-domain joint distribution to be as similar to the OLD-domain distribution as possible, subject to the constraint that its marginals match those of q.

To illustrate our goal, consider an example. Imagine in the OLD-domain parallel data we find that *accorder* translates as *grant* 10 times and as *tune* 1 time. In the NEW-domain comparable data, we find that *accorder* occurs 5 times, but *grant* occurs only once, and *tune* occurs 4 times. Clearly *accorder* no longer translates as *grant* most of the time; perhaps we should shift much of its mass onto the translation *tune* instead. Figure 2 shows the intuition.

First, we present an objective function and set of constraints over joint distributions to minimize the divergence from the OLD-domain distribution while matching both the source and target NEW-domain marginal distributions. Next, we augment the objective with information about word string similarity, which is particularly useful for the French-English language pair. Optimizing this objective with a single pair of source and target marginals can be performed using an off-the-shelf solver. In practice, though, we have a large set of document pairs, each of which can induce a pair of marginals. Using these per-document marginals provides additional information to the learning function but would overwhelm a common solver. Therefore, we present a sequential learning method for approximately matching the large set of document pair marginal distributions. Finally, we describe how we identify comparable document pairs relevant to the NEW-domain.

3.1 Marginal Matching Objective

Given word-aligned parallel data in the OLD-domain and source and target comparable corpora in the NEW-domain, we first estimate a joint distribution $p^{\text{old}}(s,t)$ over word pairs (s,t) in the OLD-domain, where s and t range over source and target language words, respectively. For the OLD-domain joint distribution, we use a simple maximum likelihood estimate based on non-null automatic word alignments (using grow-diag-final GIZA++ alignments (Och and Ney, 2003)). Next, we find source and target marginal distributions, $q(s)$ and $q(t)$, by relative frequency estimates over the source and target comparable corpora. Our goal is to recover a joint distribution $p^{\text{new}}(s,t)$ for the new domain that matches the marginals, $q(s)$ and $q(t)$, but is minimally different from the original joint distribution, $p^{\text{old}}(s,t)$.

We cast this as a linear programming problem:

$$p^{\text{new}} = \arg\min_{p} \left\| p - p^{\text{old}} \right\|_1 \tag{1}$$

$$\text{subject to: } \sum_{s,t} p(s,t) = 1, \quad p(s,t) \geq 0$$

$$\sum_{s} p(s,t) = q(t), \quad \sum_{t} p(s,t) = q(s)$$

In the objective function, the joint probability matrices p and p^{old} are interpreted as large vectors over all word pairs (s,t). The first two constraints force the result to be a well-formed distribution, and the final two force the marginals to match.

Following prior work (Ravi and Knight, 2011), we would like the matrix to remain as sparse as possible; that is, introduce the smallest number of new translation pairs necessary. A regularization term captures this goal:

$$\Omega(p) = \sum_{\substack{s,t: \\ p^{\text{old}}(s,t)=0}} \lambda_r \times p(s,t) \tag{2}$$

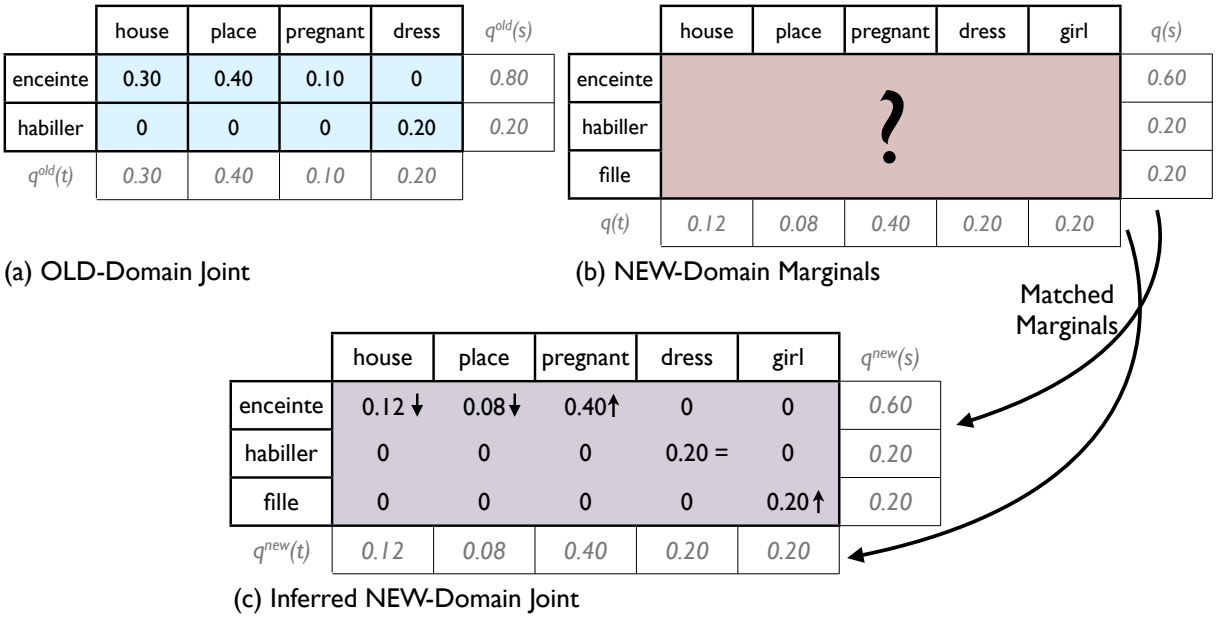


Figure 2: Starting with a joint distribution derived from OLD-domain data, we infer a NEW-domain joint distribution based on the intuition that the new joint should match the marginals that we observe in NEW-domain comparable corpora. In this example, a translation is learned for the previously OOV word *fille*, and *pregnant* becomes a preferred translation for *enceinte*.

If the old domain joint probability $p^{\text{old}}(s,t)$ was nonzero, there is no penalty. Otherwise, the penalty is λ_r times the new joint probability $p(s,t)$. To discourage the addition of translation pairs that are unnecessary in the new domain, we use a value of λ_r greater than one. Thus, the benefit of a more sparse matrix overwhelms the desire for preventing change. Any value greater than one seems to suffice; we use $\lambda_r = 1.1$ in our experiments.

Inspired by the preference for sparse matrices captured by $\Omega(p)$, we include another orthogonal cue that words are translations of one another: their string similarity. In prior work, string similarity was a valuable signal for inducing translations, particularly for closely related languages such as French and English (Daumé III and Jagarlamudi, 2011). We define a penalty function $f(p)$ as follows: if the normalized Levenshtein edit distance between s *without accents* and t is less than 0.2, no penalty is applied; a penalty of 1 is applied otherwise. We chose the 0.2 threshold manually by inspecting results on our development sets.

$$f(p) = \sum_{s,t} p(s,t) \cdot \begin{cases} 0 & \text{if } \frac{\text{lev}(t,\text{strip}(s))}{\text{len}(s)+\text{len}(t)} < 0.2 \\ 1 & \text{otherwise} \end{cases}$$

The objective function including this penalty is:

$$p^{\text{new}} = \arg\min_{p} \left\| p - p^{\text{old}} \right\|_1 + \Omega(p) + f(p)$$

In principle, additional penalties could be encoded in a similar way.[3] This objective can be optimized by any standard LP solver; we use the Gurobi package (Gurobi Optimization Inc., 2013).

3.2 Document Pair Modification

The above formulation applies whenever we have access to comparable *corpora*. However, often we have access to comparable *documents*, such as those given by Wikipedia inter-language links. We modify our approach to take advantage of the document correspondences within our comparable corpus. In particular, we would like to match the marginals for *all document pairs*.[4] By maintaining separate marginal distributions, our algorithm is presented with more

[3] We experimented with penalties measuring document-pair co-occurrence and monolingual frequency differences but did not see gains on our development sets.

[4] This situation is not unique to our application; multiple marginals are likely to exist in many cases.

information. For example, imagine that one document pair uses "dog" and "chien", where another document pair uses "cat" and "chat", each with similar frequency. If we sum these marginals to produce a single marginal distribution, it is now difficult to identify that "dog" should correspond to "chien" and not "chat." Document pair alignments add information at the cost of additional constraints.

An initial formulation of our problem with multiple comparable document pairs might require the p^{new} marginals to match *all* of the document marginals. In general, this constraint set is likely to result in an infeasible problem. Instead, we take an incremental, online solution, considering a single comparable document pair at a time. For document pair k, we solve the optimization problem in Eq (1) to find the joint distribution minimally different from $p^{\text{k-1}}$, while matching the marginals of *this pair only*. This gives a new joint distribution, tuned specifically for this pair. We then update our current guess of the new domain joint *toward* this document-pair-specific distribution, much like a step in stochastic gradient ascent.

More formally, suppose that before processing the kth document we have a guess at the NEW-domain joint distribution, $p^{\text{new}}_{1:k-1}$ (the subscript indicates that it includes *all* document pairs up to and including document $k-1$). We first solve Eq (1) solely on the basis of this document pair, finding a joint distribution p^{new}_{k} that matches the marginals of the kth document pair *only* and is minimally different from $p^{\text{new}}_{1:k-1}$. Finally, we form a new estimate of the joint distribution by moving $p^{\text{new}}_{1:k-1}$ in the direction of p^{new}_{k}, via:

$$p^{\text{new}}_{1:k} = p^{\text{new}}_{1:k-1} + \eta_u \left[p^{\text{new}}_{k} - p^{\text{new}}_{1:k-1} \right]$$

The learning rate η_u is set to 0.001.[5]

This incremental update of parameters is similar to the margin infused relaxed algorithm (MIRA) (Crammer et al., 2006). Like MIRA and the perceptron, there is not an overall "objective" function that we are attempting to optimize (as one would in many stochastic gradient steps). Instead, we're aiming for a solution that makes a small amount of progress on each example, in such a way if it received that example again, it would "do better" (in this case: have a closer match of marginals). Also like MIRA, our learning rate is constant. We parallelize learning with mini-batches for increased speed. Eight parallel learners update an initial joint distribution based on 100 document pairs (i.e. each learner makes 100 incremental updates), and then we merge results using an average over the 8 learned joint distributions.

3.3 Comparable Data Selection

It remains to select comparable document pairs. We assume that we have enough monolingual NEW-domain data in one language to rank comparable document pairs (here, Wikipedia pages) according to how NEW-*domain-like* they are. In particular, we estimate the similarity to a source language (here, French) corpus in the NEW domain. For our experiments, we use the French side of a NEW-domain parallel corpus.[6] We could have targeted our learning even more by using our NEW-domain MT test sets. Doing so would increase the chances that our source language words of interest appear in the comparable corpus. However, to avoid overfitting any particular test set, we use the French side of the training data.

For each Wikipedia document pair, we compute the percent of French phrases up to length four that are observed in the French monolingual NEW-domain corpus and rank document pairs by the geometric mean of the four overlap measures. More sophisticated ways to identify NEW-domain-like Wikipedia pages (e.g. Moore and Lewis (2010)) may yield additional performance gains, but, qualitatively, the ranked Wikipedia pages seemed reasonable to the authors.

4 Experimental setup

4.1 Data

We use French-English Hansard parliamentary proceedings[7] as our OLD-domain parallel corpus. With over 8 million parallel lines of text, it is one of the largest freely available parallel corpora for any lan-

[5] We tuned η_u on semi-extrinsic results on the development set. Note that although 0.001 seems small, the values we are moving are *joint probabilities,* which are tiny and so small learning rates make sense.

[6] We could have, analogously, used the *target language* (English) side of the parallel corpus and measure overlap with the English Wikipedia documents, or even used both.

[7] http://www.parl.gc.ca

guage pair. In order to simulate more typical data settings, we sample every 32nd line, using the resulting parallel corpus of $253,387$ lines and $5,051,016$ tokens to train our baseline model.

We test our model using three NEW-domain corpora: (1) the EMEA medical corpus (Tiedemann, 2009), (2) a corpus of scientific abstracts (Carpuat et al., 2013a), and (3) a corpus of translated movie subtitles (Tiedemann, 2009). We use development and test sets to tune and evaluate our MT models. We use the NEW-domain parallel training corpora *only* for language modeling and for identifying NEW-domain-like comparable documents.

4.2 Machine translation

We use the Moses MT framework (Koehn et al., 2007) to build a standard statistical phrase-based MT model using our OLD-domain training data. Using Moses, we extract a phrase table with a phrase limit of five words and estimate the standard set of five feature functions (phrase and lexical translation probabilities in each direction and a constant phrase penalty feature). We also use a standard lexicalized reordering model and two language models based on the English side of the Hansard data and the given NEW-domain training corpora. Features are combined using a log-linear model optimized for BLEU, using the n-best batch MIRA algorithm (Cherry and Foster, 2012). We call this the "simple baseline." In Section 5.2 we describe several other baseline approaches.

4.3 Experiments

For each domain, we use the marginal matching method described in Section 3 to learn a new, domain-adapted joint distribution, $p_k^{\mathsf{new}}(s,t)$, over all French and English words. We use the learned joint to compute conditional probabilities, $p_k^{\mathsf{new}}(t|s)$, for each French word s and rank English translations t accordingly. First, we evaluate the learned joint directly using the distribution based on the word-aligned NEW-domain development set as a gold standard. Then, we perform end-to-end MT experiments. We supplement phrase tables with translations for OOV and low frequency words (we experiment with training data frequencies less than 101, 11, and 1) and include $p_k^{\mathsf{new}}(t|s)$ and $p_k^{\mathsf{new}}(s|t)$ as new translation features for those supplemental translations. For these new phrase pairs, we use the average lexicalized reordering values from the existing reordering tables. For phrase pairs extracted bilingually, we use the bilingually estimated translation probabilities and uniform scores for the new translation features. We experimented with using $p_k^{\mathsf{new}}(t|s)$ and $p_k^{\mathsf{new}}(s|t)$ to estimate additional lexical translation probabilities for the bilingually extracted phrase pairs but did not observe any gains (experimental details omitted due to space constraints). We re-run tuning in all experiments.

We also perform oracle experiments in which we identify translations for French words in word-aligned development and test sets and append these translations to baseline phrase tables.

5 Results

5.1 Semi-extrinsic evaluation

Before doing end-to-end MT experiments, we evaluate our learned joint distribution, $p_k^{\mathsf{new}}(s,t)$, by comparing it to the joint distribution taken from a word aligned NEW-domain parallel development set, $p^{\mathsf{gold}}(s,t)$. We call this evaluation semi-extrinsic because it involves neither end-to-end MT (our extrinsic task) nor an intrinsic evaluation based on our training objective (L1 norm). We find it informative to evaluate the models using bilingual lexicon induction metrics before integrating our output into full MT. That is, we do not compare the full joint distributions, but, rather, for a given French word, how our learned model ranks the word's most probable translation under the gold distribution. In particular, because we are primarily concerned with learning translations for previously unseen words, we evaluate over OOV French word types. In some cases, the correct translation for OOV words is the identical string (e.g. *na+*, *lycium*). Because it is trivial to produce these translations,[8] we evaluate over the subset of OOV development set French words for which the correct translation is not the same string.

Figure 3 shows the mean reciprocal rank for the learned distribution, $p_k^{\mathsf{new}}(s,t)$, for each domains as a function of the number of comparable document pairs used in learning. In all domains, the comparable document pairs are sorted according to their sim-

[8] And, indeed, by default our decoder copies OOV strings into its output directly.

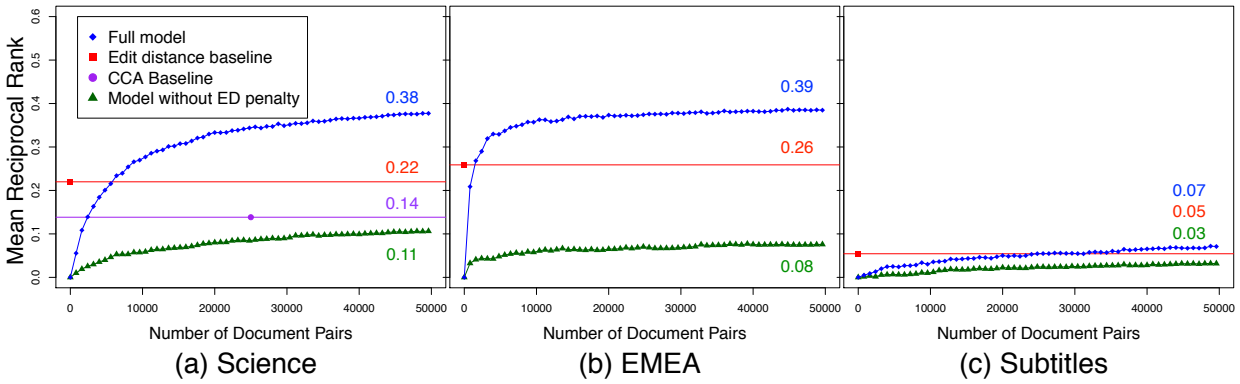


Figure 3: Semi-extrinsic bilingual lexicon induction results. Mean reciprocal rank is computed over all OOV development set words for which identity is not the correct translation.

ilarity with the NEW-domain. Figure 3 also shows the performance of baseline models and our learner without the edit distance penalty. For each source word s, the edit distance (ED) baseline ranks all English words t in our monolingual data by their edit distance with s.[9] The Canonical Correlation Analysis (CCA) baseline uses the approach of Daumé III and Jagarlamudi (2011) and the top $25,000$ ranked document pairs as a comparable corpus. That model performs poorly largely because of sparse word context counts. Interestingly, for Science and EMEA, the performance of our full model at $50,000$ document pairs is higher than the sum of the edit distance baseline and the model without the edit distance penalty, indicating that our approach effectively combines the marginal matching and edit distance signals.

The learning curves for the three domains vary substantially. For Science, learning is gradual and it appears that additional gains could be made by iterating over even more document pairs. In contrast, the model learns quickly for the EMEA domain; performance is stable after $20,000$ document pairs. Given these results and our experience with the two domains, we hypothesize that the difference is due to the fact that the Science data is much more heterogenous than the EMEA data. The Science data includes physics, chemistry, and biology abstracts, among others. The drug labels that make up most of the EMEA data are more homogeneous. In Section 6 we comment on the poor Subtitles performance, which persists in our MT experiments.

We experimented with making multiple learning passes over the document pairs and observed relatively small gains from doing so. In all experiments, learning from some number of additional new document pairs resulted in higher semi-extrinsic performance gains than passing over document pairs which were already observed.

In the case of OOV words, it's clear that learning something about how to translate a previously unobserved French word is beneficial. However, our learning method also learns domain-specific new-translation senses (NTS). Table 1 shows some examples of what the marginal matching method learns for different types of source words (OOVs, low frequency, and NTS).

5.2 MT evaluation

By default, the Moses decoder copies OOV words directly into its translated output. In some cases, this is correct (e.g. *ensembles*, *blumeria*, *google*). In other cases, French words can be translated into English correctly by simply stripping accent marks off of the OOV word and then copying it to the output (e.g. *caméra*, *éléments*, *molécules*). In the Science and EMEA domains, we found that our baseline BLEU scores improved from 21.91 to 22.20 and 23.67 to 24.45, respectively, when we changed the default handling of OOVs to strip accents before

[9] In particular, for each domain and each OOV French word, we ranked the set of all English words that appeared at least five times in the set of 50,000 most NEW-domain like Wikipedia pages. Using a frequency threshold of five helped eliminate French words and improperly tokenized English words from the set of candidates.

French	OLD top $p^{\text{old}}(t\|s)$	NEW top $p^{\text{gold}}(t\|s)$	MM-learned top $p^{\text{new}}(t\|s)$
OOV words			
cisaillement	-	shear strength shearing	shear viscous newtonian
courbure	-	curvature bending curvatures	curvature curved manifold
Low frequency words			
linéaires	linear	linear nonlinear non-linear	linear linearly nonlinear
récepteur	receiver	receptor receiver y1	receptor receiver receptors
New translation sense words			
champ	field jurisdiction scope	field magnetic near-field	field magnetic fields
marche	working march work	walk step walking	march walk walking

Table 1: Hand-picked examples of Science-domain French words and their top English translations in the OLD-domain, NEW-domain, and marginal matching distributions. The first two are OOVs. The next two only appeared four and one time, respectively, in the training data and only aligned to a single English word. The last two are NTS French words: words that appeared frequently in the training data but for which the word's sense in the new domain shifts.

copying into the output. Interestingly, performance on the Subtitles domain text did not change at all with this baseline modification. This is likely due to the fact that there are fewer technical OOVs (the terms typically captured by this accent-stripping pattern) in the subtitles domain.

Throughout our experiments, we found it critical to retain correct 'freebie' OOV translations. In the results presented below, including the baselines, we supplement phrase tables with a new candidate translation but also include accent-stripped identity, or 'freebie,' translations in the table for all OOV words. We experimented with classifying French words as freebies or needing a new translation, but oracle experiments showed very little improvement (about 0.2 BLEU improvement in the Science domain), so instead we simply include both types of translations in the phrase tables.

In addition to the strip-accents baseline, we compare results with four other baselines. First, we drop OOVs from the output translations. Second, like our semi-extrinsic baseline, we rank English words by their edit distance away from each French OOV word (ED baseline). Third, we rank English words by their document-pair co-occurrence score with each French OOV word. That is, for all words w, we compute $D(w)$, the vector indicating the document pairs in which w occurs, over the set of 50,000 document-pairs which are most NEW-*domain-like*. For French and English words s and t, if $D(s)$ and $D(t)$ are dissimilar, it is less likely (s, t) is a valid translation pair. We weight $D(w)$ entries with BM25 (Robertson et al., 1994). For all French OOVs, we rank all English translations according to the cosine similarity between the pair of $D(w)$ vectors. The fourth baseline uses the CCA model described in Daumé III and Jagarlamudi (2011) to rank English words according to their distributional similarity with each French word. For the CCA baseline comparison, we only learned translations using 25,000 Science-domain document pairs, rather than the full 50,000 and for all domains. However, it's unlikely that learning over more data would overcome the low performance observed so far. For the final three baselines, we append French OOV words and their highest ranked English translation to the phrase table. Along with each new translation pair, we include one new phrase table feature with the relevant translation score (edit distance, document similarity, or CCA distributional similarity). For all baselines other than drop-OOVs, we also include accent-stripped translation pairs with an additional indicator feature.

Table 3 shows results appending the top ranked English translation for each OOV French word using each baseline method. None of the alternate baselines outperform the simplest baseline on the subtitles data. Using document pair co-occurrences is the strongest baseline for the Science and EMEA domains. This confirms our intuition that taking advantage of document pair alignments is worthwhile. For Science and EMEA, supplementing a model with OOV translations learned through our marginal matching method drastically outperforms all base-

OOVs translated correctly and incorrectly	
Input	les résistances au **cisaillement** par **poinçonnement** ...
Ref	the punching **shear strengths**...
Baseline	the resistances in **cisaillement** by **poinconnement** ...
MM	the resistances in **shear reinforcement**...
OOV translated incorrectly	
Input	présentation d' un logiciel permettant de gérer les données **temporelles** .
Ref	presentation of software which makes it possible to manage **temporal** data .
Baseline	introduction of a software to manage **temporelles** data .
MM	introduction of a software to manage data **plugged** .
Low frequency French words	
Input	...limite est liée à la **décroissance** très rapide du **couplage** électron-phonon avec la température .
Ref	...limit is linked to the rapid **decrease** of the electron-phonon **coupling** with temperature .
Baseline	...limit is linked to the **decline** very rapid electron-phonon **linkage** with the temperature .
MM	...limit is linked to the **linear** very rapid electron-phonon **coupling** with the temperature .

Table 2: Example MT outputs for Science domain. The baseline strips accents (Table 3). In the first example, the previously OOV word *cisaillement* is translated correctly by an MM-supplemented model. The OOV *poinçonnement* is translated as *reinforcement* instead of *strengths*, which is incorrect with respect to the reference but arguably not bad. In the second example, *temporelles* is not translated correctly in the MM output. In the third example, the MM-hypothesized correct translation of low frequency word *couplage*, *coupling*, is chosen instead of incorrect *linkage*. Also in the third example, the low frequency word *décroissance* is translated as the MM-hypothesized incorrect translation *linear*. In the case of *décroissance*, the baseline's translation, *decline*, is much better than the MM translation *linear*.

lines. Using our model to translate OOV words yields scores of 23.62 and 26.97 in the Science and EMEA domains, or 1.19 and 1.94 BLEU points, respectively, above the strongest baseline. We observe additional gains by also supplementing the model with translations for low frequency French words. For example, when we use our approach to translate source words in the Science domain which appear ten or fewer times in our OLD-domain training data, the BLEU score increases to 24.28.

We tried appending top-k translations, varying k. However, we found that for the baselines as well as our MM translations, using only the top-1 English translations outperformed using more.

Table 3 also shows the result of supplementing a baseline phrase table with oracle OOV translations. Using the marginal matching learned OOV translations takes us 30% and 40% of the way from the baseline to the oracle upper bound for Science and EMEA, respectively.

We have focused on supplementing an SMT model trained on a sample of the Hansard parallel corpus in order to mimic typical data conditions, but we have also performed experiments supplementing a model trained on the full dataset.[10] Beginning with the larger model, we observe performance gains of 0.8 BLEU points for both the EMEA and the Science domains over the strongest baselines, which are based on document similarity, when we add OOV

	Science	EMEA	Subs
Simple Baseline	21.91	23.67	**13.18**
Drop OOVs	20.22	18.95	11.86
Accent-Stripped	22.20	24.45	13.13
ED Baseline	22.10	24.35	12.95
Doc Sim Baseline.	**22.43**	**25.03**	13.02
CCA Baseline	21.41	-	-
MM Freq<1 (OOV)	23.62	26.97	**13.07**
MM Freq<11	**24.28**	**27.26**	12.97
MM Freq<101	23.96	26.82	12.92
Oracle OOV	26.38	29.99	15.06

Table 3: BLEU results using: (1) baselines, (2) phrase tables augmented with top-1 translations for French words with indicated OLD training data frequencies, (3) phrase tables augmented with OOV oracle translations.

[10] We still use the joint that was learned starting with the one estimated over the sample; we may observe greater gains over the full Hansard baseline with a stronger initial joint.

translations. As expected, these gains are less than what we observe when our baseline model is estimated over less data, but they are still substantial.

In all experiments, we have assumed that we have no NEW-domain parallel training data, which is the case for the vast majority of language pairs and domains. However, In the case that we do have some NEW-domain parallel data, OOV rates will be somewhat lower, but our method is still applicable. For example, we would need 2.3 million words of Science (NEW-domain) parallel data to cover just 50% of the OOVs in our Science test set, and 4.3 million words to cover 70%.

6 Discussion

BLEU score performance gains are substantial for the Science and EMEA domains, but we don't observe gains on the subtitles text. We believe this difference relates to the difference between a corpus domain and a corpus register. As Lee (2002) explains, a text's *domain* is most related to its topic, while a text's *register* is related to its type and purpose. For example, religious, scientific, and dialogue texts may be classified as separate registers, while political and scientific expositions may have a single register but different domains. Our science and EMEA corpora are certainly different in domain from the OLD-domain parliamentary proceedings, and our success in boosting MT performance with our methods indicates that the Wikipedia comparable corpora that we mined match those domains well. In contrast, the subtitles data differs from the OLD-domain parliamentary proceedings in both domain and register. Although the Wikipedia data that we mined may be closer in domain to the subtitles data than the parliamentary proceedings,[11] its register is certainly not film dialogues.

Although the use of marginal matching is, to the best of our knowledge, novel in MT, there are related threads of research that might inspire future work. The intuition that we should match marginal distributions is similar to work using no example labels but only label proportions to estimate labels, for example in Quadrianto et al. (2008). Unlike that work, our label set corresponds to entire vocabularies, and we have multiple observed label proportions. Also, while the marginal matching objective seems effective in practice, it is difficult to optimize. A number of recently developed approximate inference methods use a decomposition that bears a strong resemblance to this objective function. Considering the marginal distributions from each document pair to be a separate subproblem, we could approach the global objective of satisfying all subproblems as an instance of dual decomposition (Sontag et al., 2010) or ADMM (Gabay and Mercier, 1976; Glowinski and Marrocco, 1975).

We experiment with French-English because tuning and test sets are available in several domains for that language pair. However, our techniques are directly applicable to other language pairs, including those that are less related. We have observed that many domain-specific terms, particularly in medical and science domains, are borrowed across languages, whether or not the languages are related. Even for languages with different character sets, one could do transliteration before measuring orthographical similarity.

Although we were able to identify translations for some NTS words (Table 1), we did not make use of them in our MT experiments. Recent work has identified NTS words in NEW-domain corpora (Carpuat et al., 2013b), and in future work we plan to incorporate discovered translations for such words into MT.

7 Conclusions

We proposed a model for learning a joint distribution of source-target word pairs based on the idea that its marginals should match those observed in NEW-domain comparable corpora. Supplementing a baseline phrase-based SMT model with learned translations results in BLEU score gains of about two points in the medical and science domains.

Acknowledgments

We gratefully acknowledge the support of the 2012 JHU Summer Workshop and NSF Grant No 1005411. We would like to thank the entire DAMT team (`http://hal3.name/damt/`) and Sanjeev Khudanpur for their help and suggestions. We also acknowledge partial support from DARPA

[11] In fact, we believe that it is. Wikipedia pages that ranked very high in our subtitles-like list included, for example, the movie *The Other Side of Heaven* and actor *Frank Sutton*.

CSSG Grant D11AP00279 and DARPA BOLT Contract HR0011-12-C-0015 for Hal Daumé III and support from the Johns Hopkins University Human Language Technology Center of Excellence for Ann Irvine. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

References

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Marine Carpuat, Hal Daumé III, Alexander Fraser, Chris Quirk, Fabienne Braune, Ann Clifton, Ann Irvine, Jagadeesh Jagarlamudi, John Morgan, Majid Razmara, Aleš Tamchyna, Katharine Henry, and Rachel Rudinger. 2013a. Domain adaptation in machine translation: Final report. In *2012 Johns Hopkins Summer Workshop Final Report*.

Marine Carpuat, Hal Daumé III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013b. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December.

Hal Daumé III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

S. Della Pietra, V. Della Pietra, R. L. Mercer, and S. Roukos. 1992. Adaptive language modeling using minimum discriminant estimation. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Marcello Federico. 1999. Efficient language model adaptation through mdi estimation. In *Proceedings of EUROSPEECH*.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Daniel Gabay and Bertrand Mercier. 1976. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17 – 40.

Roland Glowinski and A. Marrocco. 1975. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité, d'une classe de problèmes de dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*, 140:41–76.

Gurobi Optimization Inc. 2013. Gurobi optimizer reference manual.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ann Irvine, John Morgan, Marine Carpuat, Hal Daumé III, and Dragos Munteanu. 2013. Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics (TACL)*.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *ACL Workshop on Unsupervised Lexical Acquisition*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

David Lee. 2002. Genres, registers, text types, domains and styles: Clarifying the concepts and navigating a path through the bnc jungle. *Language and Computers*, 42(1):247–292.

Mausam, Stephen Soderland, Oren Etzioni, Daniel S. Weld, Kobi Reiter, Michael Skinner, Marcus Sammer,

and Jeff Bilmes. 2010. Panlingual lexical translation via probabilistic inference. *Artificial Intelligence*, 174:619–637, June.

David Mimno, Hanna Wallach, Jason Naradowsky, David Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

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

Emmanuel Prochasson and Pascale Fung. 2011. Rare word translation extraction from aligned comparable documents. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Novi Quadrianto, Alex J. Smola, Tiberio S. Caetano, and Quoc V. Le. 2008. Estimating labels from label proportions. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Majid Razmara, Maryam Siahbani, Gholamreza Haffari, and Anoop Sarkar. 2013. Graph propagation for paraphrasing out-of-vocabulary words in statistical machine translation. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *Proceedings of the Text REtrieval Conference*.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Charles Schafer. 2006. *Translation Discovery Using Diverse Similarity Measures*. Ph.D. thesis, Johns Hopkins University.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

David Sontag, A. Globerson, and Tommi Jaakola, 2010. *Introduction to dual decomposition for inference*, chapter 1. MIT Press.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing (RANLP)*.

Efficient Left-to-Right Hierarchical Phrase-based Translation with Improved Reordering

Maryam Siahbani, Baskaran Sankaran, Anoop Sarkar
Simon Fraser University
Burnaby BC. CANADA
{msiahban,baskaran,anoop}@cs.sfu.ca

Abstract

Left-to-right (LR) decoding (Watanabe et al., 2006b) is a promising decoding algorithm for hierarchical phrase-based translation (Hiero). It generates the target sentence by extending the hypotheses only on the right edge. LR decoding has complexity $O(n^2b)$ for input of n words and beam size b, compared to $O(n^3)$ for the CKY algorithm. It requires a single language model (LM) history for each target hypothesis rather than two LM histories per hypothesis as in CKY. In this paper we present an augmented LR decoding algorithm that builds on the original algorithm in (Watanabe et al., 2006b). Unlike that algorithm, using experiments over multiple language pairs we show two new results: our LR decoding algorithm provides demonstrably more efficient decoding than CKY Hiero, four times faster; and by introducing new distortion and reordering features for LR decoding, it maintains the same translation quality (as in BLEU scores) obtained phrase-based and CKY Hiero with the same translation model.

1 Introduction

Hiero (Chiang, 2007) models translation using a lexicalized synchronous context-free grammar (SCFG) extracted from word aligned bitexts. Typically, CKY-style decoding is used for Hiero with time complexity $O(n^3)$ for source input with n words. Scoring the target language output using a language model within CKY-style decoding requires two histories per hypothesis, one on the left edge of each span and one on the right, due to the fact that the target side is not generated in left to right order, but rather built bottom-up from sub-spans. This leads to complex problems in efficient language model integration and requires state reduction techniques (Heafield et al., 2011; Heafield et al., 2013). The size of a Hiero SCFG grammar is typically larger than phrase-based models extracted from the same data creating challenges in rule extraction and decoding time especially for larger datasets (Sankaran et al., 2012).

In contrast, the LR-decoding algorithm could avoid these shortcomings such as faster time complexity, reduction in the grammar size and the simplified left-to-right language model scoring. It means LR decoding has the potential to replace CKY decoding for Hiero. Despite these attractive properties, we show that the original LR-Hiero decoding proposed by (Watanabe et al., 2006b) does not perform to the same level of the standard CKY Hiero with cube pruning (see Table 3). In addition, the current LR decoding algorithm does not obtain BLEU scores comparable to phrase-based or CKY-based Hiero models for different language pairs (see Table 4). In this paper we propose modifications to the LR decoding algorithm that addresses these limitations and provides, for the first time, a true alternative to the standard CKY Hiero algorithm that uses left-to-right decoding.

We introduce a new extended version of the LR decoding algorithm presented in (Watanabe et al., 2006b) which is demonstrably more efficient than the CKY Hiero algorithm. We measure the efficiency of the LR Hiero decoder in a way that is independent of the choice of system and programming language by measuring the number of language model queries. Although more efficient, the new LR decoding algorithm suffered from lower BLEU scores compared to CKY Hiero. Our analysis of left to right decoding showed that it has more potential for search errors due to early pruning of good hypotheses. This is unlike bottom-up decoding (CKY) which keeps best hypotheses for each span. To address this issue, we introduce two novel features into the Hiero SMT model that deal with reordering and distortion. Our experiments show that LR decoding with these features using prefix lexi-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1089–1099,
Seattle, Washington, USA, 18-21 October 2013.

calized target side rules equals the scores obtained by CKY decoding with prefix lexicalized target side rules and phrase-based translation system. It performs four times fewer language model queries on average, compare to CKY Hiero decoding with unrestricted Hiero rules: 6466.7 LM queries for CKY Hiero (with cube pruning) compared to 1500.45 LM queries in LR Hiero (with cube pruning). While translation quality suffers by only about 0.67 in BLEU score on average, across two different language pairs.

2 Left-to-Right Decoding for Hiero

Hierarchical phrase-based SMT (Chiang, 2005; Chiang, 2007) uses a synchronous context free grammar (SCFG), where the rules are of the form $X \rightarrow \langle \gamma, \alpha \rangle$, where X is a non-terminal, γ and α are strings of terminals and non-terminals.

Chiang (2007) places certain constraints on the extracted rules in order to simplify decoding. This includes limiting the maximum number of non-terminals (rule arity) to two and disallowing any rule with consecutive non-terminals on the foreign language side. It further limits the length of the initial phrase-pair as well as the number of terminals and non-terminals in the rule. For translating sentences longer than the maximum phrase-pair length, the decoder relies on additional glue rules $S \rightarrow \langle X, X \rangle$ and $S \rightarrow \langle SX, SX \rangle$ that allows monotone combination of phrases. The glue rules are used when no rules could match or the span length is larger than the maximum phrase-pair length.

2.1 Rule Extraction for LR Decoding

Left-to-right Hiero (Watanabe et al., 2006b) generates the target hypotheses left to right, but for synchronous context-free grammar (SCFG) as used in Hiero. The target-side rules are constrained to be prefix lexicalized. These constrained SCFG rules are defined as:

$$X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle \tag{1}$$

where γ is a mixed string of terminals and non-terminals. $\bar{b}$ is a terminal sequence prefixed to the possibly empty non-terminal sequence β. For the sake of simplicity, We refer to these type of rules as *GNF rules*[1] in this paper.

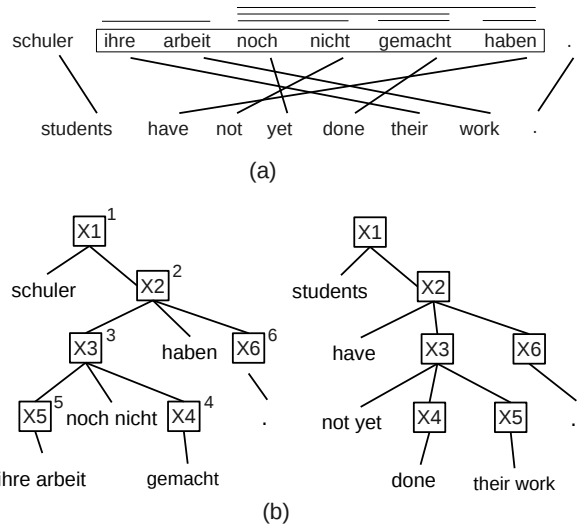


Figure 1: (a): A word-aligned German-English sentence pair. The bars above the source words indicate phrase-pairs having at least two words. (b): its corresponding left-to-right target derivation tree. Superscripts on the source non-terminals show the indices of the rules (see Fig 2) used in derivation.

Rule extraction is similar to Hiero, except any rules violating GNF form on the target side are excluded. Rule extraction considers each smaller source-target phrase pair within a larger phrase pair and replaces the spans with non-terminal X, yielding hierarchical rules. Figure 1(a) shows a word-aligned German-English sentence with a phrase pair $\langle ihre\ arbeit\ noch\ nicht\ gemacht\ haben,\ have\ not\ yet\ done\ their\ work \rangle$ that will lead to a SCFG rule. Given other smaller phrases (marked by bars above the source side), we extract a GNF rule[2]:

$$X \rightarrow \langle X_1\ noch\ nicht\ X_2\ haben, have\ not\ yet\ X_2\ X_1 \rangle \tag{2}$$

In order to avoid data sparsity and for better generalization, Watanabe et al. (2006b) adds four *glue* rules for each lexical rule $\langle \bar{f}, \bar{e} \rangle$ which are analogous to the glue rules defined in (Chiang, 2007) (see above) except that these glue rules for LR decoding

[1] Griebach Normal Form (GNF), although the synchronous grammar is not in this normal form, rather only the target side is prefix lexicalized as if it were in GNF form.

[2] LR-Hiero rule extraction excludes non-GNF rules such as $X \rightarrow \langle X_1\ noch\ nicht\ gemacht\ X_2, X_2\ not\ yet\ done\ X_1 \rangle$.

allow reordering as well.

$$\begin{array}{ll} X \rightarrow \langle \bar{f} X_1, \bar{e} X_1 \rangle & X \rightarrow \langle X_1 \bar{f} X_2, \bar{e} X_1 X_2 \rangle \\ X \rightarrow \langle X_1 \bar{f}, \bar{e} X_1 \rangle & X \rightarrow \langle X_1 \bar{f} X_2, \bar{e} X_2 X_1 \rangle \end{array} \quad (3)$$

It might appear that the restriction that target-side rules be GNF is a severe restriction on the coverage of possible hypotheses compared to the full set of rules permitted by the Hiero extraction heuristic. However there is some evidence in the literature that discontinuous spans on the source side in translation rules is a lot more useful than discontinuous spans in the target side (which is disallowed in the GNF). For instance, (Galley and Manning, 2010) do an extensive study of discontinuous spans on source and target side and show that source side discontinuous spans are very useful but removing discontinuous spans on the target side only lowers the BLEU score by 0.2 points (using the Joshua SMT system on Chinese-English). Removing discontinuous spans means that the target side rules have the form: uX, Xu, XuX, XXu, or uXX of which we disallow Xu, XuX, XXu. Zhang and Zong (2012) also conduct a study on discontinuous spans on source and target side of Hiero rules and conclude that source discontinuous spans are always more useful than discontinuities on the target side with experiments on four language pairs (zh-en, fr-en, de-en and es-en). As we shall also see in our experimental results (see Table 4) we can get close to the BLEU scores obtained using the full set of Hiero rules by using only target lexicalized rules in our LR decoder.

2.2 LR-Hiero Decoding

LR-Hiero decoding uses a top-down depth-first search, which strictly grows the hypotheses in target surface ordering. Search on the source side follows an Earley-style search (Earley, 1970), the dot jumps around on the source side of the rules based on the order of nonterminals on the target side. This search is integrated with beam search or cube pruning to efficiently find the k-best translations.

Several important details about the algorithm of LR-Hiero decoding are implicit and unexplained in (Watanabe et al., 2006b). In this section we describe the LR-Hiero decoding algorithm in more detail than the original description in (Watanabe et al., 2006b). We explain our own modified algorithm for LR decoding with cube pruning in Section 2.3.

Algorithm 1: LR-Hiero Decoding

Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
$\mathcal{F}$ = FutureCost($\mathbf{f}$) (Precompute future cost for spans)
for $i = 0, \dots, n$ **do**
  $S_i = \{\}$ (Create empty stacks)
$h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
Add h_0 to S_0 (Push initial hyp into first Stack)
for $i = 0, \dots, n-1$ **do**
  for each h in S_i **do**
    $[u, v]$ = pop(h_s) (Pop first uncovered span from list)
    R = GetSpanRules($[u, v]$) (Extract rules matching the entire span $[u, v]$)
    for $r \in R$ **do**
      h' = GrowHypothesis($h, r, [u, v], \mathcal{F}$) (New hypothesis)
      Add h' to S_l, where $l = |h'_{cov}|$ (Add new hyp to stack)
return $\arg\max(S_n)$

GrowHypothesis$(h, r, [u, v], \mathcal{F})$
  $h' = (h'_t = \emptyset, h'_s = h_s, h'_{cov} = \emptyset, h'_c = 0)$
  $r_X = \{X_j, X_k, \dots | j \lhd k \lhd \dots\}$ (Get NTs in surface order)
  for each X in reverse(r_X) **do**
    push($h'_s, span(X)$) (Push uncovered spans to LIFO list)
  h'_t = Concatenate(h_t, r_t)
  h'_{cov} = UpdateCoverage(h_{cov}, r_s)
  h'_c = ComputeCost($g(h'), \mathcal{F}_{\neg h'_{cov}}$)
  return h'

Algorithm 1 shows the pseudocode for LR decoding. Decoding the example in Figure 1(b) is explained using a walk-through shown in Figure 2. Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c): consisting of a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c. The initial hypothesis is a null string with just a sentence-initial marker $\langle s \rangle$ and the list h_s containing a span of the whole sentence, $[0, n]$. The hypotheses are stored in stacks $S_0, \dots, S_n$, where each stack corresponds to a coverage vector of same size, covering same number of source words (Koehn et al., 2003).

At the beginning of beam search the initial hy-

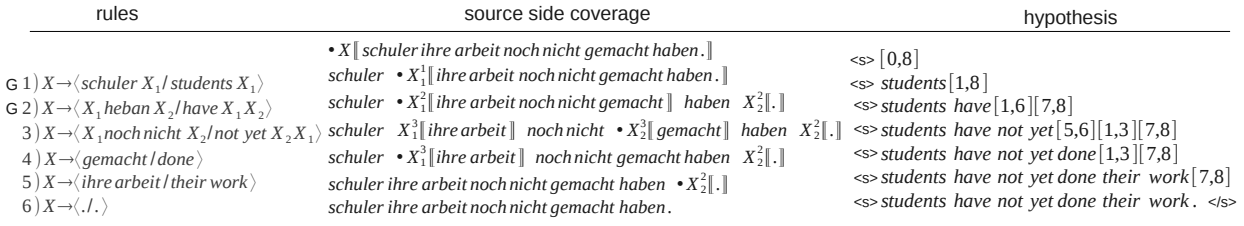


Figure 2: Illustration of the LR-Hiero decoding process in Figure 1. (a) Rules pane show the rules used in the derivation (glue rules are marked by G) (b) Decoder state using Earley dot notation (superscripts show rule#) (c) Hypotheses pane showing translation prefix and ordered list of yet-to-be-covered spans.

pothesis h_0 is added to the decoder stack S_0 (line 6 in Algoorithm 1). Hypotheses in each decoder stack are expanded iteratively, generating new hypotheses, which are added to the latter stacks corresponding to the number of source words covered. In each step it pops from the LIFO list h_s, the span $[u, v]$ of the next hypothesis h to be processed.

All rules that match the entire span $[u, v]$ are then obtained efficiently via pattern matching (Lopez, 2007). *GetSpanRules* addresses possible ambiguities in matched rules to the given span $[u, v]$. For example, given a rule r, with source side r_s : $\langle X_1 \; the \; X_2 \rangle$ and source phrase p : $\langle$*ok, the more the better*$\rangle$. There is ambiguity in matching r to p. *GetSpanRules* returns a distinct matched rule for each possible matching.

The *GrowHypothesis* routine creates a new candidate by expanding given hypothesis h using rule r and computes the complete hypothesis score including language model score. Since the target-side rules are in GNF, the translation prefix of the new hypothesis is obtained by simply concatenating the terminal prefixes of h and r in same order (line 20). *UpdateCoverage* updates source word coverage set using the source side of r. The h_s list is built by pushing the non-terminal spans of rule r in a reverse order (lines 17 and 18). The reverse ordering maintains the left-to-right generation of the target side.

In the walk-through in Figure 2, the derivation process starts by expanding the initial hypothesis h_0 (first item in the right pane of Fig 2) with the rule (rule #1 in left pane) to generate a new partial candidate having a terminal prefix of $\langle s \rangle$ *students* (second item in right pane). The second item in the middle pane shows the current position of the parser employing Earley's dot notation, indicating that the first word has already been translated. Now the decoder considers the second hypothesis and pops the span $[1, 8]$. It then matches the rule (#2) and pushes the spans $[1, 6]$ and $[7, 8]$ into the list h_s in the reverse order of their appearance in the target-side rule. At each step the new hypothesis is added to the decoder stack S_l depending on the number of covered words in the new hypothesis (line 13 in Algorithm 1).

For pruning we use an estimate of the future cost[3] of the spans uncovered by current hypothesis together with the hypothesis cost. The future cost is precomputed (line 2 Algorithm 1) in a way similar to the phrase-based models (Koehn et al., 2007) using only the terminal rules of the grammar. The ComputeCost method (line 22 in Algorithm 1) uses the usual log-linear model and scores a hypothesis based on its different feature scores $g(h')$ and the future cost of the *yet to be covered* spans ($\mathcal{F}_{\neg h'_{cov}}$). Time complexity of left to right Hiero decoding with beam search is $O(n^2 b)$ in practice where n is the length of source sentence and b is the size of beam (Huang and Mi, 2010).

2.3 LR-Hiero Decoding with Cube Pruning

The Algorithm 1 presented earlier does an exhaustive search as it generates all possible partial translations for a given stack that are reachable from the hypotheses in previous stacks. However only a few of these hypotheses are retained, while majority of them are pruned away. The cube pruning technique (Chiang, 2007) avoids the wasteful generation of poor hypotheses that are likely to be pruned away by efficiently restricting the generation to only high scoring partial translations.

We modify the cube pruning for LR-decoding that takes into account the next uncovered span to

[3] Watanabe et al. (2006b) also use a similar future cost, even though it is not discussed in the paper (p.c.).

Algorithm 2: LR-Hiero Decoding with Cube Pruning

1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
2: $\mathcal{F}$ = FutureCost($\mathbf{f}$) (Precompute future cost for spans)
3: $S_0 = \{\}$ (Create empty initial stack)
4: $h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
5: Add h_0 to S_0 (Push initial hyp into first Stack)
6: **for** $i = 1, \dots, n$ **do**
7: $cubeList = \{\}$ (MRL is max rule length)
8: **for** $p = max(i - \text{MRL}, 0), \dots, i - 1$ **do**
9: $\{G\}$ = Grouped(S_p) (Group based on the first uncovered span)
10: **for** $g \in \{G\}$ **do**
11: $[u, v]$ = g_{span}
12: R = GetSpanRules($[u, v]$)
13: **for** $R_s \in R$ **do**
14: $cube = [g_{hyps}, R_s]$
15: Add $cube$ to $cubeList$
16: S_i = Merge($cubeList, \mathcal{F}$) (Create stack S_i and add new hypotheses to it, see Figure 3)
17: **return** $\arg\max(S_n)$

18: **Merge**($CubeList, \mathcal{F}$)
19: $heapQ = \{\}$
20: **for** each (H, R) in $cubeList$ **do**
21: $[u, v]$ = span of rule R
22: h' = GrowHypothesis($h_1, r_1, [u, v], \mathcal{F}$) (from Algorithm 1)
23: push($heapQ, (h'_c, h', [H, R])$
24: $hypList = \{\}$
25: **while** $|heapQ| > 0$ and $|hypList| < K$ **do**
26: $(h'_c, h', [H, R])$ = pop($heapQ$)
27: push($heapQ, GetNeighbours([H, R])$
28: Add h' to $hypList$
29: **return** $hypList$

be translated indicated by the Earley's dot notation. The Algorithm 2 shows the pseudocode for LR-decoding using cube pruning. The structure of stacks and hypotheses and computing the future cost is similar to Algorithm 1 (lines 1-5). To fill stack S_i, it iterates over previous stacks (line 8 in Algorithm 2) [4]. All hypotheses in each stack S_p (covering p words on the source-side) are first partitioned into a set of groups, $\{G\}$, based on their first uncovered span (line 9) [5]. Each group g is a 2-tuple (g_{span}, g_{hyps}), where g_{hyps} is a list of hypotheses which share the same first uncovered span g_{span}. Rules matching the span g_{span} are obtained from routine *GetSpanRules*, which are then grouped based on unique source side rules (i.e. each R_s contains rules that share the same source side s but have different target sides). Each g_{hyps} and possible R_s[6] create a cube which is added to $cubeList$.

In LR-Hiero, each hypothesis is developed with only one uncovered span, therefore each cube always has just two dimensions: (1) hypotheses with the same number of covered words and similar first uncovered span, (2) rules sharing the same source side. In Figure 3(a), each group of hypotheses, g_{hyps}, is shown in a green box (in stacks), and each rectangle on the top is a cube. Figure 3 is using the example in Figure 2.

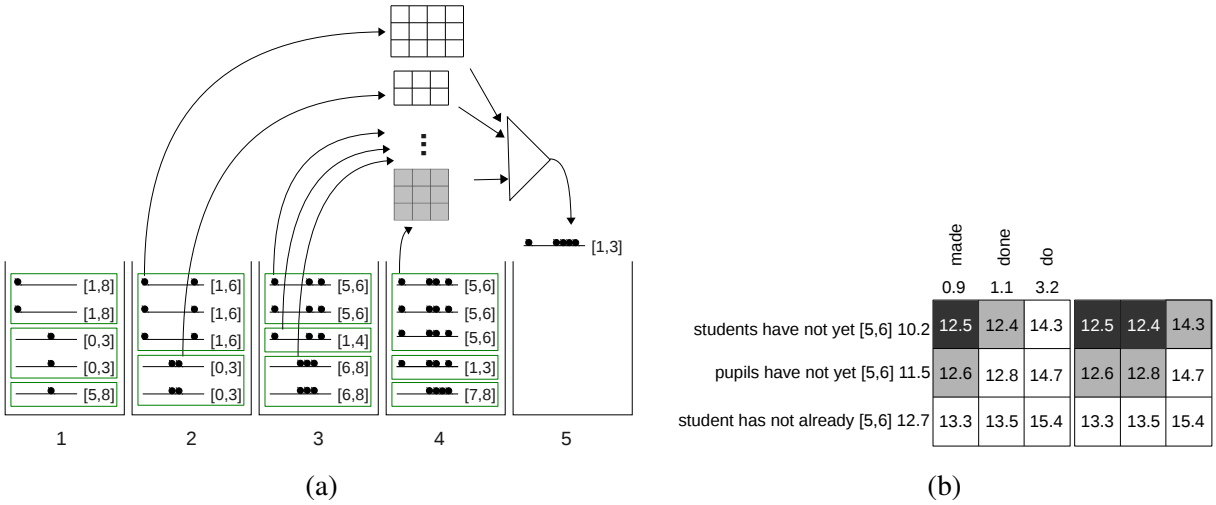
The *Merge* routine is the core function of cube pruning which generates the best hypotheses from all cubes (Chiang, 2007). For each possible cube, (H, R), the best hypothesis is generated by calling $GrowHypothesis(h_1, r_1, \textit{span}, \mathcal{F})$ where h_1 and r_1 are the best hypothesis and rule in H and R respectively (line 22). Figure 3 (b) shows a more detailed view of a cube (shaded cube in Figure 3(a)). Rows are hypotheses and columns are rules which are sorted based on their scores.

The first best hypotheses, h', along with their score, h'_c and corresponding cube, (H, R) are placed in a priority queue, $heapQ$ (triangle in Figure 3). Iteratively the best hypothesis is popped from the queue (line 26) and its neighbours in the cube are added to the priority queue (using $GetNeighbours([H, Q])$). It continues to generate all K best hypotheses. Using cube pruning technique, each stack is filled with K best hypotheses without generating all possible hypotheses in each cube.

[4] As the length of rules are limited (at most MRL), we can ignore stacks with index less than $i - \text{MRL}$

[5] The beam search decoder in Phrase-based system (Huang and Chiang, 2007; Koehn et al., 2007; Sankaran et al., 2010) groups the hypotheses in a given stack based on their coverage vector. But this idea does not work in LRHiero decoding in which the expansion of each hypothesis is restricted to its first uncovered span. We have also tried another way of grouping hypotheses: group by all uncovered spans, h_s. Our experiments did not show any significant difference between the final results (BLEU score), therefore we decided to stick to the simpler idea: using first uncovered span for grouping.

[6] Note that, just rules whose number of terminals in their source side is equal to $i - p$ can be used.



(a) (b)

Figure 3: Example of generating hypotheses in cube pruning using Figure 2: (a) Hypotheses in previous stacks are grouped based on their first uncovered span, and build cubes (grids on top). Cubes are in different sizes because of different number of rules and group sizes. Cubes are fed to a priority queue (triangle) and new hypotheses are iteratively popped from the queue and added to the current stack, S_5. (b) Generating hypotheses from a cube. The top side of the grid denotes the target side of rules sharing the same source side (R_s) along with their scores. Left side of the grid shows the hypotheses in a same group, their first uncovered span and their scores. Hypothesis generated from row 1 and column 1 is added to the queue at first. Once it is popped from the queue, its neighbours (in the grid) are subsequently added to the queue.

Figure 3 (b) shows the derivation of the two best hypotheses from the cube. The best hypothesis of this cube which is likely created from the best hypothesis and rule (left top most entry) is popped at first step. Then, *GetNeighbours* calls *GrowHypothesis* to generate next potential best hypotheses of this cube (neighbours of the popped entry which are shaded in Figure 3(b)). These hypotheses are added to the priority queue. In the next iteration, the best hypothesis is popped from all candidates in the queue and algorithm continues.

3 Features

We use the following standard SMT features for the log-linear model of LR-Hiero: relative-frequency translation probabilities $p(f|e)$ and $p(e|f)$, lexical translation probabilities $p_l(f|e)$ and $p_l(e|f)$, a language model probability, word count and phrase count. In addition we also use the glue rule count and the two reordering penalty features employed by Watanabe et al. (2006b; 2006a). These features compute the *height* and *width* (span size of the entire subtree) of all subtrees which are *backtraced* in the derivation of a hypothesis. A non-terminal X_i is pushed into the LIFO list of a partial hypothesis; it's *backtrace* refers to the set of NTs that must be popped before X_i.

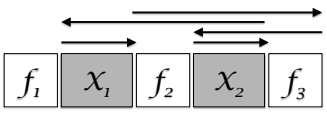
In Figure 1(b), X_2 has two subtrees X_3 and X_6, where X_3 should be processed before X_6. The subtree rooted at X_3 in Figure 1(b) has a height of 2 and span $[1, 6]$ having a width of 5. Similarly, X_4 should be backtraced before X_5 and has height and width of 1. Backtracing applies only for rules having at least two non-terminals. Thus the total height and width penalty for this derivation are 3 and 6 respectively.

However, the height and width features do not distinguish between a rule that reorders the non-terminals in source and target from one that preserves the ordering. Rules #2 and #3 in Figure 2 are treated equally although they have different orderings. The decoder is thus agnostic to this difference and would not be able to exploit this effectively to control reordering and instead would rely on the partial LM score. This issue is exacerbated for glue rules, where the decoder has to choose from different possibilities without any way to favour one over the others. Instead of the rule #2, the decoder could use its reordered version $\langle X_1\ haben\ X_2, have\ X_2\ X_1 \rangle$ leading to a poor translation.

The features we introduce can be used to learn if the model should favour monotone translations at the cost of re-orderings or vice versa and hence can easily adapt to different language pairs. Further, our experiments (see Section 4) suggest that the features h and w are not sufficient by themselves to model reordering for language pairs exhibiting very different syntactic structure.

3.1 Distortion Features

Our distortion features are inspired by their namesake in phrase-based system, with some modifications to adapt the idea for the discontiguous phrases in LR-Hiero grammar.



(a)

$r : \langle f_1 X_1 f_2 X_2 f_3, tX_2X_1 \rangle \quad I = [\vdash, f_1, f_2, f_3, X_2, X_1, \dashv]$

$r : \langle X_1 \textit{ noch nicht } X_2 / \textit{not yet } X_2 X_1 \rangle$

$_1\textit{ihre}_2 \textit{arbeit}_3 \textit{noch}_4 \textit{nicht}_5 \textit{gemacht}_6$

$I = [(1,1),(3,5),(5,6),(1,3),(6,6)]$

(b)

Figure 4: (a) Distortion feature computation using a rule r. (b) Example of distortion computation for applying r_3 on phrase $\langle$*ihre arbeit noch nicht gemacht haben*$\rangle$. subscripts between words show the indices which are used to build I. Distortion would be: $d = 2 + 0 + 5 + 3$.

Consider a rule $r = \langle \gamma, \bar{b} \ \beta \rangle$, with the source term γ being a mixed string of terminals and non-terminals. Representing the non-terminal spans and each sequence of terminals in γ as distinct *items*, our distortion feature counts the total length of *jumps* between the items during Earley parsing.

Figure 4 (a) explains the computation of our distortion feature for an example rule r. Let $I = [I_0, \ldots, I_k]$ be the *items* denoting the terminal sequences and non-terminal spans with I_0 and I_k being dummy items ($\vdash$ and $\dashv$ in Fig) marking the left and right indices of the rule r in input sentence f. Other items are arranged by their realization order on the target-side with the terminal sequences preceding non-terminal spans. The items for the example rule are shown in Figure 4 (a). The distortion feature is computed as follows:

$$d(r) = \sum_{j=1}^{k} |I_j^{\mathcal{L}} - I_{j-1}^{\mathcal{R}}| \tag{4}$$

where superscripts refer to position of left ($\mathcal{L}$) and right ($\mathcal{R}$) edge of each item in the source sentence f. These are then aggregated across the rules of a derivation D as: $d = \sum_{r \in D} d(r)$. For each item I_j, we count the jump from the end of previous item to the beginning of the current. In Figure 4 (a) the jumps are indicated by the arrows above the rule. Figure 4 (b) shows an example of distortion computation for r_3 and phrase $\langle$*ihre arbeit noch nicht gemacht haben*$\rangle$ from Figure 2.

Since the glue rules are likely to be used in the top levels (possibly with large distortion) of the derivation, we would want the decoder to learn the distortion for regular and glue rules separately. We thus use two distortion features for the two rule types and we call them d_p and d_g.

These features do not *directly* model the source-target reordering, but only capture the source-side jumps. Furthermore they apply for both monotone and reordering rules. We now introduce a new feature for exclusively modelling the reordering.

3.2 Reordering Feature

This feature simply counts the number of *reordering rules*, where the non-terminals in source and target sides are reordered. Thus $r_{\langle\rangle} = \text{rule}(D, \langle\rangle)$, where $\text{rule}(D, \langle\rangle)$ is the number of reordering rules in D. Similar to width and height, this feature is applied for rule having at least two non-terminals. This feature is applied to regular and glue rules.

4 Experiments

We conduct different types of experiments to evaluate LR-Hiero decoding developed by cube pruning and integrating new features into LR-Hiero system for two language pairs: German-English (de-en) and Czech-English (cs-en).Table 1 shows the dataset details.

4.1 System Setup

In our experiments we use four baselines as well as our implementation of LR-Hiero (written in Python):

	Corpus	Train/Dev/Test
cs-en	Europarl(v7), CzEng(v0.9); News commentary	7.95M/3000/3003
de-en	Europarl(v7); News commentary	1.5M/2000/2000

Table 1: Corpus statistics in number of sentences

Model	cs-en	de-en
Phrase-based	233.0	77.2
Hiero	1,961.6	858.5
LR-Hiero	230.5	101.3

Table 2: Model sizes (millions of rules). We do not count glue rules for LR-Hiero which are created at runtime as needed.

- **Hiero:** we used Kriya, our open-source implementation of Hiero in Python, which performs comparably to other open-source Hiero systems (Sankaran et al., 2012). Kriya can obtain statistically significantly equal BLEU scores when compared with Moses (Koehn et al., 2007) for several language pairs (Razmara et al., 2012; Callison-Burch et al., 2012).

- **Hiero-GNF:** where we use Hiero decoder with the restricted LR-Hiero grammar (GNF rules).

- **LR-Hiero:** our implementation of LR-Hiero (Watanabe et al., 2006b) in Python.

- **phrase-based:** Moses (Koehn et al., 2007)

- **LR-Hiero+CP:** LR-Hiero decoding with cube pruning.

We use a 5-gram LM trained on the Gigaword corpus and use KenLM (Heafield, 2011) for LM scoring during decoding. We tune weights by minimizing BLEU loss on the dev set through MERT (Och, 2003) and report BLEU scores on the test set. We use comparable pop limits in each of the decoders: 1000 for Moses and LR-Hiero and 500 with cube pruning for CKY Hiero and LR-Hiero+CP. Other extraction and decoder settings such as maximum phrase length, etc. were identical across settings so that the results are comparable.

Table 2 shows how the LR-Hiero grammar is much smaller than CKY-based Hiero.

Model	cs-en #queries / time(ms)	de-en #queries / time(ms)
Hiero	5,679.7 / 16.12	7,231.62 / 20.33
Hiero-GNF	4,952.5 / 14.71	5,858.74 / 18.23
LR-Hiero (1000)	46,333.21 / 163.6	83,518.63 / 328.11
LR-Hiero (500)	24,141.03 / 97.61	42,783.12 / 192.23
LR-Hiero+CP	**1,303.2 / 4.2**	**1,697.7 / 5.67**

Table 3: Comparing average number and time of language model queries.

4.2 Time Efficiency Comparison

To evaluate the performance of LR-Hiero decoding with cube pruning (LR-Hiero+CP), we compare it with three baselines: (i) CKY Hiero, (ii) CKY Hiero-GNF, and (iii) LR-Hiero (without cube pruning) with two different beam size 500 and 1000. When it comes to instrument timing results, there are lots of system level details that we wish to abstract away from, and focus only on the number of "edges" processed by the decoder. In comparison of parsing algorithms, the common practice is to measure the number of edges processed by different algorithms for the same reason (Moore and Dowding, 1991). By analogy to parsing algorithm comparisons, we compare the different decoding algorithms with respect to the number of calls made to the language model (LM) since that directly corresponds to the number of hypotheses considered by the decoder. A decoder is more time efficient if it can consider fewer translation hypotheses while maintaining the same BLEU score. All of the baselines use the same wrapper to query the language model, and we have instrumented the wrapper to count the statistics we need and thus we can say this is a fair comparison. For this experiment we use a sample set of 50 sentences taken from the test sets.

Table 3 shows the results in terms of average number of language model queries and times in milliseconds.

4.3 Reordering Features

To evaluate the new reordering features proposed to LR-Hiero (Section 3.2), LR-Hiero+CP with new features is compared to all baselines. Table 4 shows the BLEU scores of different models in two language pairs. The baseline (Watanabe et al., 2006b) model uses all the features mentioned therein but is

Model	cs-en	de-en
Phrase-based	20.32	24.71
CKY Hiero	20.64	25.52
CKY Hiero-GNF	20.04	24.84
LR-Hiero	18.30	23.47
LR-Hiero + reordering feats	20.20	24.90
LR-Hiero + CP + reordering feats	20.15	24.83
CKY Hiero-GNF + reordering feats	20.52	25.09
CKY Hiero + reordering feats	**20.77**	**25.72**

Table 4: BLEU scores. The rows are grouped such that each group use the same model. The last row in part 2 of table shows LR-Hiero+CP using our new features in addition to the baseline Watanabe features (line *LR-Hiero baseline*). The last part shows CKY Hiero using new reordering features. The reordering features used are d_p, d_g and $r_{\langle\rangle}$. LR-Hiero+CP has a beam size of 500 while LR-Hiero has a beam size of 1000, c.f. with the LM calls shown in Table 3.

worse than both phrase-based and CKY-Hiero baselines by up to 2.3 BLEU points.

All the reported results are obtained from a single optimizer run. However we observed insignificant changes in different tuning runs in our experiments. We find a gain of about 1 BLEU point when we add a single distortion feature d and a further gain of 0.3 BLEU (not shown due to lack of space) when we split the distortion feature for the two rule types (d_p and d_g). The last line in part two of Table 4 shows a consistent gain of 1.6 BLEU over the LR-Hiero baseline for both language pairs. It shows that LR-Hiero maintains the BLEU scores obtained by "phrase-based" and "CKY Hiero-GNF".

We performed statistical significance tests using two different tools: Moses bootstrap resampling and MultEval (Clark et al., 2011). The difference between "LR-Hiero+CP+reordering feat" and three baselines: "phrase-based", "CKY Hiero-GNF", "LR-Hiero+reordering feat" are not statistically significant even for p-value of 0.1 for both tools.

To investigate the impact of proposed reordering features with other decoder or models. We add these features to both Hiero and Hiero-GNF[7]. The last part of Table 4 shows the performance CKY decoder with different models (full Hiero and GNF) with the new reordering features in terms of BLEU score. The results show that these features are helpful in both models. Although, they do not make a big difference in Hiero with full model, they can alleviate the lack of non-GNF rules in Hiero-GNF.

Nguyen and Vogel (2013) integrate traditional phrase-based features: distortion and lexicalized reordering into Hiero as well. They show that such features can be useful to boost the translation quality of CKY Hiero with the full rule set. Nguyen and Vogel (2013) compute the distortion feature in a different way, only applicable to CKY. The distortion for each cell is computed after the translation for non-terminal sub-spans is complete. In LR-decoding, we compute distortion for rules even though we are yet to translate some of the sub-spans. Thus our approach computes the distortion incrementally for the untranslated sub-spans which are later added. Unlike (Nguyen and Vogel, 2013), our distortion feature can be applied to both LR and CKY-decoding (Table 4). We have also introduced another reordering feature (Section 3.2) not proposed previously.

5 Conclusion and Future Work

We provided a detailed description of left-to-right Hiero decoding, many details of which were only implicit in (Watanabe et al., 2006b). We presented an augmented LR decoding algorithm that builds on the original algorithm in (Watanabe et al., 2006b) but unlike that algorithm, using experiments over multiple language pairs we showed two new results: (i) Our LR decoding algorithm provides demonstrably more efficient decoding than CKY Hiero and the original LR decoding algorithm in (Watanabe et al., 2006b). And, (ii) by introducing new distortion and reordering features for LR decoding we show that it maintains the BLEU scores obtained by phrase-based and CKY Hiero-GNF.

CKY Hiero uses standard Hiero-style translation rules capturing better reordering model than prefix lexicalized target-side translation rules used in LR-Hiero. Our LR-decoding algorithm is 4 times faster in terms of LM calls while translation quality suffers by about 0.67 in BLEU score on average.

Unlike Watanabe et al. (2006b), our new features can easily adapt to the reordering requirements of different language pairs. We also introduce the use

[7]Feature $r_{\langle\rangle}$ is defined for SCFG rules and cannot be adopted to phrase-based translation systems; and Moses uses distortion feature therefore we omit Moses from this experiment.

of future cost in decoding algorithm which is an essential part in decoding. We have shown in this paper that left-to-right (LR) decoding can be considered as a potential faster alternative to CKY decoding for Hiero-style machine translation systems.

In future work, we plan to apply lexicalized reordering models to LR-Hiero. It has been shown to be useful for Hiero in some languages therefore it is promising to improve translation quality in LR-Hiero which suffers from lack of modeling power of non-GNF target side rules. We also plan to extend the glue rules in LR-Hiero to provide a better reordering model. We believe such an extension would be very effective in reducing search errors and capturing better reordering models in language pairs involving complex reordering requirements like Chinese-English.

Acknowledgments

This research was partially supported by an NSERC, Canada (RGPIN: 264905) grant and a Google Faculty Award to the third author. The authors wish to thank Taro Watanabe and Marzieh Razavi for their valuable discussions and suggestions, and the anonymous reviewers for their helpful comments.

References

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *In ACL*, pages 263–270.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February.

Michel Galley and Christopher D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June. Association for Computational Linguistics.

Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. 2011. Left language model state for syntactic machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 183–190, San Francisco, California, USA, 12.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2013. Grouping language model boundary words to speed K-Best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, USA, 6.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *In Proc. of the Sixth Workshop on Statistical Machine Translation*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *In ACL 07*.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *EMNLP-CoNLL*, pages 976–985.

Robert C. Moore and John Dowding. 1991. Efficient bottom-up parsing. In *HLT*. Morgan Kaufmann.

Thuylinh Nguyen and Stephan Vogel. 2013. Integrating phrase-based reordering features into chart-based decoder for machine translation. In *Proc. of ACL*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Majid Razmara, Baskaran Sankaran, Ann Clifton, and Anoop Sarkar. 2012. Kriya - the sfu system for translation task at wmt-12. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 356–361, Stroudsburg, PA, USA. Association for Computational Linguistics.

Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar. 2010. Incremental decoding for phrase-based statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 216–223, Stroudsburg, PA, USA. Association for Computational Linguistics.

Baskaran Sankaran, Majid Razmara, and Anoop Sarkar. 2012. Kriya - an end-to-end hierarchical phrase-based mt system. *The Prague Bulletin of Mathematical Linguistics (PBML)*, (97):83–98, apr.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2006a. NTT statistical machine translation for iwslt 2006. In *Proceedings of IWSLT 2006*, pages 95–102.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006b. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL*.

Jiajun Zhang and Chenqqing Zong. 2012. A Comparative Study on Discontinuous Phrase Translation. In *NLPCC 2012*, pages 164–175.

A Systematic Exploration of Diversity in Machine Translation

Kevin Gimpel[*] **Dhruv Batra**[†] **Chris Dyer**[‡] **Gregory Shakhnarovich**[*]
[*]Toyota Technological Institute at Chicago, Chicago, IL 60637, USA
[†]Virginia Tech, Blacksburg, VA 24061, USA
[‡]Carnegie Mellon University, Pittsburgh, PA 15213, USA
Corresponding author: kgimpel@ttic.edu

Abstract

This paper addresses the problem of producing a diverse set of plausible translations. We present a simple procedure that can be used with any statistical machine translation (MT) system. We explore three ways of using diverse translations: (1) system combination, (2) discriminative reranking with rich features, and (3) a novel post-editing scenario in which multiple translations are presented to users. We find that diversity can improve performance on these tasks, especially for sentences that are difficult for MT.

1 Introduction

From the perspective of user interaction, the ideal machine translator is an agent that reads documents in one language and produces accurate, high quality translations in another. This interaction ideal has been implicit in machine translation (MT) research since the field's inception. It is the way we interact with commercial MT services (such as Google Translate and Microsoft Translator), and the way MT systems are evaluated (Bojar et al., 2013). Unfortunately, when a real, imperfect MT system makes an error, the user is left trying to guess what the original sentence means.

Multiple Hypotheses. In contrast, when we look at the way other *computer systems* consume output from MT systems (or similarly unreliable tools), we see a different pattern. In a pipeline setting it is commonplace to propagate not just a *single*-best output but the M-best hypotheses (Venugopal et al., 2008). Multiple solutions are also used for reranking (Collins, 2000; Shen and Joshi, 2003; Collins and Koo, 2005; Charniak and Johnson, 2005), tuning (Och, 2003), minimum Bayes risk decoding (Kumar and Byrne, 2004), and system combination (Rosti et al., 2007). When dealing with error-prone systems, knowing about alternatives has benefits over relying on only a single output (Finkel et al., 2006; Dyer, 2010).

Need for Diversity. Unfortunately, M-best lists are a poor surrogate for structured output spaces (Finkel et al., 2006; Huang, 2008). In MT, for example, many translations on M-best lists are extremely similar, often differing only by a single punctuation mark or minor morphological variation. Recent work has explored reasoning about sets using packed representations such as lattices and hypergraphs (Macherey et al., 2008; Tromble et al., 2008; Kumar et al., 2009), or sampling translations proportional to their probability (Chatterjee and Cancedda, 2010). We argue that the implicit goal behind these techniques is to better explore the output space by introducing diversity into the surrogate set.

Overview and Contributions. In this work, we elevate diversity to a first-class status and directly address the problem of generating a set of **diverse, plausible translations**. We use the recently proposed technique of Batra et al. (2012), which produces diverse M-best solutions from a probabilistic model using a generic **dissimilarity** function $\Delta(\cdot, \cdot)$ that specifies how two solutions differ. Our first contribution is a family of dissimilarity functions for MT that admit simple algorithms for generating diverse translations. Other contributions are empirical: we show that diverse translations can lead to improvements for system combination and discriminative reranking. We also perform a novel human

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1100–1111,
Seattle, Washington, USA, 18-21 October 2013.

post-editing evaluation in order to measure whether diverse translations can help users make sense of noisy MT output. We find that diverse translations can help post-editors produce better outputs for sentences that are the most difficult for MT. While we focus on machine translation in this paper, we note that our approach is applicable to other structure prediction problems in NLP.

2 Preliminaries and Notation

Let $\mathcal{X}$ denote the set of all strings in a source language. For an $x \in \mathcal{X}$, let $\mathcal{Y}_x$ denote the set of its possible translations $\mathbf{y}$ in the target language. MT models typically include a latent variable that captures the derivational structure of the translation process. Regardless of its specific form, we refer to this variable as a **derivation** $\mathbf{h} \in \mathcal{H}_x$, where $\mathcal{H}_x$ is the set of possible values of $\mathbf{h}$ for x. Derivations are coupled with translations and we define $\mathcal{T}_x \subseteq \mathcal{Y}_x \times \mathcal{H}_x$ as the set of possible $\langle \mathbf{y}, \mathbf{h} \rangle$ pairs for x.

We use a linear model with a parameter vector $\mathbf{w}$ and a vector $\boldsymbol{\phi}(x, \mathbf{y}, \mathbf{h})$ of feature functions on x, $\mathbf{y}$, and $\mathbf{h}$ (Och and Ney, 2002). The translation of x is selected using a simple decision rule:

$$\langle \hat{\mathbf{y}}, \hat{\mathbf{h}} \rangle = \underset{\langle \mathbf{y}, \mathbf{h} \rangle \in \mathcal{T}_x}{\operatorname{argmax}} \; \mathbf{w}^\intercal \boldsymbol{\phi}(x, \mathbf{y}, \mathbf{h}) \qquad (1)$$

where we also maximize over the latent variable $\mathbf{h}$ for efficiency. Translation models differ in the form of $\mathcal{T}_x$ and the choice of the feature functions $\boldsymbol{\phi}$. In this paper we focus on phrase-based (Koehn et al., 2003) and hierarchical phrase-based (Chiang, 2007) models, which include several bilingual and monolingual features, including n-gram language models.

3 Diversity in Machine Translation

We now address the task of producing a set of diverse high-scoring translations.

3.1 Generating Diverse Translations

We use a recently proposed technique (Batra et al., 2012) that constructs diverse lists via a greedy iterative procedure as follows. Let $\mathbf{y}^1$ be the model-best translation (Eq. 1). On the m-th iteration, the m-th best (diverse) translation is obtained as $\langle \mathbf{y}^m, \mathbf{h}^m \rangle =$

$$\underset{\langle \mathbf{y}, \mathbf{h} \rangle \in \mathcal{T}_x}{\operatorname{argmax}} \; \mathbf{w}^\intercal \boldsymbol{\phi}(x, \mathbf{y}, \mathbf{h}) + \sum_{j=1}^{m-1} \lambda_j \Delta(\mathbf{y}^j, \mathbf{y}) \qquad (2)$$

where Δ is a dissimilarity function and λ_j is the weight placed on dissimilarity to previous translation j relative to the model score. Intuitively, we seek a translation that is highly-scoring under the model while being different (as measured by Δ) from all previous translations. The λ parameters determine the trade-off between model score and diversity. We refer to Eq. (2) as **dissimilarity-augmented** decoding.

The objective in Eq. (2) is a Lagrangian relaxation for an intractable constrained objective specifying a minimum dissimilarity Δ_{min} between translations in the list, i.e., $\Delta(\mathbf{y}^j, \mathbf{y}) \geq \Delta_{min}$ (Batra et al., 2012). Instead of setting the dissimilarity threshold Δ_{min}, we set the weights λ_j. While the formulation allows for a different λ_j for each previous solution j, we simply use a single $\lambda = \lambda_j$ for all j. This was also done in the experiments in (Batra et al., 2012).

Note that if the dissimilarity function factors across the parts of the output variables $\langle \mathbf{y}, \mathbf{h} \rangle$ in the same way as the features $\boldsymbol{\phi}$, then *the same decoding algorithm* can be used as for Eq. (1). We discuss design choices for Δ next.

3.2 Dissimilarity Functions for MT

When designing a dissimilarity function $\Delta(\cdot, \cdot)$ for MT, we want to consider variation both in individual word choice and longer-range sentence structure. We also want a function that can be easily incorporated into extant statistical MT systems. We propose a dissimilarity function that simply counts the number of times any n-gram is present in both translations, then negates. Letting $q = n - 1$:

$$\Delta_n(\mathbf{y}, \mathbf{y}') = -\sum_{i=1}^{|\mathbf{y}|-q} \sum_{j=1}^{|\mathbf{y}'|-q} [\![\mathbf{y}_{i:i+q} = \mathbf{y}'_{j:j+q}]\!] \qquad (3)$$

where $[\![\cdot]\!]$ is the Iverson bracket (1 if input condition is true, 0 otherwise) and $\mathbf{y}_{i:j}$ is the subsequence of $\mathbf{y}$ from word i to word j (inclusive).

Importantly, Eq. (2) can be solved with no change to the decoding algorithm. The dissimilarity terms can simply be incorporated as an additional language model in ARPA format that sets the log-probability to the negated count for each n-gram in previous diverse translations, and sets to zero all other n-grams' log-probabilities and back-off weights.

The advantage of this dissimilarity function is its simplicity. It can be easily used with any translation system that uses n-gram language models without any change to the decoder. Indeed, we use both phrase-based and hierarchical phrase-based models in our experiments below.

4 Related Work

MT researchers have recently started to consider diversity in the context of system combination (Macherey and Och, 2007). Most closely-related is work by Devlin and Matsoukas (2012), who proposed a way to generate diverse translations by varying particular "traits," such as translation length, number of rules applied, etc. Their approach can be viewed as solving Eq. (2) with a richer dissimilarity function that requires a special-purpose decoding algorithm. We chose our n-gram dissimilarity function due to its simplicity and applicability to most MT systems without requiring any change to decoders.

Among other work, Xiao et al. (2013) used bagging and boosting to get diverse system outputs for system combination and Cer et al. (2013) used multiple identical systems trained jointly with an objective function that encourages the systems to generate complementary translations.

There is also similarity between our approach and minimum Bayes risk decoding (Kumar and Byrne, 2004), variational decoding (Li et al., 2009), and other "consensus" decoding algorithms (DeNero et al., 2009). These all seek a single translation that is most similar on average to the model's preferred translations. In this way, they try to capture the model's range of beliefs in a single translation. We instead seek a *set* of translations that, when considered as a whole, similarly express the full range of the model's beliefs about plausible translations for the input.

Also related is work on determinantal point processes (DPPs; Kulesza and Taskar, 2010), an elegant probabilistic model over sets of items that naturally prefers diverse sets. DPPs have been applied to summarization (Kulesza and Taskar, 2011) and discovery of topical threads in document collections (Gillenwater et al., 2012). Unfortunately, in the *structured* setting, DPPs make severely restrictive assumptions on the scoring function, while our framework does not.

5 Experimental Setup

We now embark on an extensive empirical evaluation of the framework presented above. We begin by analyzing our diverse sets of translations, showing how they differ from standard M-best lists (Section 6), followed by three tasks that illustrate how diversity can be exploited to improve translation quality: system combination (Section 7), discriminative reranking (Section 8), and a novel human post-editing task (Section 9). In the remainder of this section, we describe details of our experimental setup.

5.1 Language Pairs and Datasets

We use three language pairs: Arabic-to-English (AR→EN), Chinese-to-English (ZH→EN), and German-to-English (DE→EN). For AR→EN and DE→EN, we used a phrase-based model (Koehn et al., 2003) and for ZH→EN we used a hierarchical phrase-based model (Chiang, 2007).

Each language pair has two tuning and one test set: TUNE1 is used for tuning the baseline systems with minimum error rate training (MERT; Och, 2003), TUNE2 is used for training system combiners and rerankers, and TEST is used for evaluation. There are four references for AR→EN and ZH→EN and one for DE→EN.

For AR→EN, we used data provided by the LDC for the NIST evaluations, which includes 3.3M sentences of UN data and 982K sentences from other (mostly news) sources. Arabic text was preprocessed using an HMM segmenter that splits attached prepositional phrases, personal pronouns, and the future marker (Lee et al., 2003). The common stylistic sentence-initial *w+* (*and*) clitic was removed. The resulting corpus contained 130M Arabic tokens and 130M English tokens. We used the NIST MT06 test set as TUNE1, a 764-sentence subset of MT05 as TUNE2, and MT08 as TEST.

For ZH→EN, we used 303k sentence pairs from the FBIS corpus (LDC2003E14). We segmented the Chinese data using the Stanford Chinese segmenter (Chang et al., 2008) in "CTB" mode, giving us 7.9M Chinese tokens and 9.4M English tokens. We used the NIST MT02 test set as TUNE1, MT05

as TUNE2, and MT03 as TEST.

For DE→EN, we used data released for the WMT2011 shared task (Callison-Burch et al., 2011). German compound words were split using a CRF segmenter (Dyer, 2009). We used the WMT2010 test set as TUNE1, the 2009 test set as TUNE2, and the 2011 test set as TEST.

5.2 Baseline Systems

We used the Moses MT toolkit (Koehn et al., 2007; Hoang et al., 2009) with default settings and features for both phrase-based and hierarchical systems. Word alignment was done using GIZA++ (Och and Ney, 2003) in both directions, with the `grow-diag-final-and` heuristic used to symmetrize the alignments and a max phrase length of 7 used for phrase extraction.

Language models used the target side of the parallel corpus in each case augmented with 24.8M lines (601M tokens) of randomly-selected sentences from the Gigaword v4 corpus (excluding the NY Times and LA Times). We used 5-gram models, estimated using the SRI Language Modeling toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). The minimum count cut-off for unigrams, bigrams, and trigrams was 1 and the cut-off for 4-grams and 5-grams was 3. Language model inference used KenLM (Heafield, 2011).

Uncased IBM BLEU was used for evaluation (Papineni et al., 2002). MERT was used to train the feature weights for the baseline systems on TUNE1. We used the learned parameters to generate M-best and diverse lists for TUNE2 and TEST to use for subsequent experiments.

5.3 Diverse List Generation

Generating diverse translations depends on two hyperparameters: the n-gram order used by the dissimilarity function Δ_n (§3.2) and the λ_j weights on the dissimilarity terms in Eq. (2). Though our framework permits different λ_j for each j, we use a single λ value for simplicity, as was also done in (Batra et al., 2012). The values of n and λ were tuned on a 200 sentence subset of TUNE1 separately for each language pair (which we call TUNE_{200}), so as to maximize the oracle BLEU score of the diverse lists.[1] We considered n values in $\{2, 3, \ldots, 9\}$ and λ values in $\{0.005, 0.01, 0.05, 0.1\}$. We give details on optimal values for these hyperparameters when discussing particular tasks below.

	AR→EN	ZH→EN	DE→EN
1 best	50.1	36.9	21.8
20 best	54.0	40.3	24.7
200 best	57.5	43.8	27.7
1000 best	59.8	46.4	29.8
unique 20 best	56.6	44.1	26.7
unique 200 best	59.6	46.4	29.5
20 diverse	58.5	46.4	28.6
20 div × 10 best	61.3	48.7	30.3
20 div × 50 best	**63.2**	**50.6**	**31.6**

Table 1: Oracle BLEU scores on TEST for various sizes of M-best and diverse lists. Unique lists were obtained from 1,000-best lists and therefore may not contain the target number of unique translations for all sentences.

Though simple, our approach is computationally expensive as M grows because it requires decoding M times for each sentence. So, we assume $M \leq 20$. But we also extract an N-best list for each of the M diverse translations.[2] Many MT decoders, including the phrase-based and hierarchical implementations in Moses, permit efficient extraction of N-best lists, so we exploit this to obtain larger lists that still exhibit diversity. But we note that these N-best lists for each diverse solution are not in themselves diverse; with more computational power or more efficient algorithms (Devlin and Matsoukas, 2012) we could potentially generate larger, more diverse lists.

6 Analysis of Diverse Lists

We now characterize our diverse lists by comparing them to M-best lists. Table 1 shows oracle BLEU scores on TEST for M-best lists, unique M-best lists, and diverse lists of several sizes. To get unique lists, we first generated 1000-best lists, then retained only the highest-scoring derivation for each unique translation. When comparing M-best and diverse lists of comparable size, the diverse lists al-

[1] Since BLEU does not decompose additively across segments, we chose translations for individual sentences that maximized BLEU+1 (Lin and Och, 2004), then computed "oracle" corpus BLEU of these translations.

[2] We did not consider n-grams from previous N-best lists when computing the dissimilarity function, but only those from the previous *diverse* translations.

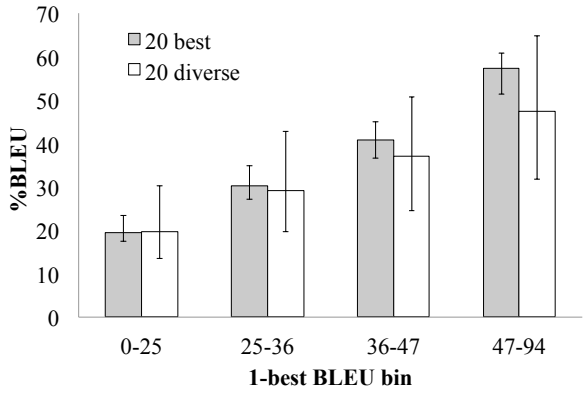


Figure 1: Median, min, and max BLEU+1 of 20-best and 20-diverse lists for the ZH→EN test set, divided into quartiles according to the BLEU+1 score of the 1-best translation, and averaged across sentences in each quartile. Heights of the bars show median and "error bars" indicate max and min.

ways have higher oracle BLEU. The differences are largest when comparing 20-best lists and 20-diverse lists, where they range from 4 to 6 BLEU points.

When generating these diverse lists, we used the n and λ values that were tuned for each language pair to maximize oracle BLEU on TUNE_{200} for the "20 div $\times$ 50 best" configuration. The optimal values of n were 6 for ZH→EN and AR→EN and 7 for DE→EN.[3] When instead tuning to maximize oracle BLEU for 20-diverse lists, the optimal n stayed at 7 for DE→EN, but increased to 7 for AR→EN and 9 for ZH→EN. These values are noticeably larger than n-gram sizes typically used in language modeling and evaluation. They suggest that for optimal oracle BLEU, translations with long-spanning amounts of repeated material should be avoided, while short overlapping n-grams are permitted.

Figure 1 shows other statistics on TEST for ZH→EN. Plots for AR→EN and DE→EN are qualitatively similar. We divided the TEST sentences into quartiles based on BLEU+1 of the 1-best translations from the baseline system. We computed the median, min, and max BLEU+1 on each list and averaged over the sentences in each quartile. As shown in the plot, the ranges of 20-diverse lists subsume those of 20-best lists, though the medians of diverse lists drop when the baseline system has high BLEU score. This matches intuition: when the baseline system is performing well, forcing it to find different translations is likely to result in worse translations. So we may expect diverse lists to be most helpful for more difficult sentences, a point we return to in our experiments below.

7 System Combination Experiments

One way to evaluate the quality of our diverse lists is to use them in system combination, as was similarly done by Devlin and Matsoukas (2012) and Cer et al. (2013). We use the system combination framework of Heafield and Lavie (2010b), which has an open-source implementation (Heafield and Lavie, 2010a).[4]

We use our baseline systems (trained on TUNE1) to generate lists for system combination on TUNE2 and TEST. We compare M-best lists, unique M-best lists, and M-diverse lists, with $M \in \{10, 15, 20\}$.[5] For each choice of list type and M, we trained the system combiner on TUNE2 and tested on TEST with the learned parameters. System combination hyperparameters (whether to use feature length normalization; the size of the k-best lists generated by the system combiner during tuning, $k \in \{300, 600\}$) were chosen to maximize BLEU on TUNE_{200}. Also, we removed the `individual` features from the default feature set because they correspond to individual systems in the combination; they did not seem appropriate for us since our hypotheses all come from the same system.

The results are shown in Table 2. Like Devlin and Matsoukas (2012), we see no gain from system combination using M-best lists. We see some improvement with unique lists, particularly for AR→EN, although it is not consistent across M values. But we see larger improvements with diverse lists for AR→EN and ZH→EN. For these language pairs, our

[3]The optimal values of λ were 0.005 for AR→EN and 0.01 for ZH→EN and DE→EN. Since these values depend on the scale of the weights learned by MERT, they are difficult to interpret in isolation.

[4]The implementation uses MERT to tune parameters, but we found this to be time-consuming and noisy for the larger feature sets. So we used a structured support vector machine learning framework instead (described in Section 8), using multiple iterations of learning interleaved with (system combiner) N-best list generation, and accumulating N-best lists across iterations.

[5]Dissimilarity hyperparameters n and λ were again chosen to maximize oracle BLEU on TUNE_{200}, separately for each M and for each language pair.

	AR→EN			ZH→EN			DE→EN		
	10	15	20	10	15	20	10	15	20
baseline (no system combination)	50.1			36.9			21.8		
M-best	50.2	50.1	50.0	36.7	36.9	37.0	21.7	21.7	21.8
unique M-best (from 1000-best list)	50.6	50.0	50.8	37.1	36.9	37.1	21.8	**21.9**	**21.9**
M-diverse	**51.4**	**51.2**	**51.2**	**37.6**	**37.6**	**37.5**	**22.0**	21.8	21.6

Table 2: System combination results (%BLEU on TEST). Size of lists is $M \in \{10, 15, 20\}$. Highest score in each column is bold.

	AR→EN				ZH→EN				DE→EN			
	q1	q2	q3	q4	q1	q2	q3	q4	q1	q2	q3	q4
baseline	30.1	44.1	55.1	**70.0**	15.2	28.9	41.0	**57.5**	5.3	14.4	23.7	**40.9**
15-best	30.1	44.6	55.5	68.8	15.9	29.2	40.5	56.8	6.0	15.0	23.6	40.0
unique 15-best	30.4	44.7	55.2	68.4	16.7	29.0	41.2	56.6	5.9	14.9	**23.8**	40.6
15-diverse	**31.3**	**45.3**	**57.8**	69.1	**17.7**	**30.6**	**41.7**	56.9	**7.6**	**15.2**	23.4	39.6

Table 3: System combination results (%BLEU on quartiles of TEST, $M = 15$). Source sentences were divided into quartiles (numbered "qn") according to BLEU+1 of the 1-best translations of the baseline system. Highest score in each column is bold.

gains are similar to those seen by Devlin and Matsoukas, but use our simpler dissimilarity function.[6] For DE→EN, results are similar for all settings and do not show much improvement from system combination.

In Table 3, we break down the scores according to 1-best BLEU+1 quartiles, as done in Figure 1.[7] In general, we find the largest gains for the low-BLEU translations. For the two worst BLEU quartiles, we see gains of 1.2 to 2.5 BLEU points, while the gains shrink or disappear entirely for the best quartile. This may be a worthwhile trade-off: a large improvement in the worst translations may be more significant to users than a smaller degredation on sentences that are already being translated well. In addition, quality estimation (Specia et al., 2011; Bach et al., 2011) could be used to automatically determine the BLEU quartile for each sentence. Then system combination of diverse translations might be used only when the 1-best translation is predicted to be of low quality.

8 Reranking Experiments

We now turn to discriminative reranking, which has frequently been used to easily add rich features to a model. It has been used for MT with varying degree of success (Och et al., 2004; Shen et al., 2004; Hildebrand and Vogel, 2008); some have attributed its mixed results to a lack of diversity in the M-best lists traditionally used. We propose diverse lists as a way to address this concern.

8.1 Learning Framework

Several learning formulations have been proposed for M-best reranking. One commonly-used approach in MT is MERT, used in the reranking experiments of Och et al. (2004) and Hildebrand and Vogel (2008), among others. We experimented with MERT and other algorithms, including pairwise ranking optimization (Hopkins and May, 2011), but we found best results using the approach of Yadollahpour et al. (2013), who used a slack-rescaled structured support vector machine (Tsochantaridis et al., 2005) with L2 regularization. As a sentence-level loss, we used negated BLEU+1. We used the 1-slack cutting-plane algorithm of Joachims et al. (2009) for optimization during learning.[8] A more detailed description of the reranker is provided in the supplementary material.

We used 5-fold cross-validation on TUNE2 to choose the regularization parameter C from the set $\{0.01, 0.1, 1, 10\}$. We selected the value yielding the highest average BLEU score across the held-out

[6] They reported +0.8 BLEU from system combination for AR→EN, and saw a further +0.5–0.7 from their new features.

[7] Quartile points are: 39, 49, 61 for AR→EN; 25, 36, and 47 for ZH→EN; and 14.5, 21.1, and 30.3 for DE→EN.

[8] Our implementation uses OOQP (Gertz and Wright, 2003) to solve the quadratic program in the inner loop, which uses HSL, a collection of Fortran codes for large-scale scientific computation (`www.hsl.rl.ac.uk`).

folds. This value was then used for one final round of training on the entirety of TUNE2. Additionally, we tuned the decision to return the parameters at convergence or those that produced the highest training corpus BLEU score. Since we use a sentence-level metric during training (BLEU+1) and a corpus-level metric for final evaluation (BLEU), we found that it was often better to return parameters that produced the highest training BLEU score.

This tuning procedure was repeated for each feature set and for each list type (M-best or diverse). The test set was not used for any of this tuning.

8.2 Features

In addition to the features from the baseline models (14 for phrase-based, 8 for hierarchical), we add 36 more for reranking:

Inverse Model 1 (INVMOD1): We added the "inverse" versions of the three IBM Model 1 features described in Section 2.2 of Hildebrand and Vogel (2008). The first is the probability of the source sentence given the translation under IBM Model 1, the second replaces the $\sum$ with a $\max$ in the first feature, and the third computes the percentage of words whose lexical translation probability falls below a threshold. We also include versions of the first 2 features normalized by the translation length, for a total of 5 INVMOD1 features.

Large LM (LLM): We created a large 4-gram LM by interpolating LMs from the WMT news data, Gigaword, Europarl, and the DE→EN news commentary (NC) corpus to maximize likelihood of a held-out development set (WMT08 test set). We used the average per-word log-probability as the single feature function in this category.

Syntactic LM (SYN): We used the syntactic treelet language model of Pauls and Klein (2012) to compute two features: the translation log probability and the length-normalized log probability.

Finite/Non-Finite Verbs (VERB): We ran the Stanford part-of-speech (POS) tagger (Toutanova et al., 2003) on each translation and added four features: the fraction of *words* tagged as finite/non-finite verbs, and the fraction of *verbs* that are finite/non-finite.[9]

[9]Words tagged as MD, VBP, VBZ, and VBD were counted as finite verbs, and VB, VBG, and VBN were non-finite verbs.

Reranking features	AR→EN best	AR→EN div	ZH→EN best	ZH→EN div	DE→EN best	DE→EN div
N/A (baseline)	50.1		36.9		21.8	
None	50.5	50.7	37.3	37.1	21.9	21.6
+ INVMOD1	50.3	50.8	**37.6**	37.1	22.0	21.8
+ LLM, SYN	50.5	51.1	37.4	37.3	21.7	21.7
+ VERB, DISC	50.4	51.3	37.3	37.3	21.9	22.2
+ GOOG	50.7	51.3	36.8	37.1	21.9	22.2
+ WCLM	51.2	**51.8**	37.3	37.4	22.2	**22.3**

Table 4: Reranking results (%BLEU on TEST).

Discriminative Word/Tag LMs (DISC): For each language pair, we generated 10,000-best lists for TUNE1 and computed BLEU+1 for each. From these lists, we estimated 3- and 5-gram LMs, weighting the n-gram counts by the BLEU+1 scores.[10] We repeated this procedure except using 1 *minus* BLEU+1 as the weight (learning a language model of "bad" translations). This yielded 4 features. The procedure was then repeated using POS tags instead of words, for 8 features in total.

Google 5-Grams (GOOG): Translations were compared to the Google 5-gram corpus (LDC2006T13) to compute: the number of 5-grams that matched, the number of 5-grams that missed, and a set of indicator features that fire if the fraction of 5-grams that matched in the sentence was greater than $\{0.05, 0.1, 0.2, \ldots, 0.9\}$, for a total of 12 features.

Word Cluster LMs (WCLM): Using an implementation provided by Liang (2005), we performed Brown clustering (Brown et al., 1992) on 900k English sentences, including the NC corpus and random sentences from Gigaword. We clustered words that appeared at least twice, once with 300 clusters and again with 1000. We then replaced words with their clusters in a large corpus consisting of the WMT news data, Gigaword, and the NC data. An additional cluster label was used for unknown words. For each of the clusterings (300 and 1000), we estimated 5- and 7-gram LMs with Witten-Bell smoothing (Witten and Bell, 1991). We added 4 features to the reranker, one for the log-probability of the translation under each of the word cluster LMs.

[10]Before estimating LMs, we projected the sentence weights so that the min and max per source sentence were 0 and 1.

List type	Features	
	None	All
20 best	50.3	50.6
100 best	50.6	50.8
200 best	50.4	51.2
1000 best	50.5	51.2
unique 20 best	50.5	51.2
unique 100 best	50.6	51.2
unique 200 best	50.4	51.3
20 diverse	50.5	51.1
20 div × 5 best	50.6	51.4
20 div × 10 best	**50.7**	51.3
20 div × 50 best	**50.7**	**51.8**

Table 5: List comparison for AR→EN reranking.

8.3 Results

Our results are shown in Table 4. We report results using the baseline system alone (labeled "N/A (baseline)"), and reranking standard M-best lists and our diverse lists. For diverse lists, we use the "20 div × 50 best" lists described in Section 5.3, with the tuned dissimilarity hyperparameters reported in Section 6. In the reranking settings, we also report results without adding any additional features (the row labeled "None").[11]

The remaining rows add features. For AR→EN, we see the largest gains, both over the baseline as well as differences between M-best lists and diverse lists. When using all features, we achieve a gain of 0.6 BLEU over M-best reranking and 1.7 BLEU points over the baseline system. The difference of 0.6 BLEU is consistent across feature subsets. We found the WCLM features to give the largest individual improvement, with the remaining feature sets each contributing a small amount. For Chinese and German, the gains and individual differences are smaller. Nonetheless, diverse lists appear to be more robust for these language pairs as features are added.

In Table 5, we compare several sizes and types of lists for AR→EN reranking both with no additional features and with the full set. We see that using 20-diverse lists nearly matches the performance of 200-best lists. Also, retaining 50-best lists for each diverse solution improves BLEU by 0.7.

[11]Though such results have not always been reported in prior work on reranking, we generally found them to improve over the baseline, presumably because seeing more data improves generalization ability.

		Train	
		best	div
Test	best	51.2	51.7
	div	50.5	51.8

Table 6: Comparing M-best and diverse lists for training/testing (AR→EN, all features).

Thus far, when training the reranker on M-best lists, we tested it on M-best lists, and similarly for diverse lists. Table 6 shows what happens with the other two pairings for AR→EN with the full feature set. When training on diverse lists, we see very little difference in BLEU whether testing on M-best or diverse lists. This has a practical benefit: we can use (computationally-expensive) diverse lists during offline training and then use fast M-best lists at test time. When training on M-best lists and *testing* on diverse lists, we see a substantial drop (51.2 vs 50.5). The reranker may be overfitting to the limited scope of translations present in typical M-best lists, thereby hindering its ability to correctly rank diverse lists at test time. These results suggest that part of the benefit of using diverse lists comes from seeing a larger portion of the output space during training.

9 Human Post-Editing Experiments

We wanted to determine whether diverse translations could be helpful to users struggling to understand the output of an imperfect MT system. We consider a post-editing task in which users are presented with translation output without the source sentence, and are asked to improve it. This setting has been studied; e.g., Koehn (2010) presented evidence that monolingual speakers could often produce improved translations for this task, occasionally reaching the level of an expert translator.

Here, we use a novel variation of this task in which *multiple* translations are shown to editors. We compare the use of entries from an M-best list and entries from a diverse list. Again, the original source sentence is not provided. Our goal is to determine whether multiple, diverse translations can help users to more accurately guess the meaning of the original sentence than entries from a standard M-best list. If so, commercial MT systems might permit users to request additional diverse translations for those sentences whose model-best translations are difficult to understand.

9.1 Translation List Post-Editing

We use Amazon Mechanical Turk (MTurk) for this experiment. Workers are shown 3 outputs from an MT system. They are not shown the original sentence, nor are they shown a reference. Based on the 3 imperfect translations, they are asked to write a single fluent English translation that best captures the understood meaning. Half of the time, the worker is shown 3 entries from an M-best list, and the other half of the time 3 entries from a diverse list. We then compare the outputs produced under the two conditions. The goal is to measure whether workers are able to produce translations that are closer in meaning to the (unseen) references when shown diverse translations. We refer to this task as the EDITING task.

To evaluate the outputs, we use a second task in which users are shown a reference translation along with two outputs from the first task: one created from M-best lists and one from diverse lists. Workers in this task are asked to choose which translation is a better match to the reference in terms of meaning, or they can indicate that the translations are of the same quality. We refer to this second task as the EVAL task.

9.2 Dissimilarity Functions

To generate diverse lists for the EDITING task, we use the same dissimilarity function as in reranking, but we tune the hyperparameters n and λ differently. Since our expectation here is that workers may combine information from multiple translations to produce a superior output, we are interested in the *coverage* of the translations in the diverse list, rather than the oracle BLEU score.

We designed a metric based on coverage of entire lists of translations. It is similar to BLEU+1, except (1) it uses n-gram recalls instead of n-gram precisions, (2) there is no brevity penalty term, and (3) it compares a *list* to a set of references and any translation in the list can contribute a match of an n-gram in any reference. Like BLEU, counts are clipped based on those in the references. We maximized this metric over diverse lists of length 5, for $n \in \{2, 3, \ldots, 9\}$ and $\lambda \in \{0.005, 0.01, 0.05, 0.1, 0.2\}$. The optimal values for AR→EN were $n = 4$ and $\lambda = 0.1$, while for ZH→EN they were $n = 4$ and $\lambda = 0.2$. These n values are smaller than for reranking, and the λ values are larger. This suggests that, when maximizing coverage of a small diverse list, more dissimilarity is desired among the translations.

9.3 Detailed Procedure

We focused on AR→EN and ZH→EN for this study. We sampled 200 sentences from their test sets, chosen from among those whose reference translation was between 5 and 25 words. We generated a unique 5-best list for each sentence using our baseline system (described in Section 5.2) and also generated a diverse list of length 5 using the dissimilarity function Δ with hyperparameters tuned using the procedure from the previous section. We untokenized and truecased the translations. We dropped non-ASCII characters because we feared they would confuse our workers. As a result, workers must contend with missing words in the output, often proper nouns.

Given the 2 lists for each sentence, we sampled two integers $i, j \in \{2, 3, 4, 5\}$ without replacement. The indices i and j indicate two entries from the lists. We took translations 1, i, and j from the 5-best list and created an EDITING task from them. We did the same using entries 1, i, and j from the diverse list. We repeated this process 3 times for each sentence, obtaining $3 \times 2 = 6$ tasks for each, giving us a total of 1,200 EDITING tasks per language pair.

The outputs of the EDITING tasks were evaluated with EVAL tasks. For each sentence, we had 3 post-edited outputs generated using entries in 5-best lists and 3 post-edited outputs from diverse lists. We created EVAL tasks for all 9 output pairs, for all 200 sentences per language pair. We additionally gave each task to three MTurk workers. This gave us 10,800 evaluation judgments for the EVAL task.

9.4 Results

Figure 2 shows the quartile breakdown for judgments collected from the EVAL task. The Y axis represents the percentage of judgments for which best/diverse outputs were preferred; the missing percentage for each bin is accounted for by "same" judgments.

We observe an interesting phenomenon. Overall, there is a slight preference for the post-edited outputs of M-best entries ("best") over those from diverse translations ("div"); this preference is clearest

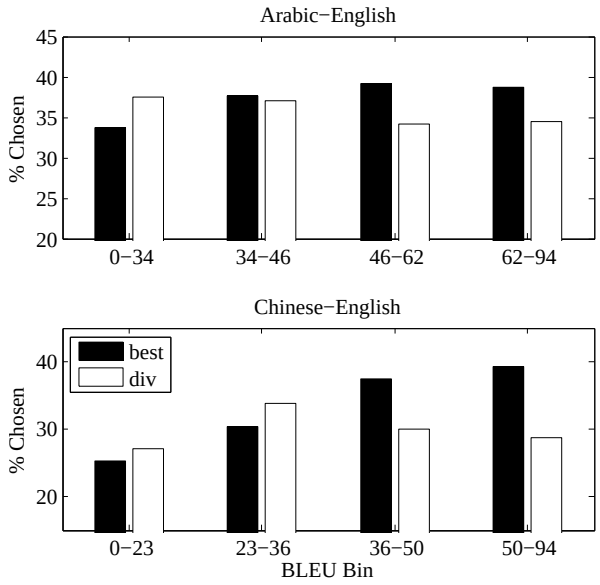


Figure 2: Percentages in which post-edited output given M-best entries ("best") was preferred by human evaluators as compared to post-edited output given diverse translations ("div"), broken down by the BLEU+1 score of the 1-best translation for the sentences. When the baseline system is doing poorly, diversity helps post-editors to produce better translations.

when the baseline system's 1-best translation had a high BLEU score. However, we see this trend reversed for sentences in which the baseline system's 1-best translation had a low BLEU score. In general, when the BLEU score of the baseline system is below 35, it is preferable to give diverse translations to users for post-editing. But when the baseline system does very well, diverse translations do not contribute anything, and in fact hurt because they may distract users from the high-quality (and typically very similar) translations from the 5-best lists.

Estimation of the quality of the output ("confidence estimation") has recently gained interest in the MT community (Specia et al., 2011; Bach et al., 2011; Callison-Burch et al., 2012; Bojar et al., 2013), including specifically for post-editing (Tatsumi, 2009; Specia, 2011; Koponen, 2012). Future work could investigate whether such automatic confidence estimation could be used to identify situations in which diverse translations can be helpful for aiding user understanding.

10 Future Work

Our dissimilarity function captures diversity in the particular phrases used by an MT system, but for certain applications we may prefer other types of diversity. Defining the dissimilarity function on POS tags or word clusters would help us to capture stylistic patterns in sentence structure, as would targeting syntactic structures in syntax-based translation.

A weakness of our approach is its computational expense; by contrast, the method of Devlin and Matsoukas (2012) obtains diverse translations more efficiently by extracting them from a single decoding of an input sentence (albeit with a wide beam). We expect their ideas to be directly applicable to our setting in order to get diverse solutions more cheaply. We also plan to explore methods of explicitly targeting multiple, diverse solutions as part of the search algorithm.

Finally, M-best lists are currently used to approximate structured spaces for many areas of MT, including tuning (Och, 2003), minimum Bayes risk decoding (Kumar and Byrne, 2004), and pipelines (Venugopal et al., 2008). Future work could replace M-best lists with diverse lists in these and related tasks, whether for MT or other areas of structured NLP.

Acknowledgments

We thank the anonymous reviewers as well as Colin Cherry, Kenneth Heafield, Silja Hildebrand, Fei Huang, Dan Klein, Adam Pauls, and Bing Xiang. DB was partially supported by the National Science Foundation under Grant No. 1353694.

References

N. Bach, F. Huang, and Y. Al-Onaizan. 2011. Goodness: A method for measuring machine translation confidence. In *Proc. of ACL*.

D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich. 2012. Diverse M-best solutions in Markov random fields. In *Proc. of ECCV*.

O. Bojar, C. Buck, C. Callison-Burch, C. Federmann, B. Haddow, P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proc. of WMT*.

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. 1992. Class-based N-gram mod-

els of natural language. *Computational Linguistics*, 18.

C. Callison-Burch, P. Koehn, C. Monz, and O. Zaidan. 2011. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proc. of WMT*.

C. Callison-Burch, P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proc. of WMT*.

D. Cer, C. D. Manning, and D. Jurafsky. 2013. Positive diversity tuning for machine translation system combination. In *Proc. of WMT*.

P. Chang, M. Galley, and C. D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proc. of WMT*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

S. Chatterjee and N. Cancedda. 2010. Minimum error rate training by sampling the translation lattice. In *Proc. of EMNLP*.

S. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report 10-98, Harvard University.

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

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1).

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

J. DeNero, D. Chiang, and K. Knight. 2009. Fast consensus decoding over translation forests. In *Proc. of ACL*.

J. Devlin and S. Matsoukas. 2012. Trait-based hypothesis selection for machine translation. In *Proc. of NAACL*.

C. Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proc. of HLT-NAACL*.

C. Dyer. 2010. *A Formal Model of Ambiguity and its Applications in Machine Translation*. Ph.D. thesis, University of Maryland.

J. R. Finkel, C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*.

E. M. Gertz and S. J. Wright. 2003. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29(1).

J. Gillenwater, A. Kulesza, and B. Taskar. 2012. Discovering diverse and salient threads in document collections. In *Proc. of EMNLP*.

K. Heafield and A. Lavie. 2010a. Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. *The Prague Bulletin of Mathematical Linguistics*, 93.

K. Heafield and A. Lavie. 2010b. Voting on n-grams for machine translation system combination. In *Proc. of AMTA*.

K. Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proc. of WMT*.

A. Hildebrand and S. Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proc. of AMTA*.

H. Hoang, P. Koehn, and A. Lopez. 2009. A Unified Framework for Phrase-Based, Hierarchical, and Syntax-Based Statistical Machine Translation. In *Proc. of IWSLT*.

M. Hopkins and J. May. 2011. Tuning as ranking. In *Proc. of EMNLP*.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

T. Joachims, T. Finley, and C. Yu. 2009. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1).

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

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL (demo session)*.

P. Koehn. 2010. Enabling monolingual translators: Post-editing vs. options. In *Proc. of NAACL*.

M. Koponen. 2012. Comparing human perceptions of post-editing effort with post-editing operations. In *Proc. of WMT*.

A. Kulesza and B. Taskar. 2010. Structured determinantal point processes. In *Proc. of NIPS*.

A. Kulesza and B. Taskar. 2011. Learning determinantal point processes. In *Proc. of UAI*.

S. Kumar and W. Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proc. of HLT-NAACL*.

S. Kumar, W. Macherey, C. Dyer, and F. Och. 2009. Efficient minimum error rate training and minimum

Bayes-risk decoding for translation hypergraphs and lattices. In *Proc. of ACL-IJCNLP*.

Y. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. 2003. Language model based Arabic word segmentation. In *Proc. of ACL*.

Z. Li, J. Eisner, and S. Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proc. of ACL*.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

C. Lin and F. J. Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. of COLING*.

W. Macherey and F. J. Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *Proc. of EMNLP-CoNLL*.

W. Macherey, F. J. Och, I. Thayer, and J. Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proc. of EMNLP*.

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

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

F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*.

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

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

A. Pauls and D. Klein. 2012. Large-scale syntactic language modeling with treelets. In *Proc. of ACL*.

A.-V. Rosti, N. F. Ayan, B. Xiang, S. Matsoukas, R. Schwartz, and B. Dorr. 2007. Combining outputs from multiple machine translation systems. In *HLT-NAACL*.

L. Shen and A. K. Joshi. 2003. An SVM-based voting algorithm with application to parse reranking. In *Proc. of CoNLL*.

L. Shen, A. Sarkar, and F. J. Och. 2004. Discriminative reranking for machine translation. In *Proc. of HLT-NAACL*.

L. Specia, N. Hajlaoui, C. Hallett, and W. Aziz. 2011. Predicting machine translation adequacy. In *Proc. of MT Summit XIII*.

L. Specia. 2011. Exploiting objective annotations for measuring translation post-editing effort. In *Proc. of EAMT*.

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

M. Tatsumi. 2009. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *Proc. of MT Summit XII*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*.

R. Tromble, S. Kumar, F. J. Och, and W. Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proc. of EMNLP*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6.

A. Venugopal, A. Zollmann, N.A. Smith, and S. Vogel. 2008. Wider pipelines: N-best alignments and parses in MT training. In *Proc. of AMTA*.

I. H. Witten and T. C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4).

T. Xiao, J. Zhu, and T. Liu. 2013. Bagging and boosting statistical machine translation systems. *Artif. Intell.*, 195.

P. Yadollahpour, D. Batra, and G. Shakhnarovich. 2013. Discriminative re-ranking of diverse segmentations. In *Proc. of CVPR*.

Max-Violation Perceptron and Forced Decoding for Scalable MT Training

Heng Yu[1]* **Liang Huang**[2]† **Haitao Mi**[3] **Kai Zhao**[2]

[1]Institute of Computing Tech. Chinese Academy of Sciences
yuheng@ict.ac.cn

[2]Queens College & Grad. Center City University of New York
{huang@cs.qc,kzhao@gc}.cuny.edu

[3]T.J. Watson Research Center IBM
hmi@us.ibm.com

Abstract

While large-scale discriminative training has triumphed in many NLP problems, its definite success on machine translation has been largely elusive. Most recent efforts along this line are not scalable (training on the small dev set with features from top ∼100 most frequent words) and overly complicated. We instead present a very simple yet theoretically motivated approach by extending the recent framework of "violation-fixing perceptron", using forced decoding to compute the target derivations. Extensive phrase-based translation experiments on both Chinese-to-English and Spanish-to-English tasks show substantial gains in BLEU by up to +2.3/+2.0 on dev/test over MERT, thanks to 20M+ sparse features. This is the first successful effort of large-scale online discriminative training for MT.

1 Introduction

Large-scale discriminative training has witnessed great success in many NLP problems such as parsing (McDonald et al., 2005) and tagging (Collins, 2002), but not yet for machine translation (MT) despite numerous recent efforts. Due to scalability issues, most of these recent methods can only train on a small dev set of about a thousand sentences rather than on the full training set, and only with 2,000–10,000 rather "dense-like" features (either unlexicalized or only considering highest-frequency words), as in MIRA (Watanabe et al., 2007; Chiang et al., 2008; Chiang, 2012), PRO (Hopkins and May, 2011), and RAMP (Gimpel and Smith, 2012). However, it is well-known that the most important features for NLP are lexicalized, most of which can not be seen on a small dataset. Furthermore, these methods often involve complicated loss functions and intricate choices of the "target" derivations to update towards or against (e.g. k-best/forest oracles, or hope/fear derivations), and are thus hard to replicate. As a result, the classical method of MERT (Och, 2003) remains the default training algorithm for MT even though it can only tune a handful of dense features. See also Section 6 for other related work.

As a notable exception, Liang et al. (2006) do train a structured perceptron model on the training data with sparse features, but fail to outperform MERT. We argue this is because structured perceptron, like many structured learning algorithms such as CRF and MIRA, assumes exact search, and search errors inevitably break theoretical properties such as convergence (Huang et al., 2012). Empirically, it is now well accepted that standard perceptron performs poorly when search error is severe (Collins and Roark, 2004; Zhang et al., 2013).

To address the search error problem we propose a very simple approach based on the recent framework of "violation-fixing perceptron" (Huang et al., 2012) which is designed specifically for inexact search, with a theoretical convergence guarantee and excellent empirical performance on beam search parsing and tagging. The basic idea is to update when search error happens, rather than at the end of the search. To adapt it to MT, we extend this framework to handle latent variables corresponding to the hidden derivations. We update towards "gold-standard" derivations computed by forced decoding so that each derivation leads to the exact reference translation. Forced decoding is also used as a way of data selection, since those reachable sentence pairs are generally more literal and of higher quality, which the training should focus on. When the reachable subset is small for some language pairs, we augment

*Work done while visiting City University of New York.
†Corresponding author.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1112–1123,
Seattle, Washington, USA, 18-21 October 2013.

it by including reachable prefix-pairs when the full sentence pair is not.

We make the following contributions:

1. Our work is the first successful effort to scale online structured learning to a large portion of the training data (as opposed to the dev set).

2. Our work is the first to use a principled learning method customized for inexact search which updates on partial derivations rather than full ones in order to fix search errors. We adapt it to MT using latent variables for derivations.

3. Contrary to the common wisdom, we show that simply updating towards the exact reference translation is helpful, which is much simpler than k-best/forest oracles or loss-augmented (e.g. hope/fear) derivations, avoiding sentence-level BLEU scores or other loss functions.

4. We present a convincing analysis that it is the search errors and standard perceptron's inability to deal with them that prevent previous work, esp. Liang et al. (2006), from succeeding.

5. Scaling to the training data enables us to engineer a very rich feature set of sparse, lexicalized, and non-local features, and we propose various ways to alleviate overfitting.

For simplicity and efficiency reasons, in this paper we use phrase-based translation, but our method has the potential to be applicable to other translation paradigms. Extensive experiments on both Chinese-to-English and Spanish-to-English tasks show statistically significant gains in BLEU by up to +2.3/+2.0 on dev/test over MERT, and up to +1.5/+1.5 over PRO, thanks to 20M+ sparse features.

2 Phrase-Based MT and Forced Decoding

We first review the basic phrase-based decoding algorithm (Koehn, 2004), which will be adapted for forced decoding.

2.1 Background: Phrase-based Decoding

We will use the following running example from Chinese to English from Mi et al. (2008):

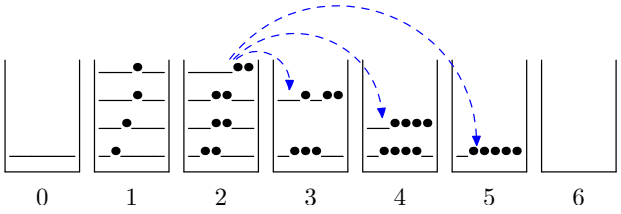


Figure 1: Standard beam-search phrase-based decoding.

Bùshí yǔ Shālóng jǔxíng le huìtán
Bush with Sharon hold -ed meeting

'Bush held a meeting with Sharon'

Phrase-based decoders generate partial target-language outputs in left-to-right order in the form of *hypotheses* (or states) (Koehn, 2004). Each hypothesis has a *coverage vector* capturing the source-language words translated so far, and can be extended into a longer hypothesis by a phrase-pair translating an uncovered segment. For example, the following is one possible derivation:

$$\frac{\dfrac{\dfrac{(0\_\_\_\_\_) : (0, \text{“”})}{(\bullet_1\_\_\_\_\_) : (s_1, \text{“Bush”})}\ r_1}{(\bullet\_\_\bullet\bullet\bullet_6) : (s_2, \text{“Bush held talks”})}\ r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet) : (s_3, \text{“Bush held talks with Sharon”})}\ r_3$$

where a $\bullet$ in the coverage vector indicates the source word at this position is "covered" and where each s_i is the score of each state, each adding the rule score and the distortion cost (dc) to the score of the previous state. To compute the distortion cost we also need to maintain the ending position of the last phrase (e.g., the $_3$ and $_6$ in the coverage vectors). In phrase-based translation there is also a *distortion-limit* which prohibits long-distance reorderings.

The above states are called $-$LM states since they do not involve language model costs. To add a bigram model, we split each $-$LM state into a series of $+$LM states; each $+$LM state has the form $(v,^a)$ where a is the last word of the hypothesis. Thus a $+$LM version of the above derivation might be:

$$\frac{\dfrac{\dfrac{(0\_\_\_\_\_,^{\text{<s>}}) : (0, \text{“<s>”})}{(\bullet_1\_\_\_\_\_,^{\text{Bush}}) : (s'_1, \text{“<s> Bush”})}\ r_1}{(\bullet\_\_\bullet\bullet\bullet_6,^{\text{talks}}) : (s'_2, \text{“<s> Bush held talks”})}\ r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet,^{\text{Sharon}}) : (s'_3, \text{“<s> Bush held ... with Sharon”})}\ r_3$$

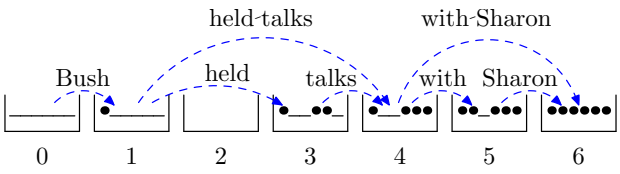


Figure 2: Forced decoding and y-good derivation lattice.

where the score of applying each rule now also includes a *combination cost* due to the bigrams formed when applying the phrase-pair, e.g.

$$s'_3 = s'_2 + s(r_3) + dc(|6-3|) - \log P_{lm}(\text{with} \mid \text{talk})$$

To make this exponential-time algorithm practical, beam search is the standard approximate search method (Koehn, 2004). Here we group +LM states into n bins, with each bin B_i hosting at most b states that cover exactly i Chinese words (see Figure 1).

2.2 Forced Decoding

The idea of forced decoding is to consider only those (partial) derivations that can produce (a prefix of) the exact reference translation (assuming single reference). We call these partial derivations "y-good" derivations (Daumé, III and Marcu, 2005), and those that deviate from the reference translation "y-bad" derivations. The forced decoding algorithm is very similar to +LM decoding introduced above, with the new "forced decoding LM" to be defined as only accepting two consecutive words on the reference translation, ruling out any y-bad hypothesis:

$$P_{forced}(b \mid a) = \begin{cases} 1 & \text{if } \exists j, \text{ s.t. } a = y_j \text{ and } b = y_{j+1} \\ 0 & \text{otherwise} \end{cases}$$

In the +LM state, we can simply replace the boundary word by the index on the reference translation:

$$\frac{\dfrac{\dfrac{(0\_\_\_\_\_,{}^{0}) : (0, \text{“<s>”})}{(\bullet_1\_\_\_\_,{}^{1}) : (w'_1, \text{“<s> Bush”})}\, r_1}{(\bullet\_\_\bullet\bullet\bullet_6,{}^{3}) : (w'_2, \text{“<s> Bush held talks”})}\, r_2}{(\bullet\bullet\bullet_3\bullet\bullet\bullet,{}^{5}) : (w'_3, \text{“<s> Bush held talks with Sharon”})}\, r_3$$

The complexity of this forced decoding algorithm is reduced to $O(2^n n^3)$ where n is the source sentence length, without the expensive bookkeeping for English boundary words.

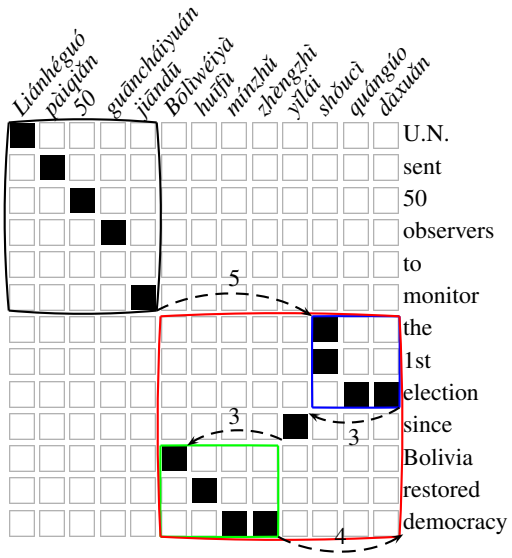


Figure 3: Example of unreachable sentence pair and reachable prefix-pair. The first big jump is disallowed for a distortion limit of 4, but we can still extract the top-left box as a reachable prefix-pair. Note that this example is perfectly reachable in syntax-based MT.

2.3 Reachable Prefix-Pairs

In practice, many sentence pairs in the parallel text fail in forced decoding due to two reasons:

1. **distortion limit**: long-distance reorderings are disallowed but are very common between languages with very different word orders such as English and Chinese.

2. **noisy alignment and phrase limit**: the word-alignment quality (typically from GIZA++) are usually very noisy, which leads to unnecessarily big chunks of rules beyond the phrase limit.

If we only rely on the reachable whole sentence pairs, we will not be able to use much of the training set for Chinese-English. So we propose to augment the set of reachable examples by considering reachable prefix-pairs (see Figure 3 for an example).

3 Violation-Fixing Perceptron for MT

Huang et al. (2012) establish a theoretical framework called "violation-fixing perceptron" which is tailored for structured learning with inexact search and has provable convergence properties. The high-level idea is that standard full update does not fix search errors; to do that we should instead update when search error occurs, e.g., when the gold-

standard derivation falls below the beam. Huang et al. (2012) show dramatic improvements in the quality of the learned model using violation-fixing perceptron (compared to standard perceptron) on incremental parsing and part-of-speech tagging.

Since phrase-based decoding is also an incremental search problem which closely resembles beam-search incremental parsing, it is very natural to employ violation-fixing perceptron here for MT training. Our goal is to produce the exact reference translation, or in other words, we want at least one y-good derivation to survive in the beam search.

To adapt the violation-fixing perceptron framework to MT we need to extend the framework to handle latent variables since the gold-standard derivation is not observed. This is done in a way similar to the latent variable structured perceptron (Zettlemoyer and Collins, 2005; Liang et al., 2006; Sun et al., 2009) where each update is from the best (y-bad) derivation towards the best y-good derivation in the current model; the latter is a constrained search which is exactly forced decoding in MT.

3.1 Notations

We first establish some necessary notations. Let $\langle x, y \rangle$ be a sentence pair in the training data, and

$$d = r_1 \circ r_2 \circ \ldots \circ r_{|d|}$$

be a (partial) derivation, where each $r_i = \langle c(r_i), e(r_i) \rangle$ is a rule, i.e., a Chinese-English phrase-pair. Let $|c(d)| \triangleq \sum_i |c(r_i)|$ be the number of Chinese words covered by this derivation, and $e(d) \triangleq e(r_1) \circ e(r_2) \ldots \circ e(r_{|d|})$ be the English prefix generated so far. Let $D(x)$ be the set of all possible partial derivations translating part of the input sentence x. Let $pre(y) \triangleq \{y_{[0:j]} \mid 0 \leq j \leq |y|\}$ be the set of prefixes of the reference translation y, and $good_i(x, y)$ be the set of partial y-good derivations whose English side is a prefix of the reference translation y, and whose Chinese projection covers exactly i words on the input sentence x, i.e.,

$$good_i(x, y) \triangleq \{d \in D(x) \mid e(d) \in pre(y), |c(d)| = i\}.$$

Conversely, we define the set of y-bad partial derivations covering i Chinese words to be:

$$bad_i(x, y) \triangleq \{d \in D(x) \mid e(d) \notin pre(y), |c(d)| = i\}.$$

Basically, at each bin B_i, y-good derivations $good_i(x, y)$ and y-bad ones $bad_i(x, y)$ compete for the b slots in the bin:

$$B_0 = \{\epsilon\} \quad (1)$$

$$B_i = \mathbf{top}^b \bigcup_{j=1..l} \{d \circ r \mid d \in B_{i-j}, |c(r)| = j\} \quad (2)$$

where r is a rule covering j Chinese words, l is the phrase-limit, and $\mathbf{top}^b\ S$ is a shorthand for $\mathbf{argtop}^b_{d \in S}\ \mathbf{w} \cdot \mathbf{\Phi}(x, d)$ which selects the top b derivations according to the current model $\mathbf{w}$.

3.2 Algorithm 1: Early Update

As a special case of violation-fixing perceptron, early update (Collins and Roark, 2004) stops decoding whenever the gold derivation falls off the beam, makes an update on the prefix so far and move on to the next example. We adapt it to MT as follows: if at a certain bin B_i, all y-good derivations in $good_i(x, y)$ have fallen off the bin, then we stop and update, rewarding the best y-good derivation in $good_i(x, y)$ (with respect to current model $\mathbf{w}$), and penalizing the best y-bad derivation in the same step:

$$d_i^+(x, y) \triangleq \mathop{\mathbf{argmax}}_{d \in good_i(x,y)} \mathbf{w} \cdot \mathbf{\Phi}(x, d) \quad (3)$$

$$d_i^-(x, y) \triangleq \mathop{\mathbf{argmax}}_{d \in bad_i(x,y) \cap B_i} \mathbf{w} \cdot \mathbf{\Phi}(x, d) \quad (4)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x,\ d_i^+(x, y),\ d_i^-(x, y)) \quad (5)$$

where $\Delta\mathbf{\Phi}(x, d, d') \triangleq \mathbf{\Phi}(x, d) - \mathbf{\Phi}(x, d')$ is a shorthand notation for the difference of feature vectors. Note that the set $good_i(x, y)$ is independent of the beam search and current model and is instead precomputed in the forced decoding phase, whereas the negative signal $d_i^-(x, y)$ depends on the beam.

In practice, however, there are exponentially many y-good derivations for each reachable sentence pair, and our goal is just to make sure (at least) one y-good derivation triumphs at the end. So it is possible that at a certain bin, all y-good partial derivations fall off the bin, but the search can still continue and produce the exact reference translation through some other y-good path that avoids that bin. For example, in Figure 1, the y-good states in steps 3 and 5 are not critical; it is totally fine to miss them in the search as long as we save the y-good states

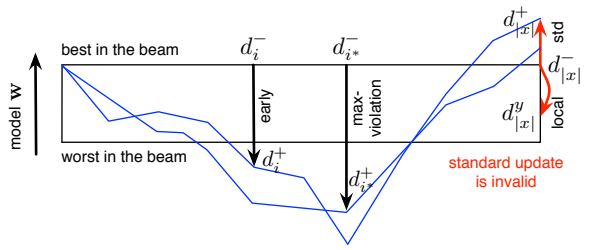


Figure 4: Illustration of four update methods. The blue paths denote (possibly lots of) gold-standard derivations from forced decoding. Standard update in this case is invalid as it reinforces the error of $\mathbf{w}$ (Huang et al., 2012).

in bins 1, 4 and 6. So we actually use a "softer" version of the early update algorithm: only stop and update when there is no hope to continue. To be more concrete, let l denote the phrase-limit then we stop where there are l consecutive bins without any y-good states, and update on the first among them.

3.3 Algorithm 2: Max-Violation Update

While early update learns substantially better models than standard perceptron in the midst of inexact search, it is also well-known to be converging much slower than the latter, since each update is on a (short) prefix. Huang et al. (2012) propose an improved method "*max-violation*" which updates at the worst mistake instead of the first, and converges much faster than early update with similar or better accuracy. We adopt this idea here as follows: decode the whole sentence, and find the step i^* where the difference between the best y-good derivation and the best y-bad one is the biggest. This amount of difference is called the amount of "violation" in Huang et al. (2012), and the place of maximum violation is intuitively the site of the biggest mistake during the search. More formally, the update rule is:

$$i^* \triangleq \operatorname*{argmin}_i \mathbf{w} \cdot \Delta\boldsymbol{\Phi}(x,\, d_i^+(x,y),\, d_i^-(x,y)) \quad (6)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\boldsymbol{\Phi}(x,\, d_{i^*}^+(x,y),\, d_{i^*}^-(x,y)) \quad (7)$$

3.4 Previous Work: Standard and Local Updates

We compare the above new update methods with the two existing ones from Liang et al. (2006).

Standard update (also known as "bold update" in Liang et al. (2006)) simply updates at the very end, from the best derivation in the beam towards the best gold-standard derivation (regardless of whether it survives the beam search):

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\boldsymbol{\Phi}(x, d_{|x|}^+(x,y), d_{|x|}^-(x,y)) \quad (8)$$

Local update, however, updates towards the derivation in the final bin that is most similar to the reference y, denoted $d_{|x|}^y(x,y)$:

$$d_{|x|}^y(x,y) = \operatorname*{argmax}_{d \in B_{|x|}} \mathrm{Bleu}^{+1}(y, e(d)) \quad (9)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\boldsymbol{\Phi}(x, d_{|x|}^y(x,y), d_{|x|}^-(x,y)) \quad (10)$$

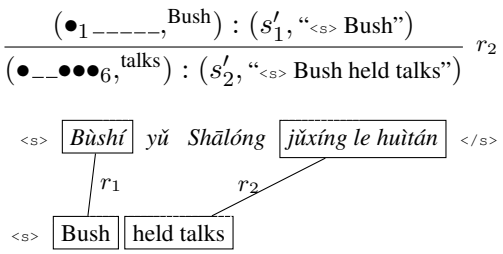
where $\mathrm{Bleu}^{+1}(\cdot,\cdot)$ returns the sentence-level BLEU.

Liang et al. (2006) observe that standard update performs worse than local update, which they attribute to the fact that the former often update towards a gold derivation made up of "unreasonable" rules. Here we give a very different but theoretically more reasonable explanation based on the theory of Huang et al. (2012), who define an update $\Delta\boldsymbol{\Phi}(x, d^+, d^-)$ to be **invalid** if d^+ scores higher than d^- (i.e., $\mathbf{w} \cdot \Delta\boldsymbol{\Phi}(x, d^+, d^-) > 0$, or update $\Delta\mathbf{w}$ points to the same direction as $\mathbf{w}$ in Fig. 4), in which case there is no "violation" or mistake to fix. Perceptron is guaranteed to converge if all updates are valid. Clearly, early and max-violation updates are valid. But standard update is not: it is possible that at the end of search, the best y-good derivation $d_{|x|}^+(x,y)$, though pruned earlier in the search, ranks even higher in the current model than anything in the final bin (see Figure 4). In other words, there is no mistake at the final step, while there must be some search error in earlier steps which expels the y-good subderivation. We will see in Section 5.3 that invalid updates due to search errors are indeed the main reason why standard update fails. Local update, however, is always valid in that definition.

Finally, it is worth noting that in terms of implementation, standard and max-violation are the easiest, while early update is more involved.

4 Feature Design

Our feature set includes the following 11 dense features: LM, four conditional and lexical translation probabilities ($p_c(e|f)$, $p_c(f|e)$, $p_l(e|f)$, $p_l(f|e)$), length and phrase penalties, distortion cost, and three lexicalized reordering features. All these features are inherited from Moses (Koehn et al., 2007).



features for applying r_2 on span $x_{[3:6]}$

WordEdges	$c(r_2)_{[0]} = \textit{jǔxíng}$, $c(r_2)_{[-1]} = \textit{huìtán}$ $e(r_2)_{[0]} = \text{held}$, $e(r_2)_{[-1]} = \text{talks}$ $x_{[2:3]} = \textit{Shālóng}$, $x_{[6:7]} = \text{</s>}$, $\lvert c(r_2)\rvert = 3$ ... (**combos** of the above **atomic** features) ...
non-local	$e(r_0 \circ r_1)_{[-2:]} \circ id(r_2)$ $id(r_1) \circ id(r_2)$

Figure 5: Examples of WordEdges and non-local features. The notation uses the Python style subscript syntax.

4.1 Local Sparse Features: Ruleid & WordEdges

We first add the rule identification feature for each rule: $id(r_i)$. We also introduce lexicalized **WordEdges** features, which are shown to be very effective in parsing (Charniak and Johnson, 2005) and MT (Liu et al., 2008; He et al., 2008) literatures. We use the following atomic features when applying a rule $r_i = \langle c(r_i), e(r_i)\rangle$: the source-side length $|c(r_i)|$, the boundary words of both $c(r_i)$ and $e(r_i)$, and the surrounding words of $c(r_i)$ on the input sentence x. See Figure 5 for examples. These atomic features are concatenated to generate all kinds of **combo features**.

Chinese	English	class size		budget
word		52.9k	64.2k	5
characters	-	3.7k	-	3
Brown cluster, full string		200		3
Brown cluster, prefix 6		6	8	2
Brown cluster, prefix 4		4	4	2
POS tag		52	36	2
word type	-	4	-	1

Table 1: Various levels of backoff for WordEdges features. Class size is estimated on the small Chinese-English dataset (Sec. 5.3). The POS tagsets are ICTCLAS for Chinese (Zhang et al., 2003) and Penn Treebank for English (Marcus et al., 1993).

4.2 Addressing Overfitting

With large numbers of lexicalized combo features we will face the overfitting problem, where some combo features found in the training data are too rare to be seen in the test data. Thus we propose three ways to alleviate this problem.

First, we introduce various levels of backoffs for each word w (see Table 1). We include w's Brown cluster and its prefixes of lengths 4 and 6 (Brown et al., 1992), and w's part-of-speech tag. If w is Chinese we also include its word type (punctuations, digits, alpha, or otherwise) and (leftmost or rightmost) character. In such a way, we significantly increase the feature coverage on unseen data.

However, if we allow arbitrary combinations, we can extract a hexalexical feature (4 Chinese + 2 English words) for a local window in Figure 5, which is unlikely to be seen at test time. To control model complexity we introduce a *feature budget* for each level of backoffs, shown in the last column in Table 1. The total budget for a combo feature is the sum of the budgets of all atomic features. In our experiments, we only use the combo features with a total budget of 10 or less, i.e., we can only include bilexical but not trilexical features, and we can include for example combo features with one Chinese word plus two English tags (total budget: 9).

Finally, we use two methods to alleviate overfitting due to one-count rules: for large datasets, we simply remove all one-count rules, but for small datasets where out-of-vocabulary words (OOVs) abound, we use a simple leave-one-out method: when training on a sentence pair (x, y), do not use the one-count rules extracted from (x, y) itself.

4.3 Non-Local Features

Following the success of non-local features in parsing (Huang, 2008) and MT (Vaswani et al., 2011), we also introduce them to capture the contextual information in MT. Our non-local features, shown in Figure 5, include bigram rule-ids and the concatenation of a rule id with the translation history, i.e. the last two English words. Note that we also use backoffs (Table 1) for the words included. Experiments (Section 5.3) show that although the set of non-local features is just a tiny fraction of all features, it contributes substantially to the improvement in BLEU.

Scale	Language Pair	Training Data # sent.	Training Data # words	Reachability sent.	Reachability words	# feats	ΔBLEU # refs	ΔBLEU dev/test	Sections
small	CH-EN	30K	0.8M/1.0M	21.4%	8.8%	7M	4	+2.2/2.0	5.2, 5.3
large		230K	6.9M/8.9M	32.1%	12.7%	23M		+2.3/2.0	5.2, 5.4
large	SP-EN	174K	4.9M/4.3M	55.0%	43.9%	21M	1	+1.3/1.1	5.5

Table 2: Overview of all experiments. The ΔBLEU column shows the absolute improvements of our method MAX-FORCE on dev/test sets over MERT. The Chinese datasets also use prefix-pairs in training (see Table 3).

5 Experiments

In order to test our approach in different language pairs, we conduct three experiments, shown in Table 2, on two significantly different language pairs (long vs. short distance reorderings), Chinese-to-English (CH-EN) and Spanish-to-English (SP-EN).

5.1 System Preparation and Data

We base our experiments on **Cubit**, a state-of-art phrase-based system in Python (Huang and Chiang, 2007).[1] We set phrase-limit to 7 in rule extraction, and beam size to 30 and distortion limit 6 in decoding. We compare our violation-fixing perceptron with two popular tuning methods: MERT (Och, 2003) and PRO (Hopkins and May, 2011).

For word alignments we use GIZA++-ℓ_0 (Vaswani et al., 2012) which produces sparser alignments, alleviating the garbage collection problem. We use the SRILM toolkit (Stolcke, 2002) to train a trigram language model with modified Kneser-Ney smoothing on 1.5M English sentences.

Our dev and test sets for CH-EN task are from the newswire portion of 2006 and 2008 NIST MT Evaluations (616/691 sentences, 18575/18875 words), with four references.[2] The dev and test sets for SP-EN task are from newstest2012 and newstest2013, with only one reference. Below both MERT and PRO tune weights on the dev set, while our method on the training set. Specifically, our method only uses the dev set to know when to stop training.

5.2 Forced Decoding Reachability on Chinese

As mentioned in Section 2.2, we perform forced decoding to select reachable sentences from the training data; this part is done with exact search without any beam pruning. Figure 6 shows the reachability ratio vs. sentence length on the small CH-EN training data, where the ratio decreases sharply with sentence length, and increases with distortion limit. We can see that there are a lot of long distance reorderings beyond small distortion limits. In the extreme case of unlimited distortion, a large amount of sentences will be reachable, but at the cost of much slower decoding ($O(n^2V^2)$ in beam search decoding, and $O(2^n n^3)$ in forced decoding). In fact forced decoding is too slow in the unlimited mode that we only plot reachability for sentences up to 30 words.

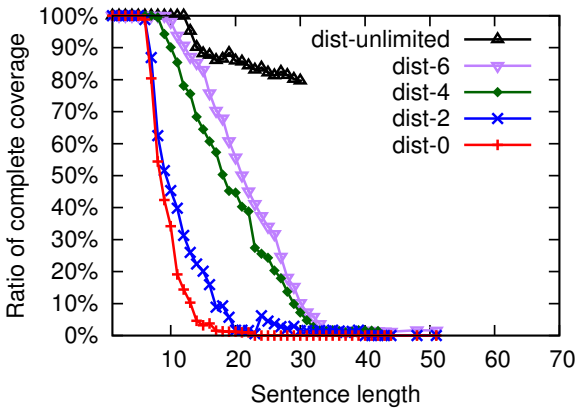


Figure 6: Reachability ratio vs. sentence length on the small CH-EN training set.

	small sent.	small words	large sent.	large words
full	21.4%	8.8%	32.1%	12.7%
+prefix	61.3%	24.6%	67.3%	32.8%

Table 3: Ratio of sentence reachability and word coverage on the two CH-EN training data (distortion limit: 6).

Table 3 shows the statistics of forced decoding on both small and large CH-EN training sets. In the

[1] http://www.cis.upenn.edu/~lhuang3/cubit/. We will release the new version at http://acl.cs.qc.edu.

[2] We use the "average" reference length to compute the brevity penalty factor, which does not decrease with more references unlike the "shortest" heuristic.

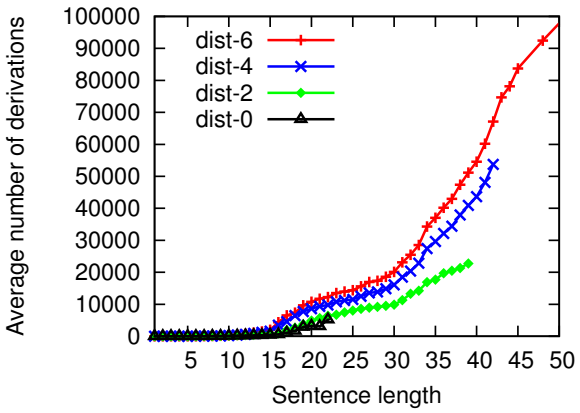


Figure 7: Average number of derivations in gold lattices.

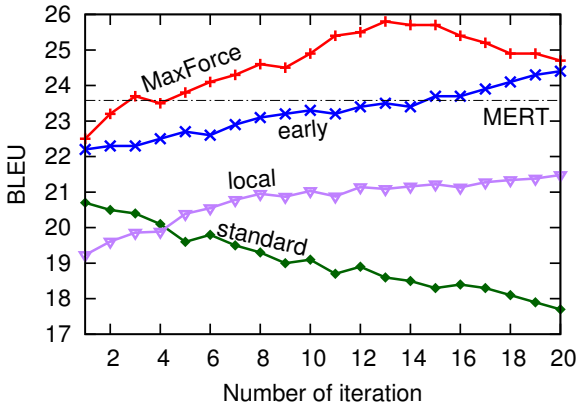


Figure 8: BLEU scores on the heldout dev set for different update methods (trained on small CH-EN data).

small data-set, 21.4% sentences are fully reachable which only contains 8.8% words (since shorter sentences are more likely to be reachable). Larger data improves reachable ratios significantly thanks to better alignment quality, but still only 12.7% words can be used. In order to add more examples for perceptron training, we pick all non-trivial reachable prefix-pairs (with 5 or more Chinese words) as additional training examples (see Section 2.2). As shown in Table 3, with prefix-pairs we can use about 1/4 of small data and 1/3 of large data for training, which is 10x and 120x bigger than the 616-sentence dev set.

After running forced decoding, we obtain gold translation lattice for each reachable sentence (or prefix) pair. Figure 7 shows, as expected, the average number of gold derivations in these lattices grows exponentially with sentence length.

5.3 Analysis on Small Chinese-English Data

Figure 8 shows the BLEU scores of different learning algorithms on the dev set. MAXFORCE[3] performs the best, peaking at iteration 13 while early update learns much slower (the first few iterations are faster than other methods due to early stopping but this difference is immaterial later). The local and standard updates, however, underperform MERT; in particular, the latter gets worse as training goes on.

As analysized in Section 3.4, the reason why standard update (or "bold update" in Liang et al. (2006)) fails is that inexact search leads to many invalid updates. This is confirmed by Figure 9, where more

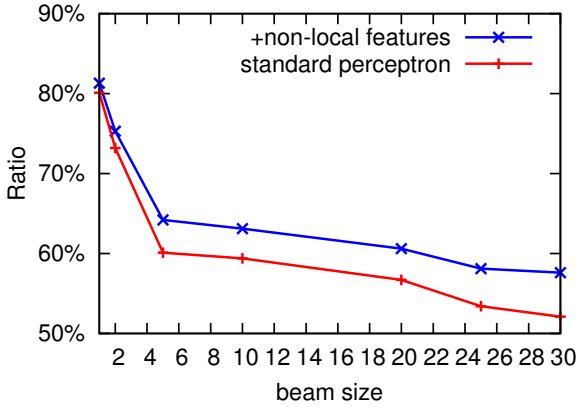


Figure 9: Ratio of invalid updates in standard update.

than half of the updates remain invalid even at a beam of 30. These analyses provide an alternative but theoretically more reasonable explanation to the findings of Liang et al. (2006): while they blame "unreasonable" gold derivations for the failure of standard update, we observe that it is the search errors that make the real difference, and that an update that respects search errors towards a gold subderivation is indeed helpful, even if that subderivation might be "unreasonable".

In order to speedup training, we use mini-batch parallelization of Zhao and Huang (2013) which has been shown to be much faster than previous parallelization methods. We set the mini-batch size to 24 and train MAXFORCE with 1, 6, and 24 cores on a small subset of the our original reachable sen-

[3]Stands for **Max**-Violation Perceptron w/ **Force**d Decoding

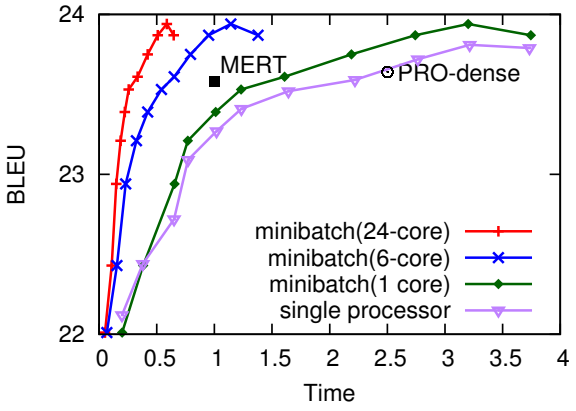


Figure 10: Minibatch parallelization speeds up learning.

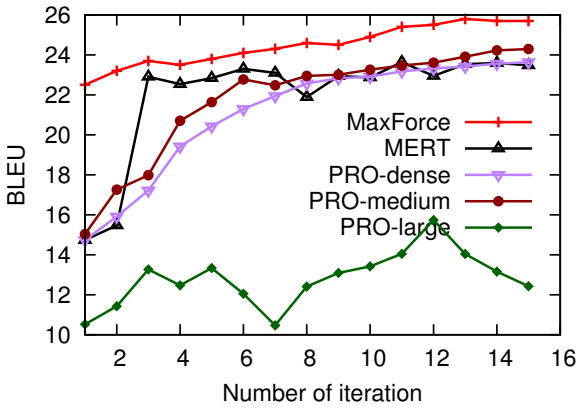


Figure 11: Comparison between different training methods. Ours trains the training set while others on dev set.

tences. The number of sentence pairs in this subset is 1,032, which contains similar number of words to our 616-sentence dev set (since reachable sentences are much shorter). Thus, it is reasonable to compare different learning algorithms in terms of speed and performance. Figure 10 shows that first of all, minibatch improves BLEU even in the serial setting, and when run on 24 cores, it leads to a speedup of about 7x. It is also interesting to know that on 1 CPU, minibatch perceptron takes similar amount of time to reach the same performance as MERT and PRO.

Figure 11 compares the learning curves of MAXFORCE, MERT, and PRO. We test PRO in three different ways: PRO-dense (dense features only), PRO-medium (dense features plus top 3K most frequent sparse features[4]), and PRO-large (dense features plus all sparse features). The results show that PRO-dense performs almost the same as MERT but with a stabler learning curve while PRO-medium improves by +0.6. However, PRO-large decreases the performance significantly, which indicates PRO is not scalable to truly sparse features. By contrast, our method handles large-scale sparse features well and outperforms all other methods by a large margin and with a stable learning curve.

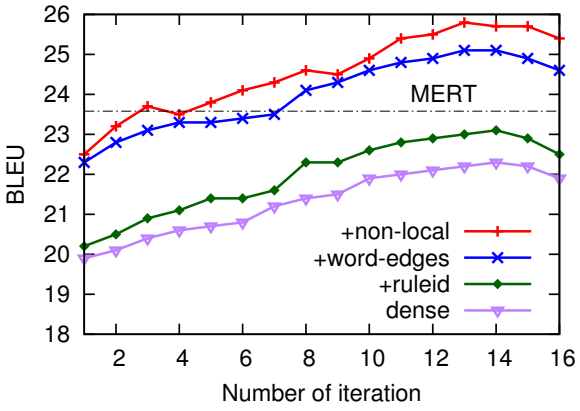


Figure 12: Incremental contributions of different feature sets (dense features, ruleid, WordEdges, and non-local).

type	count	%	BLEU
dense	11	-	22.3
+ruleid	+9,264	+0.1%	+0.8
+WordEdges	+7,046,238	+99.5%	+2.0
+non-local	+22,536	+0.3%	+0.7
all	7,074,049	100%	25.8

Table 4: Feature counts and incremental BLEU improvements. MAXFORCE with all features is +2.2 over MERT.

We also investigate the individual contribution from each group of features (ruleid, WordEdges, and non-local features). So we perform experiments by adding each group incrementally. Figure 12 shows the learning curves and Table 4 lists the counts and incremental contributions of different feature sets. With dense features alone MAXFORCE does not do

[4]To prevent overfitting we remove all lexicalized features and only use Brown clusters. It is difficult to engineer the right feature set for PRO, whereas MAXFORCE is much more robust.

system	algorithm	# feat.	dev	test
Moses	MERT	11	25.5	22.5
Cubit	MERT	11	25.4	22.5
	PRO	11	25.6	22.6
		3K	26.3	23.0
		36K	17.7	14.3
	MAXFORCE	23M	**27.8**	**24.5**

Table 5: BLEU scores (with four references) using the large CH-EN data. Our approach is +2.3/2.0 over MERT.

well because perceptron is known to suffer from features of vastly different scales. Adding ruleid helps, but still not enough. WordEdges (which is the vast majority of features) improves BLEU by +2.0 points and outperforms MERT, when sparse features totally dominate dense features. Finally, the 0.3% non-local features contribute a final +0.7 in BLEU.

5.4 Results on Large Chinese-English Data

Table 5 shows all BLEU scores for different learning algorithms on the large CH-EN data. The MERT baseline on Cubit is essentially the same as Moses. Our MAXFORCE activates 23M features on reachable sentences and prefixes in the training data, and takes 35 hours to finish 15 iterations on 24 cores, peaking at iteration 13. It achieves significant improvements over other approaches: +2.3/+2.0 points over MERT and +1.5/+1.5 over PRO-medium on dev/test sets, respectively.

5.5 Results on Large Spanish-English Data

In SP-EN translation, we first run forced decoding on the training set, and achieve a very high reachability of 55% (with the same distortion limit of 6), which is expected since the word order between Spanish and English are more similar than than between Chinese and English, and most SP-EN reorderings are local. Table 6 shows that MAXFORCE improves the translation quality over MERT by +1.3/+1.1 BLEU on dev/test. These gains are comparable to the improvements on the CH-EN task, since it is well accepted in MT literature that a change of δ in 1-reference BLEU is roughly equivalent to a change of 2δ with 4 references.

system	algorithm	# feat.	dev	test
Moses	MERT	11	27.4	24.4
Cubit	MAXFORCE	21M	**28.7**	**25.5**

Table 6: BLEU scores **(with one reference)** on SP-EN.

6 Related Work

Besides those discussed in Section 1, there are also some research on tuning sparse features on the training data, but they integrate those sparse features into the MT log-linear model as a single feature weight, and tune its weight on the dev set (e.g. (Liu et al., 2008; He et al., 2008; Wuebker et al., 2010; Simianer et al., 2012; Flanigan et al., 2013; Setiawan and Zhou, 2013; He and Deng, 2012; Gao and He, 2013)). By contrast, our approach learns sparse features only on the training set, and use dev set as held-out to know when to stop.

Forced decoding has been used in the MT literature. For example, open source MT systems Moses and cdec have implemented it. Liang et al. (2012) also use the it to boost the MERT tuning by adding more y-good derivations to the standard k-best list.

7 Conclusions and Future Work

We have presented a simple yet effective approach of structured learning for machine translation which scales, for the first time, to a large portion of the whole training data, and enables us to tune a rich set of sparse, lexical, and non-local features. Our approach results in very significant BLEU gains over MERT and PRO baselines. For future work, we will consider other translation paradigms such as hierarchical phrase-based or syntax-based MT.

Acknowledgement

We thank the three anonymous reviewers for helpful suggestions. We are also grateful to David Chiang, Dan Gildea, Yoav Goldberg, Yifan He, Abe Ittycheriah, and Hao Zhang for discussions, and Chris Callison-Burch, Philipp Koehn, Lemao Liu, and Taro Watanabe for help with datasets. Huang, Yu, and Zhao are supported by DARPA FA8750-13-2-0041 (DEFT), a Google Faculty Research Award, and a PSC-CUNY Award, and Mi by DARPA HR0011-12-C-0015. Yu is also supported by the China 863 State Key Project (No. 2011AA01A207). The views and findings in this paper are those of the authors and are not endorsed by the US or Chinese governments.

References

Peter Brown, Peter Desouza, Robert Mercer, Vincent Pietra, and Jenifer Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor, Michigan, June.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP 2008*.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *J. Machine Learning Research (JMLR)*, 13:1159–1187.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Hal Daumé, III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of ICML*.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of NAACL 2013*.

Jianfeng Gao and Xiaodong He. 2013. Training mrf-based phrase translation models using gradient ascent. In *Proceedings of NAACL:HLT*, pages 450–459, Atlanta, Georgia, June.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of NAACL 2012*.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of ACL*.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proceedings of COLING*, pages 321–328, Manchester, UK, August.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of EMNLP*.

Liang Huang and David Chiang. 2007. Forest rescoring: Fast decoding with integrated language models. In *Proceedings of ACL*, Prague, Czech Rep., June.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of NAACL*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*, pages 115–124.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*, Sydney, Australia, July.

Huashen Liang, Min Zhang, and Tiejun Zhao. 2012. Forced decoding for minimum error rate training in statistical machine translation. *Journal of Computational Information Systems*, (8):861868.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proceedings of EMNLP*, pages 89–97, Honolulu, Hawaii, October.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL*.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Hendra Setiawan and Bowen Zhou. 2013. Discriminative training of 150 million translation parameters and its application to pruning. In *Proceedings of NAACL:HLT*, pages 335–341, Atlanta, Georgia, June. ACL.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of ACL*, Jeju Island, Korea.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Xu Sun, Takuya Matsuzaki, and Daisuke Okanohara. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of IJCAI*.

Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. 2011. Rule markov models for fast tree-to-string translation. In *Proceedings of ACL 2011*, Portland, OR.

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the L0-norm. In *Proceedings of ACL*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proceedings of ACL*, pages 475–484, Uppsala, Sweden, July.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, pages 184–187.

Hao Zhang, Liang Huang, Kai Zhao, and Ryan McDonald. 2013. Online learning with inexact hypergraph search. In *Proceedings of EMNLP 2013*.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of NAACL 2013*.

Identifying Multiple Userids of the Same Author

Tieyun Qian
State Key Laboratory of Software Eng.
Wuhan University
16 Luojiashan Road
Wuhan, Hubei 430072, China
qty@whu.edu.cn

Bing Liu
Department of Computer Science
University of Illinois at Chicago
851 South Morgan St., Chicago
IL, USA, 60607
liub@cs.uic.edu

Abstract

This paper studies the problem of identifying users who use multiple userids to post in social media. Since multiple userids may belong to the same author, it is hard to directly apply supervised learning to solve the problem. This paper proposes a new method, which still uses supervised learning but does not require training documents from the involved userids. Instead, it uses documents from other userids for classifier building. The classifier can be applied to documents of the involved userids. This is possible because we transform the document space to a similarity space and learning is performed in this new space. Our evaluation is done in the online review domain. The experimental results using a large number of userids and their reviews show that the proposed method is highly effective.

1 Introduction

It is common knowledge that some users in social media register multiple accounts/userids to post articles, blogs, reviews, etc. There are many reasons for doing this. For example, due to past postings, a user may become despised by others. He/she then registers another userid in order to regain his/her status. A user may also use multiple userids to instigate controversy or debates to popularize a topic to make it "hot" or even just to promote activities at a website. Yet, a user may also use multiple userids to post fake or deceptive opinions to promote or demote some products (Liu, 2012). It is thus important to develop technologies to identify such *multi-id users*. This paper deals with this problem based on writing style and other linguistic clues.

Problem definition: Given a set of userids $ID = \{id_1, \ldots, id_n\}$ and each id_i has a set of documents D_i, we want to identify userids that belong to the same physical author.

The main related works to ours are in the area of authorship attribution (AA), which aims to identify authors of documents. AA is often solved using supervised learning. Let $A = \{a_1, \ldots, a_k\}$ be a set of authors (or classes) and each author $a_i \in A$ has a set of training documents D_i. A classifier is then built to decide the author a of each test document d, where $a \in A$. We will discuss this and other related works in Section 2.

This supervised AA formulation, however, is not suitable for our task because we only have userids but not real authors. Since some of the userids may belong to the same author, we cannot treat each userid as a class because in that case, we will be classifying based on userids, which won't help us find authors with multiple userids (see Section 7 also).

This paper proposes a novel algorithm. To simplify the presentation, we assume that at most two userids can belong to a single author, but the algorithm can be extended to handle more than two userids from the same author. Using this assumption, the algorithm works in two steps:

1. *Candidate identification*: For each userid id_i, we first find the most likely userid id_j ($i \neq j$) that may have the same author as id_i. We call id_j the *candidate* of id_i. We also call this function *candid-iden*, i.e., id_j = *candid-iden*(id_i). For easy presentation, here we only use one argument for

* The work was mainly done when the first author was visiting the University of Illinois at Chicago.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1124–1135,
Seattle, Washington, USA, 18-21 October 2013.

candid-iden. In the computation, it needs more arguments (see Section 4).

2. *Candidate confirmation*: In the reverse order, we apply the function *candid-iden* on id_j, which produces id_k, i.e., $id_k = candid\text{-}iden(id_j)$.

   Decision making: If $k = i$, we conclude that id_i and id_j are from the same author. Otherwise, id_i and id_j are not from the same author.

The key of the algorithm is *candid-iden*. An obvious approach for *candid-iden* is to use an information retrieval method. We can first split the documents D_i of each id_i into two subsets, a *query set* Q_i and a *sample set* S_i. We then compare each query document in Q_i with each sample document in S_j from other userids id_j ($\in ID - \{id_i\}$). *Cosine* can be used here for similarity comparison. All the similarity scores are then aggregated and used to rank the userids in $ID - \{id_i\}$. The top ranked userid is the candidate for id_i. Note that partitioning the documents of a userid id_i into the query set Q_i and the sample set S_i is crucial here. We cannot use all documents in D_i to compare with all documents in D_j. If so and we get $candid\text{-}iden(id_i) = id_j$, we will definitely get $candid\text{-}iden(id_j) = id_i$ since the similarity function is symmetric.

This cosine similarity based method, however, does not work well (see Section 7). We propose a supervised learning method to compute the scores. For this, we need to reformulate the problem.

The idea of this reformulation is to learn in a similarity space rather than in the original document space as in traditional AA. In the new formulation, each document *d* is still represented as a feature vector, but the vector no longer represents the document *d* itself. Instead, it represents a set of similarities between the document *d* and a query *q*. We call this method *learning in the similarity space* (LSS).

Specifically, in LSS, each document *d* is first represented with a *document space vector* (called a *d-vector*) based on the document itself as in the traditional classification learning of AA. Each feature in the *d*-vector is called a *d-feature* (*document-feature*). A query document *q* is represented in the same way. We then produce a *similarity vector sv* (called *s-vector*) for *d*. *sv* consists of a set of similarity values between document *d* (in a *d*-vector) and query *q* (in a *d*-vector):

$$sv = Sim(d, q),$$

where *Sim* is a similarity function consists of a set of similarity measures. Thus, the *d*-vector for document *d* in the document space is transformed to an *s*-vector *sv* for *d* in the similarity space. Each feature in *sv* is called an *s-feature*. For example, we have the following *d*-vector for query *q*:

q: 1:1 2:1 6:2

where *x*:*z* represents a *d*-feature *x* (a word) and its frequency *z* in *q*. We also have two non-query documents, one is d_1 which is written by the author of query *q* and the other is d_2 which is not written by query author *q*. Their *d*-vectors are:

d_1: 1:2 2:1 3:1 d_2: 2:2 3:1 5:2

If we use *cosine* as the first similarity measure in *Sim*, we can generate an *s*-feature 1:0.50 for d_1 ($cosine(q, d_1) = 0.50$) and an *s*-feature 1:0.27 for d_2 ($cosine(q, d_2) = 0.27$). If we have more similarity measures more *s*-features can be produced. The resulting two *s*-vectors for d_1 and d_2 with their class labels, 1 and -1, are as follows:

d_1: 1 1:0.50 … d_2: -1 1:0.27 …

Class 1 means "written by author of query *q*", also called *q-positive*, and class -1 means "not written by author of query *q*", also called *q-negative*.

LSS gives us a two-class classification problem. In this formulation, a test userid and his/her documents do not have to be seen in training as long as a set of known documents from this userid is available. Any supervised learning method can be used to build a classifier. We use SVM. The resulting classifier is employed to compute a score for each review to be used in the two-step algorithm above to find the candidate for each userid and then the userids with the same authors.

Due to the use of query documents, the LSS formulation has some resemblance to document ranking based on learning to rank (Li, 2011; Liu, 2011). However, LSS is very different because we turn the problem into a supervised classification problem. The key difference between learning to rank and classification is that ranking will always put some documents at the top even if the desired documents do not exist. However, classification will not return any document if the desired documents do not exist in the test data (unless there are classification errors). Our Type II experiments in Section 7 were specifically designed for testing such non-existence situations.

Using online review as the application domain, we conduct experiments on a large number of reviews and their author/reviewer userids from Amazon.com. The results show that the proposed algorithm is highly accurate and outperforms three strong baselines markedly.

2 Related Work

A similar problem was attempted in (Chen et al., 2004) in the context of open forums where users interact with each other in their discussions. Their method is based on post relationships and intervals between posts. It does not use any linguistic clues. It is thus not applicable to domains like online reviews. Reviews do not involve user interactions since each review is independent of other reviews. Novak et al. also solved the same problem under the name of "Anti-aliasing" (Novak et al., 2004). They used a clustering based method which assumed the number of actual authors is known. This is unrealistic in practice as there is no way to know which author has and does not have multiple ids.

Our work is also related to authorship attribution (AA). However, to our knowledge, our problem has not been attempted in AA. Existing works focused on two main themes: finding good writing style features, and developing effective classification methods. On finding good features (d-features in our case), it was found that the most promising features are function words (Mosteller, 1964; Argamon and Levitan, 2004; Argamon et al., 2007) and rewrite rules (Halteren et al., 1996). Length (Gamon 2004; Graham et al., 2005), richness (Halteren et al., 1996; Koppel and Schler, 2004), punctuations (Graham et al., 2005), character n-grams (Grieve, 2007; Hedegaard and Simonsen, 2011), word *n*-grams (Burrows, 1992; Sanderson and Guenter 2006), POS *n*-grams (Gamon, 2004; Hirst and Feiguina, 2007), syntactic category pairs (Narayanan et al., 2012) are also useful.

On classification, numerous methods have been tried, e.g., Bayesian analysis (Mosteller, 1964), discriminant analysis (Stamatatos et al., 2000), PCA (Hoover, 2001), neural networks (Graham et al., 2005; Zheng et al., 2006; Graham et al., 2005), clustering (Sanderson and Guenter, 2006), decision trees (Uzuner and Katz, 2005; Zhao and Zobel, 2005), regularized least squares classification (Narayanan et al., 2012), and SVM (Diederich et al., 2000; Gamon 2004; Koppel and Schler, 2004; Hedegaard and Simonsen, 2011). Among them, SVM was found to be most accurate (Li et al., 2006; Kim et al., 2011). Although we also use supervised learning, we do not learn in the original document space as these existing methods do. The transformation is important because it enables us to use documents from other authors in training. The traditional supervised learning (TSL) cannot do that. In our case, the only documents that TSL can use for training are the queries in the testing set. However, as we will see in our experiments, such a method performs poorly.

Since we use online reviews as our experiment domain, our work is related to fake review detection (Jindal and Liu, 2008) as imposters can use multiple userids to post fake reviews. Existing research has proposed many methods to detect fake reviewers (Lim et al., 2010; Wang et al., 2011; Mukherjee et al., 2012) and fake reviews (Jindal and Liu, 2008; Ott et al., 2011, 2012; Li et al., 2011; Feng et al., 2012). However, none of them identifies userids belonging to the same person.

3 Learning in the Similarity Space

We now formulate the proposed supervised learning in the similarity space (LSS), which will be used in the *candid-iden* function in our algorithm to be discussed in Section 4.

The key difference between LSS and the classic document space learning is in the document representation. Another difference is in the testing phase. We discuss testing first.

Test data: We are given:

- A query q from query author (userid) aq
- A set of test documents $DT = \{dt_1, \ldots, dt_m\}$.

Goal: classify the test documents into those authored by aq and those not authored by aq.

We note the following points:

i) This is like a retrieval scenario, but we use supervised learning to perform the task.

ii) Unlike traditional supervised classification, here the test query author aq does not have to be used in training. But we are given a query document q from aq. Clearly, in practice, we can have multiple query documents from aq, which we will discuss in Section 4.

Training document representation: As noted earlier, each document is represented with a similarity vector (*s-vector*) computed using a similarity

1. **For** each author $ar_i \in AR$
2. select a set of query documents $Q_i \subseteq DR_i$
3. **For** each query $q_{ij} \in Q_i$
// produce positive *s*-training examples
4. select a set of documents from author ar_i
$DR_{ij} \subseteq DR_i - \{q_{ij}\}$
5. **For** each document $dr_{ijk} \in DR_{ij}$
6. produce an *s*-training example for dr_{ijk},
($Sim(dr_{ijk}, q_{ij})$, 1)
// produce negative *s*-training examples
7. select a set of documents from the rest of authors
$DR_{ij,rest} \subseteq (DR_1 \cup \ldots \cup DR_n) - DR_i$
8. **For** each document $dr_{ijk,rest} \in DR_{ij,rest}$
9. produce an *s*-training example for $dr_{ijk,rest}$,
($Sim(dr_{ijk,rest}, q_{ij})$, -1)

Figure 1: Generating *s*-training examples

// Author ar_1 –
// positive (1) *s*-training examples
($Sim(dr_{111}, q_{11})$, 1), …, ($Sim(dr_{11p}, q_{11})$, 1)
…
($Sim(dr_{1k1}, q_{1k})$, 1), …, ($Sim(dr_{1kp}, q_{1k})$, 1)
// negative (-1) *s*-training examples
($Sim(dr_{111.rest}, q_{11})$, -1), …, ($Sim(dr_{11u.rest}, q_{11})$, -1)
…
($Sim(dr_{1k1.rest}, q_{1k})$, -1), …, ($Sim(dr_{1ku.rest}, q_{1k})$, -1)
…

Figure 2: *s*-training examples

function *Sim*. *Sim* takes a query document and a non-query document and produces a vector of similarity values or s-features to represent the non-query document. We present the detail below:

Let the set of training authors be $AR = \{ar_1, .., ar_n\}$. Each author ar_i has a set of documents DR_i. Each document in DR_i is first represented with a document vector (or *d*-vector). The algorithm for producing the training set, called *s-training set*, is given in Figure 1.

We randomly select a small set of queries Q_i from documents DR_i of each author ar_i (lines 1, and 2). For each query $q_{ij} \in Q_i$ (line 3), it selects a set of documents DR_{ij} also from DR_i (excluding q_{ij}) of the same author (line 4) to be the positive documents for q_{ij}, called *q-positive* and labeled 1. Then, for each document dr_{ijk} in DR_{ij}, a *q*-positive *s*-training example with the label 1 is generated for dr_{ijk} by computing the similarities of q_{ij} and dr_{ijk} using the similarity function *Sim* (lines 5, 6). In line 7, it selects a set of documents $DR_{ij,rest}$ from other authors to be the negative documents for q_{ij}, called *q*-negative and labeled -1. For each document $dr_{ijk,rest}$ in $DR_{ij,rest}$ (line 8), a *q*-negative *s*-training example with label -1 is generated for dr_{ijk} by computing the similarities of q_{ij} and $dr_{ijk,rest}$ using *Sim* (line 9). How to select Q_i, DR_{ij} and $DR_{ij,rest}$ (lines 2, 4 and 7) is left open intentionally to give flexibility in implementation.

1. **For** each document set D_i of $id_i \in ID$ **do**
2. partition D_i into two subsets:
(1) query set Q_i and (2) sample set S_i;
3. **For** each document set D_i of $id_i \in ID$ **do**
// step 1: *candidate identification*
4. id_j = candid-iden(id_i, *ID*), $i < j$;
// step 2: *candidate confirmation*
5. id_k = candid-iden(id_j, *ID*), $k \neq j$;
6. **If** $k = i$ **then** id_i and id_j are from the same author
8. **else** id_i and id_j are not from the same author

Figure 3: Identifying userids from the same authors

Function candidate-iden(id_i, *ID*)
1. **For** each sample document set S_j of $id_j \in ID$-$\{id_i\}$ **do**
2. $pcount[id_j]$, $psum[id_j]$, $psqsum[id_j]$, $max[id_j] = 0$;
3. **For** each query $q_i \in Q_i$ **do**
4. **For** each sample $s_{jf} \in S_j$ **do**
5. $ss_{jf} = <(id_i, q_i), (Sim(s_{jf}, q_i), ?)>$;
6. Classify ss_{jf} using the classifier built earlier;
7. **If** ss_{jf} is classified positive, i.e., 1 **then**
8. $pcount[id_j] = pcount[id_j] + 1$;
9. $psum[id_j] = psum[id_j] + ss_{jf}.score$
10 $psqsum[id_j] = psqsum[id_j] + (ss_{jf}.score)^2$
11. **If** $ss_{if}.score > max[id_j]$ **then**
12. $max[id_j] = sr_{jf}.score$
// Four methods to decide which id_j is the candidate for id_i
13. **If** for all $id_j \in ID$-$\{id_i\}$, $pcount[id_i] = 0$ **then**
14. $cid = \arg\max_{id_j \in ID-\{id_i\}} (\max[id_j])$
15. **Else** $cid = \arg\max_{id_j \in ID-\{id_i\}} (\frac{pcount[id_j]}{|S_j|})$ **// 1. Voting**
16. $cid = \arg\max_{id_j \in ID-\{id_i\}} (\frac{psum[id_j]}{|S_j|})$ // 2. **ScoreSum**
17. $cid = \arg\max_{id_j \in ID-\{id_i\}} (\frac{(psum[id_j])^2}{|S_j|})$ // 3. **ScoreSqSum**
18. $cid = \arg\max_{id_j \in ID-\{id_i\}} (\max[id_j])$ **// 4. ScoreMax**
19. return *cid*;

Figure 4: Identifying the candidate

This formulation gives us a two-class classification problem. The classes are 1 (*q-positive* meaning "written by author of query q_{ij}") and -1 (*q-negative* meaning "not written by author of query q_{ij}." Figure 2 shows what the *s*-training data looks like. For easy presentation, we assume that there are *k* queries in every Q_i, and *p* documents in every DR_{ij} and *u* documents in every $DR_{ij,rest}$. The number of authors is *n*. Each author ar_i generates $k \times (p+u)$ *s*-training examples. As we will see in Section 7, *k* can be very small, even 1.

Complexity: In the worst case, every document

can serve as a query or a non-query document. Then we need to compute all pairwise document similarities. If the number of training documents is m, the complexity is $O(m^2)$, which is both space and computation expensive. However, in practice, we don't need all pairwise comparisons. Only a small subset is sufficient (see Section 7).

Test document representation: Like training documents, test documents are represented as *s*-vectors as well in the similarity space.

Given a query q from author aq and a set of test documents DT, each test document dt_i is converted to a *s*-vector $sv_i = Sim(dt_i, q)$. To reflect sv_i is computed based on query q from author aq, a *s-test* case is thus represented as $<(aq, q), (sv_i, ?)>$.

Training: A binary classifier is learned using the *s*-training data. Each *s*-training example is represented with (sv, y), where sv is an *s*-vector and y ($\in \{1, -1\}$) is its class. Any supervised learning algorithm, e.g., SVM, can be applied.

Testing: The classifier is applied to each *s*-test case $<(aq, q), (sv_i, ?)>$ (where $sv_i = S(dt_i, q)$) to give it a class *q-positive* or *q-negative*. Note that the classifier is only applied on sv_i.

In most cases, classification based on a single query is inaccurate. Using multiple queries of an author can classify much more accurately.

4 Identify Userids of the Same Author

We now expand the sketch of the two-step algorithm in Section 1 based on the problem statement in Section 1. The algorithm is given in Figure 3.

Lines 1-2 partitions the documents set D_i of each id_i in $ID = \{id_1, id_2, \ldots, id_n\}$, the set of userids that we are working on. How to do the partition is flexible (see Section 7). Line 4 is the step 1 of candidate identification, and line 5 is the step2 of candidate confirmation. Lines 6-8 is the decision making of step 2 (see Section 1). Line 6 produced a classification score using the classifier described in Section 3. The key function here is *candid-iden*. Its algorithm is in Figure 4.

The *candid-iden* function takes two arguments: the query userid id_i and the whole set of userids ID. It classifies each sample ss_{jf} in sample set S_j of $id_j \in ID\text{-}\{id_i\}$ to positive (q_i-positive) or negative (q_i-negative) (lines 4, 5, 6). We then aggregate the classification results to determine which userid is likely to have the same author as id_i.

One simple aggregation method is voting. We count the total number of positive classifications of the sample documents of each userid in $ID\text{-}\{id_i\}$. The userid id_j with the highest count is the candidate *cid* which may share the same author as query id_i. *cid* is returned as the candidate.

There are also other methods, which can depend on what output value the classifier produces. Here we propose four methods including the voting method above. The other three methods requires the classifier to produces a prediction score, which reflects the positive and negative certainty. Many classification algorithms produce such a score. Here we use SVM. For each classification, SVM outputs a positive or negative score indicating the certainty that the test case is positive or negative.

To save space, all four alternative methods are given in Figure 4. Line 2 initializes some variables for recording the aggregated values for the final decision making. The four methods are as follows:

1). **Voting**: For each sample from userid id_j, if it is classified as positive, one vote/count is added to $pcount[id_j]$. The userid with the highest *pcount* is regarded as the candidate userid, *cid* (line 15). Note that the normalization is applied because the sizes of the sample sets S_j can be different for different userids. Lines 13 and 14 mean that if all documents of all userids are classified as negative ($pcount[id_j] = 0$, which also implies $psum[id_j] = psqsum[id_j] = 0$), we use method 4).

2). **ScoreSum**: This method works similarly to the voting method above except that instead of counting positive classifications, this method sums up all scores of positive classifications in $psum[id_j]$ for each userid (line 9). The decision is also made similarly (line 16).

3). **ScoreSqSum**: This method works similarly to ScoreSum above except that it sums up the squared scores of positive classifications in $psqsum[id_j]$ for each userid (line 10). The decision is also made similarly (line 17).

4). **ScoreMax**: This method works similarly to the voting method as well except that it finds the maximum classification score for the documents of each userid (lines 11 and 12). The decision is made in line 18.

5 D-features

We now compute *s*-features (similarity features)

for each non-query document based on a query document. Since *s*-features are calculated using *d*-features of a non-query document and a query document, we thus discuss *d*-features first, which are extracted from each document itself. We employ 26 *d*-features in four categories: length *d*-features, frequency based *d*-features, tf.idf based *d*-features, and richness *d*-features. Although many features below have been used in various tasks before, our key contribution is solving a new problem based on a new learning formulation (LSS).

Length d-feature: We derive three length *d*-features from each raw document: (1) *average sentence length* (in terms of word count); (2) *average word length* (in terms of character count in one word); (3) *average document length* (in terms of word count in one document).

Frequency based d-features: We extract lexical, syntactic, and stylistic tokens from the raw documents and the parsed syntactic trees to produce the following features:

- *Lexical tokens: word unigrams*
- *Syntactic tokens:* content-independent structures: POS n-grams ($1 \leq n \leq 3$) and rewrite rules (Halteren et al., 1996; Hirst and Feiguina, 2007). A rewrite rule is a combination of a node and its immediate constituents in a syntactic tree. For example, the rewrite rule for "the best book" is NP->DT+JJS+NN.
- *Common stylistic token: K*-length word ($1 \leq K \leq 15$), punctuations, and 157 function words (*www.flesl.net/Vocabulary/SinglewordLists/functionwordlist.php*).
- *Review specific stylistic tokens:* These tokens reflect styles of reviews: *all cap words*, *pairs of quotation marks*, *pairs of brackets*, *exclamatory marks*, *contractions*, *two or more consecutive non-alphanumeric characters*, *model auxiliaries* (e.g., *should*, *must*), word "*recommend*" or "*recommended*", *sentences with the first letter capitalized*, *sentences starting with This is (this is) or This was* (*this was*). We then treat these tokens as pseudo-words and count their frequency to form frequency *d*-features.

TF-IDF based d-feature: For the tokens listed in the frequency based features above, we also compute their tf.idf values. We list these two kinds of *d*-features separately because they will be used for different *s*-features later.

Richness d-features: This is a set of vocabulary richness functions used to quantify the diversity of vocabulary in text (Holmes and Forsyth, 1995). In this paper, we apply them to the counts of word unigrams, POS *n*-grams ($1 \leq n \leq 3$), and rewrite rules. Here POS *n*-grams and rewrite rules are treated as pseudo-words. Let *T* be the total number of tokens (words or pseudo-words), and *V(T)* be the number of different tokens in a document, *v* be the highest frequency of occurrence of a token, and *V(m, T)* be the number of tokens which occur *m* times in the document. We use the following six richness measures (Yule, 1944; Burrows, 1992; Halteren et al., 1996) given in Table 1: Yule's characteristic (*K*), Hapax dislegomena (*S*), Simpson's index (*D*), Honorës measure (*R*), Brunet's measure (*W*), and Hapax legomena (*H*). They give us a set of richness *d*-features about word unigrams, POS n-grams, and rewrite rules.

Table 1. Richness metrics

$K = 10^4 * \frac{\sum_{m=1}^{v} (m^2 * V(m,T) - T)}{T^2}$	$S = \frac{V(2,T)}{V(T)}$
$D = \frac{\sum_{m=1}^{v} (m*(m-1)*V(m,T))}{T*(T-1)}$	$R = \frac{100*\log(T)}{1 - V(1,T)/V(T)}$
$H = V(1,T)$	$W = T^{V(T)^{-a}}, a = 0.17$

6 S-Features

The extracted *d*-features are transformed into *s*-features, which are a set of similarity functions on two documents. We adopt five types of *s*-features.

Sim4 Length s-features: This is a set of four similarity functions defined by us. They are used for *d*-feature vectors of length. The four formulae are given in Table 2, where l_{wq}. (l_{wd}), l_{sq}. (l_{sd}), and l_{rq}. (l_{rd}) denote the average word, sentence, and document length respectively, either in query *q* or non-query document *d*. They produce four *s*-features.

Table 2. Sim4 for computing length s-features

$1/(1+\log(1+\lvert l_{wq} - l_{wd} \rvert))$
$1/(1+\log(1+\lvert l_{sq} - l_{sd} \rvert))$
$1/(1+\log(1+\lvert l_{rq} - l_{rd} \rvert))$
$\sum_{m \in \{w,s,r\}} (l_{mq} * l_{md}) / \sqrt{\sum_{m \in \{w,s,r\}} (l_{mq})^2} * \sqrt{\sum_{m \in \{w,s,r\}} (l_{md})^2}$

Sim3 Sentence s-features: This is a set of three sentence similarity functions (Metzler et al., 2005). We apply them (called Sim3) to documents. Sim3 *s*-features are used for frequency based *d*-features. The three formulae are given in Table 3, where $f(t, s)$ is the frequency count of token t in a document s, and l_q and l_d are the average document length of the query and non-query document, respectively.

Table 3. Sim3 for computing sentence s-features

$\sum_{t\in q\cap d} f(t,d)/(\sum_{t\in q} f(t,q)+\sum_{t\in d} f(t,d)-\sum_{t\in q\cap d} f(t,d))$
$\log_{t\in q\cap d}(\frac{N}{f(t,d)})*\frac{\sum_{t\in q\cap d} f(t,d)}{\sum_{t\in q} f(t,q)+\sum_{t\in d} f(t,d)-\sum_{t\in q\cap d} f(t,d)}$
$\frac{1}{1+\log(1+\lvert l_q-l_d\rvert)}*\sum_{t\in q\cap d}\frac{N*idf(t)}{1+\lvert f(t,q)-f(t,d)\rvert}$

Sim7 Retrieval s-features: This is a set of seven similarity functions (Table 4) applicable to all frequency based *d*-features. These functions were used in information retrieval (Cao et al., 2006).

Table 4. Sim7 for computing retrieval *s*-features

$\sum_{t\in q\cap d}\log(f(t,d)+1)$	$\sum_{t\in q\cap d}\log(\frac{\lvert D\rvert}{f(t,d)}+1)$
$\sum_{t\in q\cap d}\log(idf(t))$	$\sum_{t\in q\cap d}\log(\frac{f(t,d)}{\lvert d\rvert}+1)$
$\sum_{t\in q\cap d}\log(\frac{f(t,d)}{\lvert d\rvert}*idf(t)+1)$	$\log(BM25score)$
$\sum_{t\in q\cap d}\log(\frac{f(t,d)}{\lvert d\rvert}*\frac{\lvert D\rvert}{f(t,d)}+1)$	

In Table 4, $f(t, d)$ denotes the frequency count of token t in a non-query document d, q denotes the query, D is the entire collection, |.| is the size of a set, and *idf* is the inverse document frequency. These 7 formulae can produce 7 *s*-features.

SimC tf-idf s-feature: This is the *cosine* similarity used for *d*-vectors represented by the tf.idf based *d*-features. SimC tf-idf produces one *s*-feature.

SimC Richness s-feature: This is also cosine similarity. However, it is applied to the richness *d*-feature vectors, and produces one *s*-feature.

7 Experimental Evaluation

We now evaluate the proposed approach and compare it with baselines. All our experiments use the SVM^{perf} classifier (Joachims, 2006).

7.1 Experiment Setup

Experiment Data: We use a set of reviews and their authors/reviewers from Amazon.com as our experiment data. We select the authors who have posted more than 30 reviews in the book category. After cleaning, we have 831 authors, 731 authors for training and 100 authors for testing. The numbers of reviews in the training and test author set are 59256 and 14308, respectively. We use the Stanford parser (Klein and Manning, 2003) to generate the grammar structure of review sentences for extracting syntactic *d*-features. Note that the authors here are in fact userids. However, since they are randomly selected from a large number of userids, the probability that two sampled userids belong to the same person is very small. Thus, it should be safe to assume that each userid here represents a unique author.

Training data: We randomly choose 1 (one) review for each author as the query and all of his/her other reviews as *q*-positive reviews. The *q*-negative reviews consist of reviews randomly selected from the other 730 authors, two reviews per author. We also tried to use more queries from each author, but they make little difference.

Test data: The test authors are all unseen, i.e., their reviews have not been used in training. We prepare the test case for each author as follows.

We first divide the reviews of each author into two equal subsets. The purpose is to simulate the situation where there are two userids id_{ia} and id_{ib} from the same author a_i. Our objective is that given one userid id_{ia} and its query set, we want to find the other userid id_{ib} from the same author.

For the review subset of id_{ia} (or id_{ib}), we randomly select 9 reviews as the *query set* and another 10 reviews as the *sample set* for the userid. The two sets are disjoint. We don't use more queries or sample reviews from each author since in the review domain most authors do not have many reviews (Jindal and Liu, 2008). In the experiments, we will vary the number of test userids, the number of queries, and the number of samples. We use the following format to describe each test data: T<n>_Q<n>S<n>, where T denotes the total number of test userids, Q the query set and S the sample set, and <n> a number. For example, T50_Q9S10 stands for a test data with 50 userids, and for each userid, 9 reviews are selected as queries and 10 reviews are selected as samples. * rep-

resents a wildcard whose value we can vary.

Note that we use this "artificial" data rather than manually labeled data for our experiments because it is very hard to reliably label any gold-standard data manually in this case. The problem is similar to labeling fake reviews. In the fake review detection research, researchers have manually label fake reviews and reviewers (Yoo and Gretzel 2009; Lim et al., 2010; Li et al., 2011; Wang et al., 2011). However, based on the actual fake reviews written using Amazon Mechanical Turk, Ott et al. (2011) have showed that the accuracy of human labeling of fake reviews is very poor. We also believe that our test data is realistic for evaluation as we can image that the two sets of reviews are from two accounts (userids) of the same author (reviewer).

Two types of experiments: For each author with two userids, we conduct two types of tests.

- **Type I**: Identify two userids belong to the same author. The experiment runs iteratively to test every userid. In each iteration, we plant one userid of an author in the test set and use the other userid of the same author as the query userid. That is, in the i^{th} run, the test data consist of the following two components:
  1. Query userid id_{ia} and its query set Q_{ia}
  2. Test userids $\{id_{1a}, \ldots, id_{(i-1)a}, id_{ib}, \ldots, id_{ma}\}$ and their corresponding sample review sets $\{S_{1a}, \ldots, S_{(i-1)a}, S_{ib}, \ldots, S_{ma}\}$.

  Note that the query userid id_{ia} and the test userid id_{ib} are from the same author. Our objective is to use Q_{ia} to find id_{ib} through S_{ib}.

  Evaluation measure: We use precision, recall, and F_1 score to evaluate Type I experiments as we want to identify all matching pairs. The errors are "no pair" and "wrong pair" found.
- **Type II**: Type II experiments test the cases when no pair exists. That is, we do not plant any matching userid for the query userid. Then, the algorithm should not find anything. For the i^{th} run, the test data has these components:
  1. Query userid id_{ia} and its query set Q_{ia}
  2. Test userids $\{id_{1a}, \ldots, id_{(i-1)a}, id_{(i+1)a}, \ldots, id_{ma}\}$ and their sample review sets $\{S_{1a}, \ldots, S_{(i-1)a}, S_{(i+1)a}, \ldots, S_{ma}\}$. id_{ib} is not planted.

  Evaluation measure: Here we cannot use precision and recall because we are not trying to find any pairs. We thus use accuracy as our measure. For each id_i, if no pair is found, it is correct. If a pair is found, it is wrong.

Baseline methods: As mentioned eariler, there are only two works that tried to identify multi-id users. The first is that in (Chen et al., 2004). However, as we discussed in related work, their approach is not applicable to reviews. The other is that in (Novak et al., 2004), which used clustering but assumed that the number of actual authors (or clusters) is known. This is unrealistic in practice. Thus we designed three new baselines:

TSL: This baseline is based on the traditional supervised learning (TSL). We use it to evaluate how the traditional approach performs in the original feature space. In this case, each document in TSL has to be represented as a vector of d-features or traditional n-gram features. For each test userid *id*, we build a SVM classifier based on the *one vs. all* strategy. That is, for training we use *id*'s queries in T*_Q*S10 as the positive documents, and all queries of the other test userids (e.g., 99 userids if the test data has 100 userids) as the negative documents. Note that TSL cannot use the 731 userids for training as in LSS because they do not appear in the test data. In testing, userid *id*'s sample (non-query) documents in T*_Q*S10 are used as positive documents, and the sample documents of all other test userids are used as negative documents.

SimUG: It uses the word unigrams to compare the cosine similarity of queries and samples. Cosine similarity with unigrams is the most widely used document similarity measure.

SimAD: It uses all *d*-features to compare the cosine similarity of queries and samples.

For both *SimUG* and *SimAD*, their cosine similarity values are used in place of SVM scores of LSS or TSL. We then apply the same 4 strategies to decide the final author attribution except voting as cosine similarity cannot classify.

7.2 Results and analysis

1) Effects of positive/total ratio in training set: Since our data is highly skewed and too many negative cases may not be good for classification, we thus performed this experiment to find a good ratio. Table 5 shows the results for Type I experiments. From Table 5, we can see that the results are highly accurate. Even for 100 userids, our method can correctly identify 85% cases. Here we use the data sets T*_Q9S10 and the decision method is ScoreSqSum, which produces the best result. The

results for Type II experiments (Table 6) are also accurate. In most cases, the values of accuracy are higher than 90%. For all our experiments below, we use the model/classifier trained with 0.4 ratio.

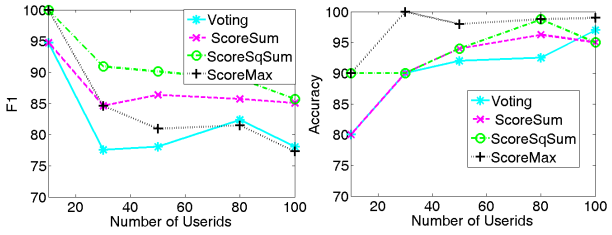
Table 5. Positive(p)/total(t) ratio in training (Type I)

	p/t	10	30	50	80	100
F1	0.3	**100.00**	84.62	86.36	**88.89**	83.72
	0.4	**100.00**	**91.91**	90.11	**88.89**	85.71
	0.5	**100.00**	90.91	**91.30**	**88.89**	87.01
	0.6	94.74	82.35	87.64	85.71	86.36
	0.7	94.74	84.62	86.36	86.53	**87.64**

Table 6. Positive(p)/total(t) ratio in training (Type II)

	p/t	10	30	50	80	100
Accuracy	0.3	90.00	90.00	92.00	97.50	94.00
	0.4	**90.00**	**90.00**	**94.00**	**98.75**	**95.00**
	0.5	80.00	86.67	94.00	97.75	95.00
	0.6	80.00	86.67	90.00	93.75	92.00
	0.7	80.00	86.67	90.00	95.00	92.00

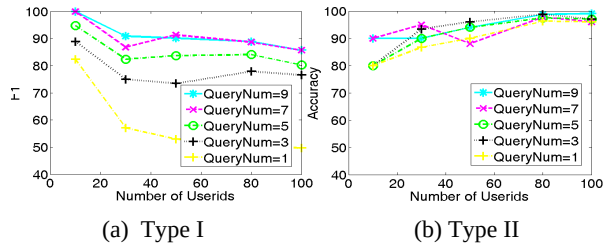
(2) Effects of different decision methods: We show the results of the four proposed decision methods: Voting, ScoreSum, ScoreSqSum, and ScoreMax, using our basic data of T*_Q9S10 with varied number of test userids. Figure 5(a) shows that ScoreSqSum is the best for Type I experiments. Figure 5(b) shows ScoreMax is the best for Type II, but ScoreSqSum also does very well. Below, ScoreSqSum is used as our default method because Type I is more important than Type II.



(a) Type I (b) Type II

Figure 5: Effect of different decision methods

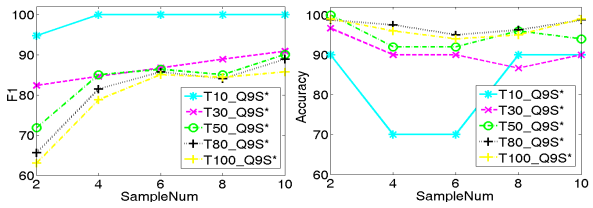
(3) Effects of number of queries per userid: Figure 6 shows the results of different numbers of queries. We see that more queries give better results, which is easy to understand because more queries give more information. We use 9 queries per userid in all other experiments.



(a) Type I (b) Type II

Figure 6: Effect of different numbers of queries

(4) Effects of number of samples per userid: We tried 2, 4, 6, 8, 10 samples per userid. Although there are some fluctuations for Type II (Fig.7(b)), we can see an upward trend for Type I in Fig. 7(a). This indicates that more sample documents give better results in general. The main reason again is that more samples from a userid give more identifying information about the userid. We use 10 test documents (samples) per userid in all experiments.



(a) Type I (b) Type II

Figure 7: Effect of different number of samples

(5) Impact of individual s-feature sets: Here we show the effectiveness of individual s-feature sets. From Table 7, we see that Sim7Retrieval s-features are extremely important for Type I test. Removing Sim7Retrieval causes about 10% to 20% F_1 score drop on different datasets. SimCTfidf s-features are also useful. The impacts of other s-features are small. The same applies to Type II test (Table 8). On average, using all features is the best. Hence we use all features in all other experiments above.

Table 7. Using different s-features (Type I)

T*_Q9S10	F1 10	F1 30	F1 50	F1 80	F1 100
All features	**100.00**	**90.91**	**90.11**	**88.89**	**85.71**
No Sim4Len	100.00	88.89	86.36	87.32	85.06
No SimCRichness	100.00	88.89	91.30	88.89	85.71
No SimCTfidf	100.00	80.00	86.36	86.53	83.72
No Sim7Retrieval	82.35	72.34	75.80	78.79	77.30
No Sim3Sent	94.74	84.62	86.36	88.11	87.64

Table 8. Using different s-features (Type II)

T*_Q9S10	Acc. 10	Acc 30	Acc. 50	Acc 80	Acc. 100
All features	**90.00**	**90.00**	**94.00**	**98.75**	**99.00**
No Sim4Len	90.00	93.33	96.00	96.25	96.00
No SimCRichness	90.00	90.00	94.00	96.25	96.00
No SimCTfidf	90.00	86.67	94.00	93.75	97.00
No Sim7Retrieval	80.00	90.00	94.00	94.00	96.00
No Sim3Sent	90.00	93.33	92.00	98.75	93.00

(6) Comparing with the three baselines: Similar to our method, the training data for TSL is highly skewed as it uses a *one-vs.-all* strategy. Hence we also investigate the effect of p/t ratio in training for TSL. Results show that 0.4 ratio is the best setting.

Thus this setting is adopted for TSL in the following experiments. Note that we cannot conduct p/t ratio experiments for SimAD and SimUG as they are unsupervised methods. We use ScoreMax for TSL, ScoreSqSum for SimUG and SimAD, respectively, since they perform the best for their corresponding approaches. Tables 9 and 10 show the results of our LSS method and the baseline methods for Type I and II tests respectively. For TSL, we use all *d*-features. Unigram features gave TSL much worse results and are thus not included here.

Table 9: Comparison with baselines (Type I)

		10	30	50	80	100
LSS	**Pre**	100.00	100.00	100.00	100.00	98.68
	Rec	100.00	83.33	82.00	80.00	75.76
	F1	**100.00**	**90.91**	**90.11**	**88.89**	**85.71**
TSL	**Pre**	50.00	50.00	33.33	0.00	0.00
	Rec	11.11	3.45	2.08	0.00	0.00
	F1	18.18	6.45	3.92	0.00	0.00
SimUG	**Pre**	100.00	100.00	100.00	100.00	100.00
	Rec	70.00	46.67	48.00	48.75	43.00
	F1	82.35	63.64	64.86	65.55	60.14
SimAD	**Pre**	100.00	75.00	100.00	33.33	0.00
	Rec	20.00	10.35	2.00	1.28	0.00
	F1	33.33	18.18	3.92	2.47	0.00

Table 10: Comparison with baselines (Type II)

Accuracy	**10**	**30**	**50**	**80**	**100**
LSS	90.00	90.00	94.00	98.75	95.00
TSL	90.00	96.67	98.00	98.75	99.00
SimUG	96.00	93.33	96.00	96.25	97.00
SimAD	90.00	96.67	98.00	98.75	99.00

From Tables 9 and 10, we can make the following observations.

- For Type I, F_1 scores of LSS are markedly better than those of the three baselines. The results of SimUG also drop more quickly than LSS with the increased number of userids. SimAD's results are extremely poor. These show that LSS is much more superior to the unsupervised methods. TSL performed the worst, indicating that traditional supervised learning is inappropriate for this task. There are two main reasons: First, for *one vs. all* learning, the negative training data actually contain positive documents which are written by the same author using another userid as the positive data, which confuses the classifier. Second, TSL is unable to build an accurate classifier using the small number of queries (which are training data). In contrast, our LSS method can exploit a large number of other authors who do not have to appear in testing and thus achieves the huge improvements.
- For Type II, LSS also performs very well. The baselines perform well too and even better, which is not surprising because they have difficulty in finding matching pairs for Type I. Since Type II datasets have no author with multiple userids, naturally the baselines will do well for Type II. But that is useless because when there are authors with multiple usersids (Type I), they are unable to find them well.

In summary, we can conclude that for Type I tests (there are authors with multiple userids), LSS is dramatically better than all baseline methods. For Type II tests (there is no author with multiple userids), it also performs very well.

8 Conclusion

This paper proposed a novel method to identify userids that may be from the same author. The core of the method is a supervised learning method which learns in a similarity space rather than the document space. This learning method is able to better determine whether a document may be written by a known author, although no document from the author has been used in training (as long as we have some documents from the author to serve as queries). To the best of our knowledge, there is no existing method based on linguistic analysis for solving the problem. Our experimental results based on a large number of reviewers and their reviews show that the proposed algorithm is highly accurate. It outperforms three baselines markedly.

Acknowledgements

We are grateful to the anonymous reviewers for their thoughtful comments. Tieyun Qian was supported in part by the NSFC Projects (61272275, 61272110, 61202036), and the 111 Project (B07037). Bing Liu was supported in part by a grant from National Science Foundation (NSF) under no. IIS-1111092.

References

Shlomo Argamon and Shlomo Levitan. 2004. Measuring the usefulness of function words for authorship attribution. *Literary and Linguistic Computing* 1-3.

Shlomo Argamon, Casey Whitelaw, Paul Chase, Sobhan Raj Hota, Navendu Garg, and Shlomo Levitan. 2007. Stylistic text classification using functional lexical features: Research articles. *J. Am. Soc. Inf. Sci. Technol*. 58:802-822.

John F. Burrows. 1992. Not unless you ask nicely: The interpretative nexus between analysis and information. *Literary and Linguistic Computing* 7:91-109.

Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting ranking svm to document retrieval. *Proc. of SIGIR*, Pages 186-193.

Hung-Ching Chen, Mark K. Goldberg, Malik Magdon-Ismail. 2004. Identifying multi-ID users in open forums. *Intelligence and Security Informatics*, Pages 176-186.

Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass, 2000. Authorship attribution with support vector machines. *Applied Intelligence* 19:109-123.

Hugo Jair Escalante, Thamar Solorio, and Manuel Montes-y-Gómez. 2011. Local histograms of character n-grams for authorship attribution. *Proc. of ACL-HLT*, Volume I: 288-298.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012. Distributional Footprints of Deceptive Product Reviews. *Proc. of ICWSM*.

Michael Gamon. 2004. Linguistic correlates of style: authorship classification with deep linguistic analysis features. *Proc. of Coling.*

Neil Graham, Graeme Hirst, and Bhaskara Marthi. 2005. Segmenting documents by stylistic character. *Natural Language Engineering*, 11:397-415.

Jack Grieve. 2007. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing* 22:251-270.

Hans van Halteren, Fiona Tweedie, and Harald Baayen. 1996. Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing* 11:121-132.

Steffen Hedegaard and Jakob Grue Simonsen. 2011. Lost in translation: authorship attribution using frame semantics. *Proc. of ACL-HLT, short papers* - Volume 2, 65-70.

Graeme Hirst and Ol'ga Feiguina. 2007. Bigrams of syntactic labels for authorship discrimination of short texts. *Literary and Linguistic Computing* 22:405-417.

David I. Holmes and R. S. Forsyth. 1995. The Federalist Revisited: New Directions in Authorship Attribution, *Literary and Linguistic Computing*, 10(2): 111-127.

David L. Hoover. 2001. Statistical stylistics and authorship attribution: an empirical investigation. *Literary and Linguistic Computing* 16:421-424.

Nitin Jindal and Bing Liu. 2008. Opinion Spam and Analysis. *Proc. of WSDM*, California, USA.

Thorsten Joachims. 2006. Training linear svms in linear time. *Proc. of KDD*.

Sangkyum Kim, Hyungsul Kim, Tim Weninger, Jiawei Han, and Hyun Duk Kim. 2011. Authorship classification: a discriminative syntactic tree mining approach. *Proc. of SIGIR*, Pages 455-464.

Dan Klein, and Christopher D. Manning. 2003. Accurate unlexicalized parsing. *Proc. of ACL*, 423-430.

Moshe Koppel and Jonathan Schler. 2004. Authorship verification as a one-class classification problem. *Proc. of ICML*.

Moshe Koppel, Jonathan Schler, Shlomo Argamon. 2011. Authorship attribution in the wild. *Lang Resources & Evaluation*, 45:83-94

Fangtao Li, Minlie Huang, Yi Yang and Xiaoyan Zhu. 2011. Learning to identify review Spam. *Proc. of IJCAI*.

Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool publishers.

Jiexun Li, Rong Zheng, and Hsinchun Chen. 2006. From fingerprint to writeprint. *Communications of the ACM*, 49:76-82.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, Hady W. Lauw. 2010. Detecting product review spammers using rating behaviors. *Proc. of CIKM*, 2010.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*, Morgan & Claypool publishers.

Tieyan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer.

Kim Luyckx, Walter Daelemans. 2008. Authorship Attribution and Verification with Many Authors and Limited Data. *Proc. of Coling*, pages 513-520.

David Madigan, Alexander Genkin, David D. Lewis, Shlomo Argamon, Dmitriy Fradkin, and Li Ye. 2005. Author Identification on the Large Scale. *Proc. of CSNA*.

Donald Metzler, Yaniv Bernstein, W. Bruce Croft, Alistair Moffat, and Justin Zobel. 2005. Similarity measures for tracking information flow. *Proc. of CIKM*. Pages 517-524.

Frederick Mosteller, David Lee Wallace. 1964. *Inference and disputed authorship: The Federalist*. Addison-Wesley.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting Fake Reviewer Groups in Consumer Reviews. *Proc. of WWW*, Pages 191-200.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, et al. 2012. On the feasibility of internet-scale author identification. *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. Pages 300-314

Jasmine Novak, Prabhakar Raghavan, Andrew Tomkins. 2004. Anti-aliasing on the web. *Proc. of WWW*, Pages 30-39

Myle Ott, Yejin Choi, Claire Cardie, Jeffrey T. Hancock. 2011. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. *Proc. of ACL*.

Myle Ott, Claire Cardie, Jeffrey T. Hancock. 2012. Estimating the prevalence of deception in online review communities. *Proc. of WWW*.

Fuchun Peng, Dale Schuurmans, Shaojun Wang, and Vlado Keselj. 2003. Language independent authorship attribution using character level language models. *Proc. of EACL*, Pages 267-274.

Conrad Sanderson and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, markov chains and author unmasking: an investigation. *Proc. of EMNLP*, Pages 482-491.

Yanir Seroussi, Fabian Bohnert, Ingrid Zukerman. 2012. Authorship Attribution with Author-aware Topic Models. *Proc. of ACL*, 2:264-269.

Thamar Solorio, Sangita Pillay, Sindhu Raghavan, Manuel Montes y G´omez. 2011. Modality Specific Meta Features for Authorship Attribution in Web Forum Posts. *Proc. of IJCNLP*, Pages 156-164.

Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538-556, Wiley.

Efstathios Stamatatos, George Kokkinakis, and Nikos Fakotakis. 2000. Automatic text categorization in terms of genre and author. *Comput. Linguist.* 26:471-495.

Özlem Uzuner and Boris Katz. 2005. A comparative study of language models for book and author recognition. *Proc. of IJCNLP*, Pages 969-980.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience, NY.

O. de Vel, A. Anderson, M. Corney and G. Mohay. 2001. Mining Email Content for Author Identification Forensics. *Sigmod Record*, 30:55-64.

Kyung-Hyan Yoo and Ulrike Gretzel. 2009. Comparison of Deceptive and Truthful Travel Reviews. *Information and Communication Technologies in Tourism*, Pages 37-47.

Georgy Udnv Yule. 1944. *The statistical study of literary vocabulary*. Cambridge University Press.

Guan Wang, Sihong Xie, Bing Liu, Philip S. Yu. 2011. Review Graph based Online Store Review Spammer Detection. *Proc. of ICDM*.

Ying Zhao and Justin Zobel. 2005. Effective and scalable authorship attribution using function words. *Proceeding of Information Retrival Technology*, Pages 174-189.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing style features and classification techniques. *Journal of the American Society of Information Science and Technology* 57:378-393.

Gender Inference of Twitter Users in Non-English Contexts

Morgane Ciot
School of Computer Science
McGill University
Montreal, Quebec, Canada
morgane.ciot@mail.mcgill.ca

Morgan Sonderegger
Department of Linguistics
McGill University
Montreal, Quebec, Canada
morgan.sonderegger@mcgill.ca

Derek Ruths
School of Computer Science
McGill University
Montreal, Quebec, Canada
derek.ruths@mcgill.ca

Abstract

While much work has considered the problem of latent attribute inference for users of social media such as Twitter, little has been done on non-English-based content and users. Here, we conduct the first assessment of latent attribute inference in languages beyond English, focusing on gender inference. We find that the gender inference problem in quite diverse languages can be addressed using existing machinery. Further, accuracy gains can be made by taking language-specific features into account. We identify languages with complex orthography, such as Japanese, as difficult for existing methods, suggesting a valuable direction for future research.

1 Introduction

A 2012 study reported that US-based Twitter users now account for only 28% of all active accounts on the platform (Semiocast, 2012). Brazil, Japan, India, and Indonesia all rank in the top 10, each with over 5% of all users. These and other findings confirm that Twitter enjoys widespread international popularity and usage. This is also reflected in the multinational community of researchers who study human behavior on Twitter and related platforms, e.g. (Sakaki et al., 2010; Tumasjan et al., 2010; Kim and Park, 2012).

It is remarkable, then, that advances in latent attribute inference on social media have been largely confined to English content, e.g. (Liu and Ruths, 2013; Zamal et al., 2012; Pennacchiotti and Popescu, 2011; Conover et al., 2011a). This bias may be partially explained in the context of the research being conducted largely by anglophone researchers. Nonetheless, it has created a notable silence in the literature concerning the large-scale analysis of languages, cultures, and people on social media who do not employ English.

In this paper, we examine the problem of latent attribute inference outside the English-language context. To our knowledge, this is the first such study ever conducted. Here we specifically focus on gender inference, as it has been the basis for significant work in recent years (Liu et al., 2012; Zamal et al., 2012; Pennacchiotti and Popescu, 2011; Rao et al., 2010; Burger et al., 2011). Our work makes two contributions. First, we quantify the extent to which established gender inference methods can be used with non-English Twitter content. Second, we explore the capacity for unique features of other languages (besides English) to improve inference accuracy. This second aspect, in particular, acknowledges the fact that latent attribute inference may be easier in some languages due not to conventions in word usage, but to syntactic structure.

In order to assess the extent to which existing gender inference machinery works for users who use languages other than English, we assembled Twitter datasets for languages that are both prevalent on Twitter and representative of diverse language families: Japanese, Indonesian, Turkish, and French. Each dataset consisted of approximately 1000 users who tweeted primarily in a given language. We used Amazon Mechanical Turk to manually label each user with their gender, using a language-agnostic labeling strategy (Liu and Ruths, 2013). For classification, we employed a performant support vector machine-based (SVM) technique that has been used in a range of studies, e.g. (Rao et al., 2010; Burger et al., 2011; Zamal et al., 2012).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1136–1145,
Seattle, Washington, USA, 18-21 October 2013.

We found that, without any modification to the types of features given to the SVM, the classifier accuracy was comparable on English, French, and Indonesian. Turkish actually performed much better, achieving 87% on average. Gender in Japanese, in contrast, could not be reliably inferred with any reasonable accuracy (61% on average) despite numerous attempts to preprocess the tweets and tune the classifier to accommodate the language's complex orthography. This indicates that existing approaches may not generalize well to language systems with thousands of distinct unigrams (as opposed to tens or hundreds in the other languages considered).[1]

To evaluate the extent to which language-specific features might be used to boost the accuracy of the SVM classifier further, we focused on French. French is a valuable case study because, unlike English, it has a number of syntax-based mechanisms that can encode the gender of the speaker. The most common instantiation of gender marking is the modification of adjective and some past participle endings to match the gender of the subject in constructions beginning with "je suis" (trans. "I am") constructions. A classifier based on this insight achieved average accuracy of 90% on the vast majority of French users, surpassing the accuracy of standard techniques on English or French.

Overall, our results show that, with little modification, existing gender inference machinery can perform comparably to English on several other languages. There are clear areas for substantial improvement: incorporating language-specific features and, in the case of Japanese, finding better ways of accommodating the complex orthography. These findings identify promising directions for future research and will, hopefully, call attention to an important area in the latent attribute inference domain in need of further work.

2 Background

Non-English Twitter data mining and studies. Existing work on non-English Twitter content can been divided into two groups: surveys of the use of several languages on the platform and studies of a social phenomenon in a non-English body of tweets. To our knowledge, only a handful of the former variety exist. One recent paper characterized the relationship between language and geography (Mocanu et al., 2012). Another measured how high-level tweet features (i.e., link, mention, and hashtag frequencies) vary across languages (Weerkamp et al., 2011). These papers show that tweet structure and content can differ widely across languages.

More work has been done in the latter category: analysis of social phenomena in a non-English context. A well-known study evaluated usage of Twitter in the aftermath of the 2010 earthquake in Japan (Sakaki et al., 2010). Another Japanese-oriented study evaluated the impact of television on tweeted content (Akioka et al., 2010). Within this category, another recurring research topic is the analysis of political discussion and elections. Outside of English-based analysis, some attention has been given to European and East Asian elections, e.g. (Tumasjan et al., 2010; Giglietto, 2012; Kim and Park, 2012). However, few of these studies have considered measures beyond simple hashtag frequencies, relative mention counts among politicians, and retweet counts. The only study using more complex features for computational text analysis involved sentiment analysis of a set of German tweets (Tumasjan et al., 2010). However, the tweets in this study (conducted at a German university) were translated into English prior to analysis, a step which underscores the significant bias towards English in the literature on analyzing microtext, and the tools available to researchers in this domain.

Gender inference methods. Gender inference is a field of research situated with the broader area of latent attribute inference. The majority of recent work in this area has focused on Twitter users (Rao et al., 2010; Pennacchiotti and Popescu, 2011; Conover et al., 2011b; Burger et al., 2011; Rao and Yarowsky, 2010; Liu and Ruths, 2013; Liu et al., 2012; Zamal et al., 2012). Classifiers have been built, predominantly, using support vector machines, e.g. (Rao et al., 2010; Pennacchiotti and Popescu, 2011; Burger et al., 2011; Zamal et al., 2012), though boosted decision trees and latent dirichlet allocation systems have also been evaluated, e.g. (Pennacchiotti and Popescu, 2011; Conover et al., 2011b). With one exception, gender inference accuracy has been re-

[1] In this work, *unigram*, *bigram*, and *k-gram* refer to one, two, and k-character sequences in a language's written form.

ported between 80% and 85%. The one study which reported 90% accuracy involved the use of a dataset which has been shown to be quite different from typical anglophone Twitter users (Burger et al., 2011). This same study did involve non-English Twitter users, but did not analyze the performance of the classifier on different languages (e.g. break down performance by language, examine to what extent its results were due to better performance on some languages), or indeed discuss fully which languages were present in their sample. Thus, little can be inferred from Burger et al.'s study about the relative performance of attribute inference methods on different languages, which is the focus of our paper.

Language families. Human languages can be classified into different *language families*, defined as a set of languages which are all descended from a single, ancient parent language. Languages which are genetically related (in the same family), however distantly, tend to share many more characteristics than languages from different families.

Each language considered in this paper belongs to a different language family: French to Indo-European, Turkish to Altaic, Japanese to Japonic, and Indonesian to Austronesian. Thus, these languages are completely genetically unrelated, by definition. Further, they are both geographically and culturally dispersed. While they all have some loanwords from English, these constitute a tiny fraction of each language's vocabulary. This selection of languages allows us to conduct the most far-reaching survey of non-English latent attribute inference performance to date.

A variety of features make each language selected interesting within the gender inference context. French is noteworthy for its grammatical gender. All nouns, including people, are grammatically "male" or "female." English, in contrast, has separate pronouns for people of different genders (e.g., "he", "she"), but does not have *grammatical* gender. (Besides a handful of exceptions like "waiter"/"waitress", there are no words besides pronouns which have different "masculine" and "feminine" forms.) Indonesian, Turkish, and Japanese are all so-called *genderless languages*. Like many languages of the world, they do not have distinct male and female pronouns (like English and French), or grammatical gender (like French).

3 General Gender Inference

In order to evaluate the extent to which existing gender inference machinery can be used on users whose tweets are in languages other than English, we developed gender-labeled datasets of Twitter users for each language and then evaluated the performance of a classifier on each.

3.1 Data

The core data for this project consisted of four datasets of content from Twitter users who tweeted predominantly in one of four languages—French, Indonesian, Turkish, and Japanese—collected using the methods described below.

Data collection. In order to identify users for candidate inclusion in a particular language's (hereafter the *target language*) dataset, we walked the streaming output of the Twitter firehose and evaluated the language of each tweet using the language models provided by the Natural Language Toolkit (Bird, 2006). Users associated with tweets written in the target language were added to a list. 5000 such users were identified for each language. The latest 1000 tweets for each user were downloaded. This comprised the base for the target language dataset.

Assigning gender labels. In prior work, e.g. (Rao et al., 2010; Pennacchiotti and Popescu, 2011; Zamal et al., 2012), the dominant way of obtaining datasets consisting of Twitter users with high-confidence gender-labels is to use gender-name associations. The use of name-gender associations are problematic when non-English content is considered because databases of anglophone name-gender associations are no longer useful (Mislove et al., 2011). We instead used Amazon Mechanical Turk workers to identify the gender of the person shown in the profile picture associated with a user's account (Liu and Ruths, 2013). In our datasets, each user's profile picture was coded by 5 separate workers. Users with non-photographic or celebrity-based profile pictures was discarded, as well as any users with profile pictures where the gender could not be confidently assessed (less than 4 out of 5 votes for one gender).

Table 1 shows the final composition of each dataset. In Japanese and Indonesian, we observed

Table 1: The composition of the different language datasets used in this study.

Language	# Males	# Females	Total Size
French	437	506	943
Indonesian	977	2260	3237
Turkish	1672	1937	3609
Japanese	309	520	829

a notable difference in the number of males and females in the dataset. Measures were taken to ensure that classifier results were not biased by these differences within the datasets.

3.2 Methods

The majority of prior work in gender inference (and latent inference in general) has used support vector machines (SVMs). We followed prior work in this regard, particularly since our intent here is to evaluate the relevance of existing gender inference machinery on other languages. For the present study, we adopted an SVM-based classifier, described in (Zamal et al., 2012), that incorporated nearly all features used in prior work and showed comparable (and sometimes better) accuracy than other methods. Parameter values and kernel choices for the SVM are discussed in the source paper.

Feature set. SVM classifiers require that each object to be classified be represented by a fixed-length feature vector. The features we employed were: k-top words, k-top digrams and trigrams, k-top hashtags, k-top mentions, tweet/retweet/hashtag/link/mention frequencies, and out/in-neighborhood size. Note that "*k-top X* features" (e.g., k-top hashtags) refers to the k most discriminating items of that type for each label (i.e., Male/Female). Thus, *k-top words* is actually $2k$ features: the k words most associated with males and the k words most associated with females.

This list of features is the same set of features used in (Zamal et al., 2012), except that *k-top stems* and *k-top co-stems* were both dropped in our version. Both of these feature types are specific to English. Of course, word stems do exist in other languages, however we found that stemmers (the algorithms that identify and extract the appropriate stem from a word) were not available across the whole bank of languages. Therefore, we omitted these stem and co-stem features. We also added features for the usage frequencies of Eastern-style and Western-style emoticons but saw no discernible change in accuracy; thus, these features are not discussed further.

It is important to note that all features included in our classifier are language-agnostic. An n-gram is simply an n-character sequence drawn from the alphabet and additional symbols (numbers, punctuation, etc.) present in tweets written in the target language. Words are sequences of characters that are bounded by whitespace or punctuation. Hashtags are words proceeded by a pound ("#") character, mentions by an "@" symbol. A system that properly supports unicode strings can implement all of these notions without knowing anything about the target language it is operating on.

Tokenization of Japanese. While all the definitions provided above for the SVM features are *operational*, there is a glaring disconnect between the whitespace-border definition of a word and written conventions in Japanese. Specifically, in Japanese words are generally not separated by whitespace.

We used a tokenizer to insert whitespace into Japanese text to break up words. Tokenization was done using Kuromoji, the software Japanese morphological analyzer used and supported by the Apache software Foundation (Atilika, 2012). Notably, this tool tokenizes the mixed character sets that are often used in informal Japanese writing.

As tokenization does involves some language-specific processing, its use here somewhat undermines the objective set out for this project. Thus, we report the accuracy achieved for both untokenized and tokenized Japanese tweets. Curiously, tokenization was found to not make a difference in overall average accuracy.

3.3 Results

For each dataset, 5-fold cross validation was used to assess the classifier's performance. The value of $k = 20$ was used for all k-top features, though the results reported are robust to changes of this value within reason (between 10 and 30). If the numbers of male and female users were unbalanced in a dataset, the larger set was subsampled randomly to obtain a set of users the same size as the smaller labeled set. During the training process, the actual

Table 2: The accuracy of the SVM-based classifier on each of the language datasets. In the case of Japanese, the performance is given for both the tokenized and untokenized versions of the dataset. (Note that tokenization did not affect overall accuracy.)

Language	Male	Female	Overall
French	0.79	0.73	0.76
Indonesian	0.87	0.80	0.83
Turkish	0.89	0.85	0.87
Japanese (t)	0.50	0.76	0.63
Japanese (u)	0.58	0.68	0.63

values of the features were extracted from the training users (e.g., the k-top differentiating words for males and females were identified). In this way, the gender model implemented by the SVM *was* language-specific, in the sense that a particular language's gender model contained a different set of features. We call our method language-agnostic on the grounds that, given a labeled set of users and tweets drawn from a particular language, a model can be built without any knowledge of the structure or content of the language itself.

Tables in Supplementary Material show the features for the classifier built over each language's entire dataset. Note that to conduct the cross-fold evaluation, new models (and hence different features) were recomputed for each fold. As a result, the features reported are slightly different from those that might have appeared in the models for a given fold. Manual inspection, however, revealed that differences were slight. The features reported in the Supplementary Material can be safely considered a consensus among the models for the individual folds.

The accuracy of the classifier for each language is shown in Table 2. Overall, the classifier demonstrated good performance on all languages except for Japanese. Below, we consider the results for each of the four languages in turn. In each case we discuss language-specific trends in which words were most informative for inferring user gender, and thus help explain the classifier's performance. Throughout, we omit discussion of non-alphanumeric "words" (such as punctuation or emoticons), and call the k-top discriminating words for male and female users the *k-top male words* and *k-top female words*.

French. The k-top words for men and women are of very different grammatical types. Most male words are prepositions or articles (16/25; e.g. *de* 'of', *un* 'a/one'); a few others are basic grammatical words (*ne* '[part of] not', *et* 'and'), or pronouns or verb forms referring to a single person or object (he/she/it), as well as one noun (*France*). In contrast, many female words (11/25) are pronouns or basic verb forms referring to the speaker or a single addressee (*je* 'I', *mon/mes/ma* 'my', *tu* 'you', *j'ai* 'I have'). Others are pronouns or basic verbs refer to a single person or object (*elle*, 'she/it', *c'est* 'it's'), as well as a few other frequent words (*trop* 'too much', *pas* '[part of] not', *oui* 'yes'). The most salient pattern is that use of words (pronouns, basic verbs) associated with talking about the speaker or addressee indicates a tweet is more likely to be from a female user. Heavy use of other common function words, specifically prepositions and articles, suggests a male user. These patterns reflect known gender differences in word usage by male and female French speakers (Witsen, 1981).

Indonesian. Indonesian achieved performance closest to the inference accuracy for English reported in the literature. The k-top lists for men and women give some justification for why the classifier performed well. Some differences can be tentatively linked to general trends in how men and women use language differently across cultures. 5/25 of men's k-top words are nouns which are either related to soccer (*vs* 'versus', *chelsea* '[name of UK soccer team]', *pemain* 'player') or which could be related to soccer (*jakarta*, *indonesia*, *malam* 'night'); in contrast, no women's words are nouns. It seems plausible that men tweet about soccer significantly more than women. In such a situation, a reasonable concern is that our classifier discriminated soccer from non-soccer enthusiasts rather than males from females. To address this, we confirmed that these topic-based words were not required for accurate classification: a classifier in which soccer words were explicitly removed performed just as well (83.8% vs. 83.3%).

More interestingly, many of the k-top words correspond to men and women using different terms of address and self-reference. Among the k-top words, 7/25 for men and 4/25 for women are terms

of address or self-reference. The terms men use are mostly highly informal, including the slang term *lu* (you) and the English borrowing *bro*; the address terms women use are mostly medium-formality, such as *aku* (I) and *kamu* (you). Thus, women seem to be using "more polite" self-reference and address terms than men on average on Indonesian Twitter, in line with the more general tendency for women to use polite forms more frequently than men cross-culturally (Holmes, 1995).

Turkish. Turkish achieved notably high accuracy: the highest of all four languages considered. In fact, to our knowledge, this is the highest accuracy achieved in the entire Twitter gender inference literature on a dataset drawn from the Twitter general population. The k-top lists of male and female words again give some justification for the classifier's performance. Many differences between the male and female lists can be linked to men and women talking about different topics, or to different people. Several of the male words refer to soccer (*gol* 'goal', *galatasaray* 'popular Istanbul team', *maç* 'match', *at* '[part of imperative for] score'), which men plausibly tweet about more. As with Indonesian, a concern is that topics represent a biased sample of the population. Thus, we tested a classifier with soccer-specific terms removed, and again found no difference in accuracy (86% vs. 87%). Many other k-top words are familiar terms of address for men (*lan*, *abi*, *karde sim*, *adam*, *kanka*) or a greeting used mainly between men (*eyvallah*), suggesting that male users are addressing or discussing men more often than female users are. In contrast, 9/25 of the k-top female words are pronouns referring to the speaker, a familiar addressee, or a third party (he/she/it), while none of the k-top male words are, suggesting female users are more often talking directly about themselves or to others. Finally, 2/25 of the k-top male words are profanity (*amk*, *ulan*), while none of the female k-top words are, suggesting male users swear more.

Japanese. Beyond the Japanese classifier's generally poor accuracy, it is striking that tokenization did not improve overall accuracy. This indicates that once words were properly tokenized, no additional gender-distinguishing signal could be extracted. This may be an indication that word-based features carry little information in languages with complex orthography, such as Japanese (with many thousands of unigrams).

Despite the classifier's poor performance, the k-top discriminating words for male and female users differ in interesting ways. Some differences can be understood as resulting from known general trends in how Japanese men and women's use language. Japanese speakers have a choice of many first-person singular pronouns (equivalent to "I"), which signal different levels of politeness and of male versus female speech. The pronoun *boku* (僕) is associated with informal male speech; accordingly, it is among the k-top male words. Japanese also uses an extensive system of verb forms corresponding to different levels of politeness, and honorifics (affixes for names used when referring to others). Women tend to use polite verb forms and honorifics more frequently than men in Japanese speech (Peng, 1981). In agreement with this pattern, several polite verb forms (*-masu*, *-mashi*) and a polite honorific (*o-*) are among the k-top female words, as is a diminutive honorific often used to refer to women (*-chan*).

4 Language-specific Features and Inference

While the classifier performed well across a diverse set of languages, recall that all features used by the SVM were language-agnostic. A natural question concerns the extent to which language-specific features relevant to the attribute of interest (e.g., gender) might improve the classifier's performance.

We examine this question within the context of French. Where gender inference is concerned, French is quite interesting because information about the gender of nouns (including the speaker) is often obligatorily marked in the syntax: many words have different 'masculine' and 'feminine' forms for referring to male and female nouns, including the speaker. Thus, it is in principle often possible to infer the gender of the speaker by which form they use, although it is not clear a priori that this method will work for Twitter data.

4.1 Method

French grammar dictates that which forms of words are used often reflects the gender of the speaker.

Adjectives and past participles all have masculine and feminine forms, which are often spelled differently, and in addition often pronounced differently.Adjectives must agree in gender with the noun they refer to. For example, "I am happy" would be *je suis heureuse* for a female speaker and *je suis heureux* for a male speaker (literally "I-am-happy"); *heureuse* and *heureux* are the feminine and masculine singular forms of the adjective, and are pronounced differently. Past participles of verbs also agree with the gender of the subject or object of the verb, for certain verbs and constructions. For example, "I went" would be *je suis allée* for a female speaker and *je suis allé* for a male speaker (here *suis* is used to form the simple past of the verb *aller*, 'to go'); *allé* and *allée* are the masculine and feminine forms of the past participle of *aller*, and are pronounced the same.

Note that the phrase *je suis* ("I am") occurs in both the adjectival and verbal constructions referring to the speaker; however, the function of *suis* differs between the two. *suis* is the first-person singular form of the verb *être* ("to be"), and functions as a copula when followed by an adjective ("I am happy") but as an auxiliary verb to mark the past tense, when followed by the past participle of certain verbs ("I went"). For our purposes, what is important is that, in both cases, a following adjective or past participle will take on the gender of the speaker.

When this construction occurs in a tweet, it is likely that *je* is referring to the author of the tweet, and the rules of French grammar dictate that the gender of the associated adjective or past participle should reflect the gender of the tweet's author. We implemented a classifier that used this logic to classify the gender of francophone Twitter users. It is worth emphasizing that the *existence* of adjectives and participles which reflect the speaker's gender does not automatically make gender identification in French tweets a trivial task. First, given the prevalence of non-standard spelling and grammar on Twitter and other online platforms, French users may sometimes not use the 'correct' gender marked form reflecting their actual gender—especially given that the male and female forms for a given adjective or participle are often pronounced the same. Second, even if gender-marked constructions are used correctly, they may not occur sufficiently often in tweets to be a reliably used for speaker gender identification. Both of these concerns are borne out in our French dataset, as described further below; the question addressed in the experiment is how useful the signal provided by gender-marked forms is, despite these two sources of noise.

Table 3: The set of patterns that were considered to be suis-constructions when encountered in a tweet.

jn suis pas, jm suis, jmsuis, jnmsuis pas, jnsuis pas, je ne suis pas, je suis pas, jsuis, jensuis pas, jemsuis, jnesuis pas, jmesuis, je me suis, je ne me suis pas

Unlike the probabilistic SVM classifier, the suis-construction classifier can be made entirely deterministic. For a given user, the set of tweets containing a suis-construction are identified, $T_{suis}(u)$. Of these, we can identify the number of those tweets that involve an adjective or past participle with a female ending $T^F_{suis}(u) \subseteq T_{suis}(u)$. Labeling a user involves selecting a threshold based on $T^F_{suis}(u)$ and $T_{suis}(u)$ below which a user receives one label and above which the user receives the other label.

Detecting suis-constructions. As expected, cursory inspection of tweets revealed that Twitter users often employed shorthand forms of the suis-construction. We accounted for this by conducting a manual survey of the shorthand forms of the suis-construction. A catalog of regular expressions was drawn up that matched the different suis-construction forms we identified, shown in Table 3.

Recognizing the gender of the adjective or past participle involved in a suis-construction required a second processing stage. The Lexique lexical database was used to tag the word trailing the suis-construction (New and Landing, 2012). If the tag was not an adjective or verb, the construction was discarded as it would not contain a gender indication. If the word was recognized as an adjective or verb, Lexique would also return the gender, which would be returned as the gender indication for that particular suis-construction.

Threshold selection. We evaluated a number of policies for assigning the user's gender based on the relative values of $T^F_{suis}(u)$ and $T_{suis}(u)$. In the end, however, the best performing threshold was $T^F_{suis}(u) > 1$: simply labeling as female any user

Table 4: The component-wise and overall accuracy of the combined suis-construction and SVM classifier.

Component	# users	Male Acc	Female Acc	Overall Acc
suis-const.	723	0.91	0.90	0.90
SVM	220	0.70	0.54	0.62
Overall	943	0.86	0.82	0.83

who employed the female construction even once. This threshold makes sense given the plausible intuition that females will (almost always) be the only users to employ a female suis-construction; however, it is quite sensitive to uses of female suis-constructions by males.

Mixed classifier. Since not all users had tweets which contained suis-constructions, we combined the SVM-based classifier used previously with the suis-construction-based classifier. The SVM component was applied to any users who lacked suis-constructions entirely in their tweet history. Any user who used even one suis-construction would be labeled according to the $T^F_{suis}(u) > 1$ threshold.

4.2 Results

We ran our classifier on the French dataset, obtaining the results shown in Table 4.

Coverage of the suis-construction. In spite of our concerns over the occurrence frequency and detectability of the suis-construction in tweets, our results show that suis-constructions were found in tweets belonging to nearly 75% of all users in the dataset. This suggests that the suis-construction classifier has quite broad coverage of the population. Of course, given the essential role of the verb "être" in French (like the role of "to be" in English), its frequent use is expected. Nonetheless, the flexible use of grammar and spelling in Twitter and other online contexts raised a genuine concern that occurrences of the suis-construction might not be detected. In fact, when we looked through the tweets of users who were flagged as not having useful suis-constructions in their tweets, we discovered that many actually did. The issue was that they employed highly irregular spellings that our implementation was not able to pick up. Thus, with additional refinement, it may be possible to improve the suis-construction coverage further, well beyond 80%.

Performance of the suis-construction classifier. On the set of users for which the suis-construction was detected, the classifier did very well, achieving an average accuracy of 90%. Recall that the threshold used to generate the results in Table 4 was $T^F_{suis}(u) > 1$. We tested other (larger) thresholds and found that the performance of the method dramatically and monotonically decreased. This was largely due to female users being misclassified as males, indicating that females do not exclusively use female suis-constructions (this was confirmed via manual inspection of a number of female tweet histories). This is different from males, most of whom are quite strict about using only male suis-constructions. Since forming the female form of an adjective or participle typically requires adding an additional character (or more) to the base of the word, this may reflect a tendency towards dropping gender modifiers in favor of typing less.

Performance of the SVM classifier. While the suis-construction classifier performed well, the SVM component did not do nearly as well on the Twitter users that could not be labeled using the suis-construction, achieving an average performance of 62%. At this level of accuracy, the classifier is performing barely better than a random classifier, which would have achieved around 50% accuracy on the label-balanced testing data. This result stands in opposition to our earlier finding that French users could be labeled with 75% accuracy. This disparity suggests that the non-suis-construction users comprise a particularly difficult-to-classify group.

The suis-construction as a filter. The finding that the SVM classifier performed poorly in the combined classification setting suggests that the suis-construction classifier is acting as a very effective filter for users that are hard for it to classify. While we might have preferred better classification accuracy all around, this result is still interesting and useful. Such filters can decrease classification error by simply flagging those users who cannot be easily classified, leaving them to be handled more carefully by more powerful classifiers or human coding. This is precisely the function that the suis-construction classifier appears to play (in addition to classifying the

other users).

This result suggests a question for future work: whether it is possible to build classifiers that accurately label the sets of users that are discarded by the suis-construction classifier.

Performance of the combined classifier. Despite the relatively poor performance of the SVM component, the accuracy of the combined classifier improved on the original SVM-only classifier by 8%, which is a substantial increase in accuracy. With some additional focus on classifying the difficult users who could not be labeled by suis-construction usage, we feel that this accuracy can be increased upwards of 90%.

5 Discussion

In this project, we have extended, for the first time, the latent attribute inference problem to users who tweet primarily in languages other than English. Our study offers several notable insights.

Existing approaches generalize. While accuracy levels certainly vary across languages, overall an existing SVM-based classifier, when trained on users from a given language, can classify the gender of other users from that same language with accuracy comparable to performance reported for English. We suspect that this result will generalize to the inference of other demographic characteristics (e.g., age and political orientation), though this must be explored in future work.

Complex orthography creates unique issues. Japanese stands out as being utterly unclassifiable using existing SVM-based approaches and feature sets. Even efforts to bridge some of the orthographic disconnects between the Japanese language and the assumptions made by the SVM failed to improve performance. This stands out as a clear direction for future work, particularly since apparent issues with the large number of unigrams used by Japanese will create issues for handling (Mandarin) Chinese, the world's most-spoken language.

Language-specific features boost performance. While unsurprising that customizing a classifier to the peculiarities of a given language boosts performance, our use of the suis-construction in French highlights how particular linguistic features may be uniquely well suited to the inference of particular attributes. The results obtained for French stand in contrast to various, relatively unsuccessful attempts to boost gender inference by incorporating syntactic features of English into the classifier (e.g., using stems and co-stems). It seems that some languages have features better suited for certain classification tasks. Identifying and leveraging such features will be an interesting and fruitful direction for future work.

Classifiers as a linguist's tool. In each language, a number of the k-top words align with or suggest gender-specific conventions in that particular language. That a language-agnostic classifier provided such insights highlights its potential for exploring language-specific word usage patterns and nuances. For example, sociolinguistics (a subfield of linguistics) has long studied the different ways men and women use language, especially in spontaneous speech (Eckert and McConnell-Ginet, 2003); recent work has begun to examine how language is used differently by men and women online as well (Bamman et al., 2012). Such studies could be radically scaled up in terms of the number of languages considered using a language-agnostic gender classifier.

6 Conclusion

Though there has been relatively little investigation into latent attribute inference outside of English-language content, we consider it both a fruitful and important area for future research. Here, we have evaluated the capacity for existing inference methods to be used outside their intended English-language context. Furthermore, we have shown how language-specific features might be incorporated in order to boost classifier accuracy further. The positive results suggest that latent attribute inference in the non-English context as a research direction worthy of further attention.

7 Acknowledgements

The authors gratefully acknowledge three anonymous reviewers whose feedback improved the clarity and correctness of the manuscript. The study was supported by grants from the Social Sciences

and Humanities and Natural Sciences and Engineering Research Councils of Canada (SSHRC Insight Grant #435-2012-1802 and NSERC Discovery Grant #125517855) and the Public Safety Canada Kanishka Program.

References

S. Akioka, N. Kato, Y. Muraoka, and H. Yamana. 2010. Cross-media impact on Twitter in Japan. In *Proceedings of the International Workshop on Search and Mining User-generated Contents*.

Atilika. 2012. Kuromoji morphological analyzer. http://www.atilika.org.

D. Bamman, J. Eisenstein, and T. Schnoebelen. 2012. Gender in Twitter: Styles, stances, and social networks. arXiv preprint arXiv:1210.4567.

S Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL Interactive Presentation Sessions*.

J.D. Burger, J. Henderson, and G. Zarrella. 2011. Discriminating gender on Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

M. Conover, B. Gonçalves, J. Ratkiewicz, A. Flammini, and F. Menczer. 2011a. Predicting the politial alignment of Twitter users. In *Proceedings of the International Conference on Social Computing*.

M.D. Conover, J. Ratkiewicz, M. Francisco, B. Gonçalves, F Menczer, and A Flammini. 2011b. Political polarization on Twitter. In *Proceedings of the International Conference on Weblogs and Social Media*.

P. Eckert and S. McConnell-Ginet. 2003. *Language and gender*. Cambridge University Press, Cambridge.

F. Giglietto. 2012. If likes were votes: An empirical study of the 2011 Italian administrative elections. In *Proceedings of the International Conference on Weblogs and Social Media*.

J. Holmes. 1995. *Women, men and politeness*. Longman, London.

M. Kim and H.W. Park. 2012. e-measuring Twitter-based political participation and deliberation in the South Korean context by using social network and Triple Helix indicators. *Scientometrics*, 90(1):121–140.

W. Liu and D. Ruths. 2013. What's in a name? Using first names as features for gender inference in Twitter. In *Analyzing Microtext: 2013 AAAI Spring Symposium*.

W. Liu, F.A. Zamal, and D. Ruths. 2012. Using social media to infer gender composition from commuter populations. In *Proceedings of the When the City Meets the Citizen Worksop*.

A. Mislove, S. Lehmann, Y.Y. Ahn, J.P. Onnela, and J.N. Rosenquist. 2011. Understanding the demographics of Twitter users. In *Proceedings of the International Conference on Weblogs and Social Media*.

D. Mocanu, A. Baronchelli, B. Gonçalves, N. Perra, and A. Vespignani. 2012. The Twitter of Babel: Mapping world languages through microblogging platforms. *ArXiv e-prints*, December.

B. New and C. Landing. 2012. Lexique 3. http://www.lexique.org/telLexique.php.

F.C.C. Peng, editor. 1981. *Male/female differences in Japanese*. The East-West Sign Language Association, Tokyo.

M. Pennacchiotti and A.M. Popescu. 2011. A machine learning approach to Twitter user classification. In *Proceedings of the International Conference on Weblogs and Social Media*.

D. Rao and D. Yarowsky. 2010. Detecting latent user properties in social media. In *Proceedings of the NIPS workshop on Machine Learning for Social Networks*.

D. Rao, D. Yarowsky, A. Shreevats, and M. Gupta. 2010. Classifying latent user attributes in Twitter. In *Proceedings of the International Workshop on Search and Mining User-generated Contents*.

T. Sakaki, M. Okazaki, and Y. Matsuo. 2010. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the International World Wide Web Conference*.

Semiocast. 2012. Brazil becomes the 2nd country on Twitter, Japan 3rd, Netherlands most active country. http://semiocast.com/publications/2012_01_31_Brazil_becomes_2nd_country_on_Twitter_superseds_Japan.

A. Tumasjan, T.O. Sprenger, P.G. Sandner, and I.M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the International Conference on Weblogs and Social Media*.

W. Weerkamp, S. Carter, and M. Tsagkias. 2011. How people use Twitter in different languages. In *Proceedings of the Web Science Conference*.

R. Schenk-Van Witsen. 1981. Les différences sexuelles dans le français parlé: une étude-pilote des différences lexicales entre hommes et femmes. *Langage et société*, 17(1):59–78.

F.A. Zamal, W. Liu, and D. Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. In *Proceedings of the International Conference on Weblogs and Social Media*.

A Multimodal LDA Model Integrating Textual, Cognitive and Visual Modalities

Stephen Roller
Department of Computer Science
The University of Texas at Austin
roller@cs.utexas.edu

Sabine Schulte im Walde
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
schulte@ims.uni-stuttgart.de

Abstract

Recent investigations into grounded models of language have shown that holistic views of language and perception can provide higher performance than independent views. In this work, we improve a two-dimensional multimodal version of Latent Dirichlet Allocation (Andrews et al., 2009) in various ways. (1) We outperform text-only models in two different evaluations, and demonstrate that low-level visual features are directly compatible with the existing model. (2) We present a novel way to integrate visual features into the LDA model using unsupervised clusters of images. The clusters are directly interpretable and improve on our evaluation tasks. (3) We provide two novel ways to extend the bimodal models to support three or more modalities. We find that the three-, four-, and five-dimensional models significantly outperform models using only one or two modalities, and that nontextual modalities each provide separate, disjoint knowledge that cannot be forced into a shared, latent structure.

1 Introduction

In recent years, an increasing body of work has been devoted to multimodal or "grounded" models of language where semantic representations of words are extended to include perceptual information. The underlying hypothesis is that the meanings of words are explicitly tied to our perception and understanding of the world around us, and textual-information alone is insufficient for a complete understanding of language.

The *language grounding problem* has come in many different flavors with just as many different approaches. Some approaches apply semantic parsing, where words and sentences are mapped to logical structure meaning (Kate and Mooney, 2007). Others provide automatic mappings of natural language instructions to executable actions, such as interpreting navigation directions (Chen and Mooney, 2011) or robot commands (Tellex et al., 2011; Matuszek et al., 2012). Some efforts have tackled tasks such as automatic image caption generation (Feng and Lapata, 2010a; Ordonez et al., 2011), text illustration (Joshi et al., 2006), or automatic location identification of Twitter users (Eisenstein et al., 2010; Wing and Baldridge, 2011; Roller et al., 2012).

Another line of research approaches grounded language knowledge by augmenting distributional approaches of word meaning with perceptual information (Andrews et al., 2009; Steyvers, 2010; Feng and Lapata, 2010b; Bruni et al., 2011; Silberer and Lapata, 2012; Johns and Jones, 2012; Bruni et al., 2012a; Bruni et al., 2012b; Silberer et al., 2013). Although these approaches have differed in model definition, the general goal in this line of research has been to enhance word meaning with perceptual information in order to address one of the most common criticisms of distributional semantics: that the "meaning of words is entirely given by other words" (Bruni et al., 2012b).

In this paper, we explore various ways to integrate new perceptual information through novel computational modeling of this grounded knowledge into a multimodal distributional model of word meaning. The model we rely on was originally developed by

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1146–1157,
Seattle, Washington, USA, 18-21 October 2013.

Andrews et al. (2009) and is based on a generalization of Latent Dirichlet Allocation. This model has previously been shown to provide excellent performance on multiple tasks, including prediction of association norms, word substitution errors, semantic inferences, and word similarity (Andrews et al., 2009; Silberer and Lapata, 2012). While prior work has used the model only with feature norms and visual attributes, we show that low-level image features are directly compatible with the model and provide improved representations of word meaning. We also show how simple, unsupervised clusters of images can act as a semantically useful and qualitatively interesting set of features. Finally, we describe two ways to extend the model by incorporating three or more modalities. We find that each modality provides useful but disjoint information for describing word meaning, and that a hybrid integration of multiple modalities provides significant improvements in the representations of word meaning. We release both our code and data to the community for future research.[1]

2 Related Work

The language grounding problem has received significant attention in recent years, owed in part to the wide availability of data sets (e.g. Flickr, Von Ahn (2006)), computing power, improved computer vision models (Oliva and Torralba, 2001; Lowe, 2004; Farhadi et al., 2009; Parikh and Grauman, 2011) and neurological evidence of ties between the language, perceptual and motor systems in the brain (Pulvermüller et al., 2005; Tettamanti et al., 2005; Aziz-Zadeh et al., 2006).

Many approaches to multimodal research have succeeded by abstracting away raw perceptual information and using high-level representations instead. Some works abstract perception via the usage of symbolic logic representations (Chen et al., 2010; Chen and Mooney, 2011; Matuszek et al., 2012; Artzi and Zettlemoyer, 2013), while others choose to employ concepts elicited from psycholinguistic and cognition studies. Within the latter category, the two most common representations have been association norms, where subjects are given a cue word and name the first (or several) associated words that come to mind (e.g., Nelson et al. (2004)), and feature norms, where subjects are given a cue word and asked to describe typical properties of the cue concept (e.g., McRae et al. (2005)).

Griffiths et al. (2007) helped pave the path for cognitive-linguistic multimodal research, showing that Latent Dirichlet Allocation outperformed Latent Semantic Analysis (Deerwester et al., 1990) in the prediction of association norms. Andrews et al. (2009) furthered this work by showing that a bimodal topic model, consisting of both text and feature norms, outperformed models using only one modality on the prediction of association norms, word substitution errors, and semantic interference tasks. In a similar vein, Steyvers (2010) showed that a different feature-topic model improved predictions on a fill-in-the-blank task. Johns and Jones (2012) take an entirely different approach by showing that one can successfully infer held out feature norms from weighted mixtures based on textual similarity. Silberer and Lapata (2012) introduce a new method of multimodal integration based on Canonical Correlation Analysis, and performs a systematic comparison between their CCA-based model and others on association norm prediction, held out feature prediction, and word similarity.

As computer vision techniques have improved over the past decade, other research has begun directly using visual information in place of feature norms. The first work to do this with topic models is Feng and Lapata (2010b). They use a Bag of Visual Words (BoVW) model (Lowe, 2004) to create a bimodal vocabulary describing documents. The topic model using the bimodal vocabulary outperforms a purely textual based model in word association and word similarity prediction. Bruni et al. (2012a) show how a BoVW model may be easily combined with a distributional vector space model of language using only vector concatenation. Bruni et al. (2012b) show that the *contextual* visual words (i.e. the visual features around an object, rather than of the object itself) are even more useful at times, suggesting the plausibility of a sort of distributional hypothesis for images. More recently, Silberer et al. (2013) show that visual attribute classifiers, which have been immensely successful in object recognition (Farhadi et al., 2009), act as excellent substitutes for feature

[1] http://stephenroller.com/research/emnlp13

norms. Other work on modeling the meanings of verbs using video recognition has also begun showing great promise (Mathe et al., 2008; Regneri et al., 2013).

The Computer Vision community has also benefited greatly from efforts to unify the two modalities. To name a few examples, Rohrbach et al. (2010) and Socher et al. (2013) show how semantic information from text can be used to improve zero-shot classification (i.e., classifying never-before-seen objects), and Motwani and Mooney (2012) show that verb clusters can be used to improve activity recognition in videos.

3 Data

Our experiments use several existing and new data sets for each of our modalities. We employ a large web corpus and a large set of association norms. We also introduce two new overlapping data sets: a collection of feature norms and a collection of images for a number of German nouns.

3.1 Textual Modality

For our **Text** modality, we use deWaC, a large German web corpus created by the WaCKy group (Baroni et al., 2009) containing approximately 1.7B word tokens. We filtered the corpus by: removing words with unprintable characters or encoding troubles; removing all stopwords; removing word types with a total frequency of less than 500; and removing documents with a length shorter than 100. The resulting corpus has 1,038,883 documents consisting of 75,678 word types and 466M word tokens.

3.2 Cognitive Modalities

Association Norms (AN) is a collection of association norms collected by Schulte im Walde et al. (2012). In association norm experiments, subjects are presented with a cue word and asked to list the first few words that come to mind. With enough subjects and responses, association norms can provide a common and detailed view of the meaning components of cue words. After removing responses given only once in the entire study, the data set contains a total of 95,214 cue-response pairs for 1,012 nouns and 5,716 response types.

Feature Norms (FN) is our new collection of feature norms for a group of 569 German nouns. We present subjects on Amazon Mechanical Turk with a cue noun and ask them to give between 4 and 8 typical descriptive features of the noun. Subjects are given ten example responses; one such example is a cue of *Tisch* 'table' and a response of *hat Beine* 'has legs'. After collection, subjects who are obvious spammers or did not follow instructions are manually filtered. Responses are manually corrected for spelling mistakes and semantically normalized.[2] Finally, responses which are only given once in the study are removed. The final data set contains 11,714 cue-response pairs for 569 nouns and 2,589 response types.

Note that the difference between association norms and feature norms is subtle, but important. In AN collection, subjects simply name related words as fast as possible, while in FN collection, subjects must carefully *describe* the cue.

3.3 Visual Modalities

BilderNetle ("little ImageNet" in Swabian German) is our new data set of German noun-to-ImageNet synset mappings. ImageNet is a large-scale and widely used image database, built on top of WordNet, which maps words into groups of images, called synsets (Deng et al., 2009). Multiple synsets exist for each meaning of a word. For example, ImageNet contains two different synsets for the word *mouse*: one contains images of the animal, while the other contains images of the computer peripheral. This BilderNetle data set provides mappings from German noun types to images of the nouns via ImageNet.

Starting with a set of noun compounds and their nominal constituents von der Heide and Borgwaldt (2009), five native German speakers and one native English speaker (including the authors of this paper) work together to map German nouns to ImageNet synsets. With the assistance of a German-English dictionary, the participants annotate each word with all its possible meanings. After discussing the annotations with the German speakers, the English speaker manually map the word meanings to synset senses in ImageNet. Finally, the German speakers review samples of the images for each word to en-

[2] For brevity, we include the full details of the spammer identification, cleansing process and normalization techniques in the Supplementary Materials.

sure the pictures accurately reflect the original noun in question. Not all words or meanings are mapped to ImageNet, as there are a number of words without entries in ImageNet, but the resulting data set contains a considerable amount of polysemy. The final data set contains 2022 word-synset mappings for just 309 words. All but three of these words overlap with our data set of feature norms. After extracting sections of images using bounding boxes when available by ImageNet (and using the entire image when bounding boxes are unavailable), the data set contains 1,305,602 images.

3.3.1 Image Processing

After the collection of all the images, we extracted simple, low-level computer vision features to use as modalities in our experiments.

First, we compute a simple Bag of Visual Words (BoVW) model for our images using SURF keypoints (Bay et al., 2008). SURF is a method for selecting points-of-interest within an image. It is faster and more forgiving than the commonly known SIFT algorithm. We compute SURF keypoints for every image in our data set using SimpleCV[3] and randomly sample 1% of the keypoints. The keypoints are clustered into 5,000 visual codewords (centroids) using k-means clustering (Sculley, 2010), and images are then quantized over the 5,000 codewords. All images for a given word are summed together to provide an average representation for the word. We refer to this representation as the **SURF** modality.

While this is a standard, basic BoVW model, each individual codeword on its own may not provide a large degree of semantic information; typically a BoVW representation acts predominantly as a feature space for a classifier, and objects can only be recognize using collections of codewords. To test that similar concepts should share similar visual codewords, we cluster the BoVW representations for all our images into 500 clusters with k-means clustering, and represent each word as membership over the image clusters, forming the **SURF Clusters** modality. The number of clusters is chosen arbitrarily. Ideally, each cluster should have a common object or clear visual attribute, and words are express in terms of these visual commonalities.

[3] `http://simplecv.org`

We also compute GIST vectors (Oliva and Torralba, 2001) for every image using LearGIST (Douze et al., 2009). Unlike SURF descriptors, GIST produces a single vector representation for an image. The vector does not find points of interest in the image, but rather attempts to provide a representation for the overall "gist" of the whole image. It is frequently used in tasks like scene identification, and Deselaers and Ferrari (2011) shows that distance in GIST space correlates well with semantic distance in WordNet. After computing the GIST vectors, each textual word is represented as the centroid GIST vector of all its images, forming the **GIST** modality.

Finally, as with the SURF features, we clustered the GIST representations for our images into 500 clusters, and represented words as membership in the clusters, forming the **GIST Clusters** modality.

4 Model Definition

Our experiments are based on the multimodal extension of Latent Dirichlet Allocation developed by Andrews et al. (2009). Previously LDA has been successfully used to infer unsupervised joint topic distributions over words and feature norms together (Andrews et al., 2009; Silberer and Lapata, 2012). It has also been shown to be useful in joint inference of text with visual attributes obtained using visual classifiers (Silberer et al., 2013). These multimodal LDA models (hereafter, mLDA) have been shown to be qualitatively sensible and highly predictive of several psycholinguistic tasks (Andrews et al., 2009). However, prior work using mLDA is limited to two modalities at a time. In this section, we describe bimodal mLDA and define two methods for extending it to three or more modalities.

4.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (Blei et al., 2003), or LDA, is an unsupervised Bayesian probabilistic model of text documents. It assumes that all documents are probabilistically generated from a shared set of K common topics, where each topic is a multinomial distribution over the vocabulary (notated as β), and documents are modeled as mixtures of these shared topics (notated as θ). LDA assumes every document in the corpus is generated using the fol-

lowing generative process:

1. A document-specific topic distribution, $\theta_d \sim Dir(\alpha)$ is drawn.
2. For the ith word in the document,
   (a) A topic assignment $z_i \sim \theta_d$ is drawn,
   (b) and a word $w_i \sim \beta_{z_i}$ is drawn and observed.

The task of Latent Dirichlet Allocation is then to automatically infer the latent document distribution θ_d for each document $d \in \mathcal{D}$, and the topic distribution β_k for each of the $k = \{1, \ldots, K\}$ topics, given the data. The probability that the ith word of document d is

$$p(w_i, \theta_d) = \sum_k p(w_i|\beta_k)p(z_i = k|\theta_d).$$

4.2 Multimodal LDA

Andrews et al. (2009) extend LDA to allow for the inference of document and topic distributions in a multimodal corpus. In their model, a document consists of a set of (word, feature) pairs,[4] rather than just words, and documents are still modeled as mixtures of shared topics. Topics consist of multinomial distributions over words, β_k, but are extended to also include multinomial distributions over features, ψ_k. The generative process is amended to include these feature distributions:

1. A document-specific topic distribution, $\theta_d \sim Dir(\alpha)$ is drawn.
2. For the ith (word, feature) pair in the document,
   (a) A topic assignment $z_i \sim \theta_d$ is drawn;
   (b) a word $w_i \sim \beta_{z_i}$ is drawn;
   (c) a feature $f_i \sim \psi_{z_i}$ is drawn;
   (d) the pair (w_i, f_i) is observed.

The conditional probability of the ith pair (w_i, f_i) is updated appropriately:

$$p(w_i, f_i, \theta_d) = \sum_k p(w_i|\beta_k)p(f_i|\psi_k)p(z_i = k|\theta_d).$$

The key aspect to notice is that the observed word w_i and feature f_i are conditionally independent given the topic selection, z_i. This powerful extension allows for joint inference over both words and features, and topics become the key link between the text and feature modalities.

[4]Here, and elsewhere, *feature* and f simply refer to a token from a nontextual modality and should not be confused with the machine learning sense of *feature*.

4.3 3D Multimodal LDA

We can easily extend the bimodal LDA model to incorporate three or more modalities by simply performing inference over n-tuples instead of pairs, and still mandating that each modality is conditionally independent given the topic. We consider the ith tuple $(w_i, f_i, f'_i, \ldots)$ in document d to have a conditional probability of:

$$p(w_i, f_i, f'_i, \ldots, \theta_d) = \sum_k p(w_i|\beta_i)p(f_i|\psi_k)p(f'_i|\psi'_i)\cdots p(z_i = k|\theta_d)$$

That is, we simply take the original mLDA model of Andrews et al. (2009) and generalize it in the same way they generalize LDA. At first glance, it seems that the inference task should become more difficult as the number of modalities increases and observed tuples become sparser, but the task remains roughly the same difficulty, as all of the observed elements of a tuple are conditionally independent given the topic assignment z_i.

4.4 Hybrid Multimodal LDA

3D Multimodal LDA assumes that all modalities share the same latent topic structure, θ_d. It is possible, however, that all modalities do *not* share some latent structure, but the modalities can still combine in order to enhance word meaning. The intuition here is that language usage is guided by all information gained in all modalities, but knowledge gained from one modality may not always relate to another modality. For example, the color red and the feature "is sweet" both enhance our understanding of strawberries. However, one cannot see that strawberries are sweet, so one should not correlate the color red with the feature "is sweet."

To this end, we define Hybrid Multimodal LDA. In this setting, we perform separate, bimodal mLDA inference according to Section 4.2 for each of the different modalities, and then concatenate the topic distributions for the words. In this way, Hybrid mLDA assumes that every modality shares some latent structure with the text in the corpus, but the latent structures are not shared between non-textual modalities.

For example, to generate a hybrid model for text, feature norms and SURF, we separately perform bimodal mLDA for the text/feature norms modalities and the text/SURF modalities. This provides us with two topic-word distributions: $\beta^{FN}_{k,w}$ and $\beta^{S}_{k',w}$, and the hybrid model is simply the concatenation of the two distributions,

$$\beta^{FN\&S}_{j,w} = \begin{cases} \beta^{FN}_{j,w} & 1 \leq j \leq K^{FN}, \\ \beta^{S}_{j-K^{FN},w} & K^{FN} < j \leq K^{FN} + K^{S}, \end{cases}$$

where K^{FN} indicates the number of topics for the Feature Norm modality, and likewise for K^S.

4.5 Inference

Analytical inference of the posterior distribution of mLDA is intractable, and must be approximated. Prior work using mLDA has used Gibbs Sampling to approximate the posterior, but we found this method did not scale with larger values of K, especially when applied to the relatively large deWaC corpus.

To solve these scaling issues, we implement Online Variational Bayesian Inference (Hoffman et al., 2010; Hoffman et al., 2012) for our models. In Variational Bayesian Inference (VBI), one approximates the true posterior using simpler distributions with free variables. The free variables are then optimized in an EM-like algorithm to minimize difference between the true and approximate posteriors. Online VBI differs from normal VBI by using randomly sampled minibatches in each EM step rather than the entire data set. Online VBI easily scales and quickly converges in all of our experiments. A listing of the inference algorithm may be found in the Supplementary Materials and the source code is available as open source.

5 Experimental Setup

5.1 Generating Multimodal Corpora

In order to evaluate our algorithms, we first need to generate multimodal corpora for each of our non-textual modalities. We use the same method as Andrews et al. (2009) for generating our multimodal corpora: for each word token in the text corpus, a feature is selected stochastically from the word's feature distribution, creating a word-feature pair. Words without grounded features are all given the same *placeholder* feature, also resulting in a word-feature pair.[5] That is, for the feature norm modality, we generate (word, feature norm) pairs; for the SURF modality, we generate (word, codeword) pairs, etc. The resulting stochastically generated corpus is used in its corresponding experiments.

The 3D text-feature-association norm corpus is generated slightly differently: for each word in the original text corpus, we check the existence of multimodal features in either modality. If a word had no features, it is represented as a triple (word, placeholder$_{FN}$, placeholder$_{AN}$). If the word had only feature norms, but no associations, it is generated as (word, feature, placeholder$_{AN}$), and similarly for association norms without feature norms. In the case of words with presence in both modalities, we generate *two* triples: (word, feature, placeholder$_{AN}$) and (word, placeholder$_{FN}$, association). This allows association norms and feature norms to influence each other via the document mixtures θ, but avoids falsely labeling explicit relationships between randomly selected feature norms and associations.[6] Other 3D corpora are generated using the same general procedure.

5.2 Evaluation

We evaluate each of our models with two data sets: a set of compositionality ratings for a number of German noun-noun compounds, and the same association norm data set used as one of our training modalities in some settings.

Compositionality Ratings is a data set of compositionality ratings originally collected by von der Heide and Borgwaldt (2009). The data set consists of 450 concrete, depictable German noun compounds along with compositionality ratings with regard to their constituents. For each compound, 30 native German speakers are asked to rate how related the meaning of the compound is to each of its constituents on a scale from 1 (highly opaque; entirely noncompositional) to 7 (highly transparent; very compositional). The mean of the 30 judgments

[5] Placeholder features must be hardcoded to have equal probability over all topics to prevent all placeholder pairs from aggregating into a single topic.

[6] We did try generating the random triples without placeholders, but the generated explicit relationships are overwhelmingly detrimental in the settings we attempted.

is taken as the gold compositionality rating for each of the compound-constituent pairs. For example, *Ahornblatt* 'maple leaf' is rated highly transparent with respect to its constituents, *Ahorn* 'maple' and *Blatt* 'leaf', but *Löwenzahn* 'dandelion' is rated non-compositional with respect to its constituents, *Löwe* 'lion' and *Zahn* 'tooth'.

We use a subset of the original data, comprising of all two-part noun-noun compounds and their constituents. This data set consists of 488 compositionality ratings (244 compound-head and 244 compound-modifier ratings) for 571 words. 309 of the targets have images (the entire image data set); 563 have feature norms; and all 571 of have association norms.

In order to predict compositionality, for each compound-constituent pair $(w_{compound}, w_{constituent})$, we compute negative symmetric KL divergence between the two words' topic distributions, where symmetric KL divergence is defined as

$$sKL(w_1||w_2) = KL(w_1||w_2) + KL(w_2||w_1),$$

and KL divergence is defined as

$$KL(w_1||w_2) = \sum_k \ln\left(\frac{p(t=k|w_1)}{p(t=k|w_2)}\right) p(t=k|w_1).$$

The values of $-sKL$ for all compound-constituent word pairs are correlated with the human judgments of compositionality using Spearman's ρ, a rank-order correlation coefficient. Note that, since KL divergence is a measure of *dissimilarity*, we use *negative* symmetric KL divergence so that our ρ correlation coefficient is positive. For example, we compute both $-sKL(Ahornblatt, Ahorn)$ and $-sKL(Ahornblatt, Blatt)$, and so on for all 488 compound-constituent pairs, and then correlate these values with the human judgments.

Additionally, we also evaluate using the **Association Norms** data set described in Section 3. Since it is not sensible to evaluate association norm prediction when they are also used as training data, we omit this evaluation for this modality. Following Andrews et al. (2009), we measure association norm prediction as an average of percentile ranks. For all possible pairs of words in our vocabulary, we compute the negative symmetric KL divergence between the two words. We then compute the percentile ranks of similarity for each word pair, e.g., "cat" is more similar to "dog" than 97.3% of the rest of the vocabulary. We report the weighted mean percentile ranks for all cue-association pairs, i.e., if a cue-association is given more than once, it is counted more than once.

5.3 Model Selection and Hyperparameter Optimization

In all settings, we fix all Dirichlet priors at 0.1, use a learning rate 0.7, and use minibatch sizes of 1024 documents. We do not optimize these hyperparameters or vary them over time. The high Dirichlet priors are chosen to prevent sparsity in topic distributions, while the other parameters are selected as the best from Hoffman et al. (2010).

In order to optimize the number of topics K, we run five trials of each modality for 2000 iterations for $K = \{50, 100, 150, 200, 250\}$ (a total of 25 runs per setup). We select the value or K for each model which minimizes the average perplexity estimate over the five trials.

6 Results

6.1 Predicting Compositionality Ratings

Table 1 shows our results for each of our selected models with our compositionality evaluation. The 2D models employing feature norms and association norms do significantly better than the text-only model (two-tailed t-test). This result is consistent with other works using this model with these features (Andrews et al., 2009; Silberer and Lapata, 2012).

We also see that the SURF visual words are able to provide notable, albeit not significant, improvements over the text-only modality. This confirms that the low-level BoVW features do carry semantic information, and are useful to consider individually. The GIST vectors, on the other hand, perform almost exactly the same as the text-only model. These features, which are usually more useful for comparing overall image likeness than object likeness, do not *individually* contain semantic information useful for compositionality prediction.

The performance of the visual modalities reverses when we look at our cluster-based models. Text

Modality	K	ρ
Text Only		
Text Only (LDA)	200	.204
Bimodal mLDA		
Text + Feature Norms	150	.310 ***
Text + Assoc. Norms	200	.328 **
Text + SURF	50	.251
Text + GIST	100	.204
Text + SURF Clusters	200	.159
Text + GIST Clusters	150	.233
3D mLDA		
Text + FN + AN	250	.259
Text + FN + SURF	100	.286 *
Text + FN + GC	200	.261 *
Hybrid mLDA		
FN, AN	150+200	.390 ***
FN, SURF	150+50	.350 ***
FN, GC	150+150	.340 ***
FN, AN, GC	150+200+150	.395 ***
FN, AN, SURF	150+200+50	.404 ***
FN, AN, SURF, GC	150+200+50+150	**.406** ***

Table 1: Average rank correlations between $-sKL(w_{compound}, w_{constituent})$ and our Compositionality gold standard. The Hybrid models are the concatenation of the corresponding Bimodal mLDA models. Stars indicate statistical significance compared to the text-only setting at the .05, .01 and .001 levels using a two-tailed t-test.

combined with SURF clusters is our worst performing system, indicating our clusters of images with common visual words are actively working against us. The clusters based on GIST, on the other hand, provide a minor improvement in compositionality prediction.

All of our 3D models are better than the text-only model, but they show a performance drop relative to one or both of their comparable bimodal models. The model combining text, feature norms, and association norms is especially surprising: despite the excellent performance of each of the bimodal models, the 3D model performs significantly worse than either of its components ($p < .05$). This indicates that these modalities provide new insight into word meaning, but cannot be forced into the same latent structure.

The hybrid models show massive performance increases across the board. Indeed, our 5 modality hybrid model obtains a performance nearly twice that of the text-only model. Not only do all 6 hybrid models do significantly better than the text-only models, they show a highly significant improvement over their individual components ($p < .001$ for all 16 comparisons). Furthermore, improvements generally continue to grow significantly with each additional modality we incorporate into the hybrid model ($p < .001$ for all but the .404 to .406 comparison, which is not significant). Clearly, there is a great deal to learn from combining three, four and even five modalities, but the modalities are learning *disjoint* knowledge which cannot be forced into a shared, latent structure.

Modality	K	Assoc.
Text Only		
Text Only (LDA)	200	.679
Bimodal mLDA		
Text + Feature Norms	150	.676
Text + SURF	50	.789 ***
Text + GIST	100	.739 ***
Text + SURF Clusters	200	.618 ***
Text + GIST Clusters	150	.690
3D mLDA		
Text + FN + SURF	100	.722 ***
Text + FN + GC	200	.601 ***
Hybrid mLDA		
FN, SURF	150+50	.800 ***
FN, GC	150+150	.742 ***
FN, GC, SURF	150+150+50	**.804** ***

Table 2: Average predicted rank similarity between cue words and their associates. Stars indicate statistical significance compared to the text-only modality, with gray stars indicating the model is statistically worse than the text model. The Hybrid models are the concatenation of the corresponding Bimodal mLDA models.

6.2 Predicting Association Norms

Table 2 shows the average weighted predicted rank similarity between all cue words and associates and trials. Here we see that feature norms do not seem to be improving performance on the association norms. This is slightly unexpected, but consistent with the result that feature norms seem to provide helpful, but disjoint semantic information as association norms.

We see that the image modalities are much more useful than they are in compositionality prediction. The SURF modality does extremely well in particular, but the GIST features also provide statistically significant improvements over the text-only model. Since the SURF and GIST image features tend to capture object-likeness and scene-likeness respectively, it is possible that words which share associates are likely related through common settings and objects that appear with them. This seems to provide additional evidence of Bruni et al. (2012b)'s suggestion that something like a distributional hypothesis of images is plausible.

Once again, the clusters of images using SURF causes a dramatic drop in performance. Combined with the evidence from the compositionality assessment, this shows that the SURF clusters are actively confusing the models and not providing semantic information. GIST clusters, on the other hand, are providing a marginal improvement over the text-only model, but the result is not significant. We take a qualitative look into the GIST clusters in the next section.

Once again, we see that the 3D models are ineffective compared to their bimodal components, but the hybrid models provide at least as much information as their components. The Feature Norms and GIST Clusters hybrid model significantly improves over both components.[7] The final four-modality hybrid significantly outperforms all comparable models. As with the compositionality evaluation, we conclude that the image and and feature norm models are providing disjoint semantic information that cannot be forced into a shared latent structure, but still augment each other when combined.

7 Qualitative Analysis of Image Clusters

In all research connecting word meaning with perceptual information, it is desirable that the inferred representations be directly interpretable. One nice property of the cluster-based modalities is that we may represent each cluster as its prototypical images, and examine whether the prototypes are related to the topics.

We chose to limit our analysis to the GIST clusters for two primary reasons: first, the SURF clusters did not perform well in our evaluations, and second, preliminary investigation into the SURF clusters show that the majority of SURF clusters are nearly identical. This indicates our SURF clusters are likely hindered by poor initialization or parameter selection, and may partially explain their poor performance in evaluations.

We select our single best Text + GIST Clusters trial from the Compositionality evaluation and look at the topic distributions for words and image clusters. For each topic, we select the three clusters with the highest weight for the topic, $p(c|\psi_k)$. We extract the five images closest to the cluster centroids, and select two topics whose prototypical images are the most interesting and informative. Figure 1 shows these selected topics.

The first example topic contains almost exclusively water-related terms. The first image, extracted from the most probable cluster, does not at first seem related to water. Upon further inspection, we find that many of the water-related pictures are scenic views of lakes and mountains, often containing a cloudy sky. It seems that the GIST cluster does not tend to group images of water, but rather nature scenes that may contain water. This relationship is more obvious in the second picture, especially when one considers the water itself contains reflections of the trees and mountain.

The second topic contains time-related terms. The "@card@" term is a special token for all non-zero and non-one numbers. The second word, "Uhr", is polysemous: it can mean *clock*, an object which tells the time, or *o'clock*, as in *We meet at 2 o'clock* ("Wir treffen uns um 2 Uhr.") The three prototypical pictures are not pictures of clocks, but round, detailed objects similar to clocks. We see GIST has a preference toward clustering images based on the predominant shape of the image. Here we see the clusters of GIST images are not providing a definite semantic relationship, but an overwhelming visual one.

8 Conclusions

In this paper, we evaluated the role of low-level image features, SURF and GIST, for their compatibility with the multimodal Latent Dirichlet Allocation model of Andrews et al. (2009). We found both fea-

[7] The gain is smaller than compared to SURF Hybrid, but there is much less variance in the trials.

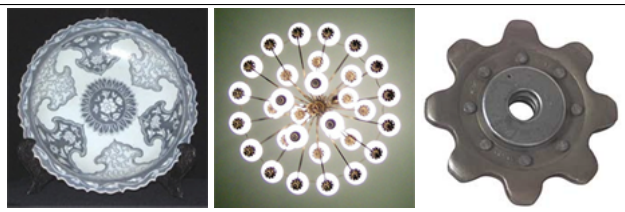
Most Probable Words	Translations	Prototypical Images
Wasser Schiff See Meer Meter Fluß	water ship lake sea meter river	
@card@ Uhr Freitag Sonntag Samstag Montag	(number) clock Friday Sunday Saturday Monday	

Figure 1: Example topics with prototypical images for the Text + GIST Cluster modality. The first topic shows water-related words, as well scenes which often appear with water. The second shows clock-like objects, but not clocks.

ture sets were directly compatible with multimodal LDA and provided significant gains in their ability to predict association norms over traditional text-only LDA. SURF features also provided significant gains over text-only LDA in predicting the compositionality of noun compounds.

We also showed that words may be represented in terms of membership of image clusters based on the low-level image features. Image clusters based on GIST features were qualitatively interesting, and were able to give improvements over the text-only model.

Finally, we showed two methods for extending multimodal LDA to three or more modalities: the first as a 3D model with a shared latent structure between all modalities, and the second where latent structures were inferred separately for each modality and joined together into a hybrid model. Although the 3D model was unable to compete with its bimodal components, we found the hybrid model consistently improved performance over its component modalities. We conclude that the combination of many modalities provides the best representation of word meaning, and that each nontextual modality is discovering disjoint information about word meaning that cannot be forced into a global latent structure.

Acknowledgments

We would like to thank the UT Natural Language Learning Reading Group and the anonymous EMNLP reviewers for their most helpful comments and suggestions. We would also like to thank the members of the SemRel group at IMS for their considerable help in the construction of BilderNetle.

The authors acknowledge the Texas Advanced Computing Center (TACC) for providing the grid computing resources necessary for these results. The research presented in this paper was funded by the DFG Collaborative Research Centre SFB 732 (Stephen Roller) and the DFG Heisenberg Fellowship SCHU-2580/1-1 (Sabine Schulte im Walde).

References

Mark Andrews, Gabriella Vigliocco, and David Vinson. 2009. Integrating experiential and distributional data to learn semantic representations. *Psychological Review*, 116(3):463.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *Transactions of the Association for Computational Linguistics*, volume 1, pages 49–62.

Lisa Aziz-Zadeh, Stephen M. Wilson, Giacomo Rizzolatti, and Marco Iacoboni. 2006. Congruent embodied representations for visually presented actions and linguistic phrases describing actions. *Current Biology*, 16(18):1818–1823.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a

collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, June.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Elia Bruni, Giang Binh Tran, and Marco Baroni. 2011. Distributional semantics from text and images. *Proceedings of the EMNLP 2011 Geometrical Models for Natural Language Semantics*, pages 22–32.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012a. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 136–145.

Elia Bruni, Jasper Uijlings, Marco Baroni, and Nicu Sebe. 2012b. Distributional semantics with eyes: Using image analysis to improve computational representations of word meaning. In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 1219–1228.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 859–865, August.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37(1):397–436.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Thomas Deselaers and Vittorio Ferrari. 2011. Visual and semantic similarity in imagenet. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1777–1784.

Matthijs Douze, Hervé Jégou, Harsimrat Sandhawalia, Laurent Amsaleg, and Cordelia Schmid. 2009. Evaluation of gist descriptors for web-scale image search. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 19:1–19:8.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. 2009. Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785.

Yansong Feng and Mirella Lapata. 2010a. How many words is a picture worth? Automatic caption generation for news images. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1239–1249.

Yansong Feng and Mirella Lapata. 2010b. Visual information in semantic representation. In *Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211.

Matthew Hoffman, David M. Blei, and Francis Bach. 2010. Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864.

Matthew Hoffman, David M. Blei, Chong Wang, and John Paisley. 2012. Stochastic variational inference. *ArXiv e-prints*, June.

Brendan T. Johns and Michael N. Jones. 2012. Perceptual inference through global lexical similarity. *Topics in Cognitive Science*, 4(1):103–120.

Dhiraj Joshi, James Z. Wang, and Jia Li. 2006. The story picturing engine—a system for automatic text illustration. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):68–89.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Proceedings of the 22nd Conference on Artificial Intelligence*, volume 7, pages 895–900.

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

Stefan Mathe, Afsaneh Fazly, Sven Dickinson, and Suzanne Stevenson. 2008. Learning the abstract motion semantics of verbs from captioned videos. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012. Learning to parse natural language commands to a robot control system. In *Proceedings of the 13th International Symposium on Experimental Robotics*.

Ken McRae, George S. Cree, Mark S. Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.

Tanvi S. Motwani and Raymond J. Mooney. 2012. Improving video activity recognition using object recognition and text mining. In *ECAI*, pages 600–605.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Aude Oliva and Antonio Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems*, pages 1143–1151.

Devi Parikh and Kristen Grauman. 2011. Relative attributes. In *International Conference on Computer Vision*, pages 503–510. IEEE.

Friedemann Pulvermüller, Olaf Hauk, Vadim V. Nikulin, and Risto J Ilmoniemi. 2005. Functional links between motor and language systems. *European Journal of Neuroscience*, 21(3):793–797.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. In *Transactions of the Association for Computational Linguistics*, volume 1, pages 25–36.

Marcus Rohrbach, Michael Stark, György Szarvas, Iryna Gurevych, and Bernt Schiele. 2010. What helps where–and why? Semantic relatedness for knowledge transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 910–917.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1500–1510.

Sabine Schulte im Walde, Susanne Borgwaldt, and Ronny Jauch. 2012. Association norms of german noun compounds. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 632–639, Istanbul, Turkey.

D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1177–1178.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1423–1433, Jeju Island, Korea, July.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August.

Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero-shot learning through cross-modal transfer. *International Conference on Learning Representations*.

Mark Steyvers. 2010. Combining feature norms and text data with topic models. *Acta Psychologica*, 133(3):234–243.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth J. Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. *Proceedings of the 25th AAAI Conference on Artificial Intelligence*.

Marco Tettamanti, Giovanni Buccino, Maria Cristina Saccuman, Vittorio Gallese, Massimo Danna, Paola Scifo, Ferruccio Fazio, Giacomo Rizzolatti, Stefano F. Cappa, and Daniela Perani. 2005. Listening to action-related sentences activates fronto-parietal motor circuits. *Journal of Cognitive Neuroscience*, 17(2):273–281.

Luis Von Ahn. 2006. Games with a purpose. *Computer*, 39(6):92–94.

Claudia von der Heide and Susanne Borgwaldt. 2009. Assoziationen zu Unter-, Basis- und Oberbegriffen. Eine explorative Studie. In *Proceedings of the 9th Norddeutsches Linguistisches Kolloquium*, pages 51–74.

Benjamin Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 11, pages 955–964.

Combining PCFG-LA Models with Dual Decomposition: A Case Study with Function Labels and Binarization

Joseph Le Roux[†], **Antoine Rozenknop**[†], **Jennifer Foster**[*]
[†] Université Paris 13, Sorbonne Paris Cité, LIPN, F-93430, Villetaneuse, France
[*] NCLT/CNGL, School of Computing, Dublin City University, Dublin 9, Ireland
joseph.leroux@lipn.fr antoine.rozenknop@lipn.fr jfoster@computing.dcu.ie

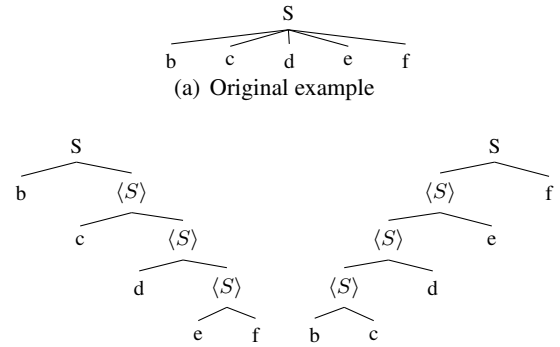
Abstract

It has recently been shown that different NLP models can be effectively combined using dual decomposition. In this paper we demonstrate that PCFG-LA parsing models are suitable for combination in this way. We experiment with the different models which result from alternative methods of extracting a grammar from a treebank (retaining or discarding function labels, left binarization versus right binarization) and achieve a labeled Parseval F-score of 92.4 on Wall Street Journal Section 23 – this represents an absolute improvement of 0.7 and an error reduction rate of 7% over a strong PCFG-LA product-model baseline. Although we experiment only with binarization and function labels in this study, there is much scope for applying this approach to other grammar extraction strategies.

1 Introduction

Because of the large amount of possibly contradictory information contained in a treebank, learning a phrase-structure-based parser implies making several choices regarding the prevalent annotations which have to be kept – or discarded – in order to guide the learning algorithm. These choices, which include whether to keep function labels and empty nodes, how to binarize the trees and whether to alter the granularity of the tagset, are often motivated empirically by parsing performance rather than by the different aspects of the language they may be able to capture.

Recently Rush et al. (2010), Martins et al. (2011) and Koo et al. (2010) have shown that Dual Decomposition or Lagrangian Relaxation is an elegant framework for combining different types of NLP tasks or for building parsers from simple *slave* processes that only check partial well-formedness. Here we propose to follow this idea, but with a different objective. We want to mix different parsers trained on different versions of a treebank each of which makes some annotation choices in order to learn more specific or richer information. We will use state-of-the-art unlexicalized probabilistic context-free grammars with latent annotations (PCFG-LA) in order to compare our approach with a strong baseline of high-quality parses. Dual Decomposition is used to mix several systems (between two and four) that may in turn be combinations of grammars, here products of PCFG-LAs (Petrov, 2010). The systems being combined make different choices with regard to i) function labels and ii) grammar binarization.



(a) Original example
(b) Left Binarized example (c) Right Binarized example

Figure 1: Binarization with markovization

Common sense would suggest that information in the form of function labels – syntactic labels such as SBJ and PRD and semantic labels such as TMP and LOC – might help in obtaining a fine-grained analysis. On the other hand, the independence hypothe-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1158–1169,
Seattle, Washington, USA, 18-21 October 2013.

sis on which CFGs rely and on which most popular parsers are based may be too strong to learn the dependencies between functions across the parse trees. Also, the number of parameters increases with the use of function labels and this can affect the learning process.

At first glance, binarization need not be an issue, as CFGs admit a binarized form recognizing exactly the same language. But binarization can be associated with horizontal markovization and in this case the recognized language will differ. Furthermore this can impose an unwanted emphasis on what frontier information is more relevant to learning (beginning or end of constituents). In the toy example of Figure 1, the original grammar consisting of a unique rule extracted from one tree only recognizes the string `bcdef`, while the grammar learned from the left binarized and markovized tree recognizes (among others) `bcdef` and `bdcef` and the grammar learned from the right binarized and markovized tree recognizes (among others) `bcdef` and `bcedf`.

We find that i) retaining the function labels in non-terminal categories loses its negative impact on parsing as the number of grammars increases in PCFG-LA product models, ii) the function labels themselves can be recovered with near state-of-the-art-accuracy, iii) combining grammars with and without function labels using dual decomposition is beneficial, iv) combining left and right-binarized grammars using dual decomposition also leads to better trees and, v) our best results (a Parseval labeled F-score of 92.4, a Stanford labeled attachment score (LAS) of 93.0 and a penn2malt unlabeled attachment score (UAS) of 94.3 on Section 23 of the Wall Street Journal) are obtained by combining three grammars which encode different function label/binarization decisions.

The paper is organized as follows. § 2 reviews related work. § 3 presents approximate PCFG-LA parsers as linear models, while § 4 shows how we can use dual decomposition to derive an algorithm for combining these models. Experimental results are presented and discussed in § 5.

2 Related Work

Parser Model Combination It is well known that improved parsing performance can be achieved by leveraging the alternative perspectives provided by several parsing models rather than relying on just one. Examples are parser co-training (Steedman et al., 2003; Sagae and Tsujii, 2007), voting over phrase structure constituents or dependency arcs (Henderson and Brill, 1999; Sagae and Tsujii, 2007; Surdeanu and Manning, 2010), dependency parsing stacking (Nivre and McDonald, 2008), product model PCFG-LA parsing (Petrov, 2010), using dual decomposition to combine dependency and phrase structure models (Rush et al., 2010) or several non-projective dependency parsing models (Koo et al., 2010; Martins et al., 2011), and using expectation propagation, a related approach to dual decomposition, to combine lexicalized, unlexicalized and PCFG-LA models (Hall and Klein, 2012). In this last example, the models must factor in the same way: in other words, the grammars must use the same binarization scheme. In our study, we employ PCFG-LA product models with dual decomposition, and we relax the constraints on factorization, as we require only a loose coupling of the models.

Function Label Parsing Although function labels have been available in the Penn Treebank (PTB) for almost twenty years (Marcus et al., 1994), they have been to a large extent overlooked in English parsing research — most studies that report parsing results on Section 23 of the Wall Street Journal (WSJ) use parsing models that are trained on a version of the WSJ trees where the function labels have been removed. Notable exceptions are Merlo and Musillo (2005) and Gabbard et al. (2006) who each trained a parsing model on a version of the PTB with function labels intact. Gabbard et al. (2006) found that parsing accuracy was not affected by keeping the function labels. There have also been attempts to use machine learning to recover the function labels post-parsing (Blaheta and Charniak, 2000; Chrupala et al., 2007). We recover function labels as part of the parsing process, and use dual decomposition to combine parsing models with and without function labels. We are not aware of any other work that leverages the benefits of both types of models.

Grammar Binarization Matsuzaki et al. (2005) compare binarization strategies for PCFG-LA parsing, and conclude that the differences between them have a minor effect on parsing accuracy as the num-

ber of latent annotations increases beyond two. Hall and Klein (2012) are forced to use head binarization when combining their lexicalized and unlexicalized parsers. Dual decomposition allows us to combine models with different binarization schemes.

3 Approximation of PCFG-LAs as Linear Models

In this section, we explain how we can use PCFG-LAs to devise linear models suitable for the dual decomposition framework.

3.1 PCFG-LA

Let us recall that PCFG-LAs are defined as tuples $G = (\mathcal{N}, \mathcal{T}, \mathcal{H}, \mathcal{R}_\mathcal{H}, S, p)$ where:

- $\mathcal{N}$ is a set of observed non-terminals, among which S is the distinguished initial symbol,
- $\mathcal{T}$ is a set of terminals (words),
- $\mathcal{H}$ is a set of *latent annotations* or hidden states,
- $\mathcal{R}_\mathcal{H}$ is a set of annotated rules, of the form $a[h_1] \rightarrow b[h_2]\ c[h_3]$ for internal rules[1] and $a[h_1] \rightarrow w$ for lexical rules. Here $a, b, c \in \mathcal{N}$ are non-terminals, $w \in \mathcal{T}$ is a terminal and $h_1, h_2, h_3 \in \mathcal{H}$ are latent annotations. Following Cohen et al. (2012) we also define the set of skeletal rules $\mathcal{R}$, in other words, rules without hidden states, of the form $a \rightarrow b\ c$ or $a \rightarrow w$.
- $p : \mathcal{R}_\mathcal{H} \rightarrow \mathbb{R}_{\geq 0}$ defines the probabilities associated with rules conditioned on their left-hand side. Like Petrov and Klein (2007), we impose that the initial symbol S has only one latent annotation. In other words, among rules with S on the left-hand side, only those of the form $S[0] \rightarrow \gamma$ are in $\mathcal{R}_\mathcal{H}$.

With such a grammar G we can define probabilities over trees in the following way. We will consider two types of trees, annotated trees and skeletal trees. An annotated tree is a sequence of rules from $\mathcal{R}_\mathcal{H}$, while a skeletal tree is a sequence of skeletal rules from R. An annotated tree $T_\mathcal{H}$ is obtained by left-most derivation from $S[0]$. Its probability is:

$$p(T_\mathcal{H}) = \prod_{r \in T_\mathcal{H}} p(r) \tag{1}$$

We define a projection ρ from annotated trees to skeletal trees. $\rho(T_\mathcal{H})$ is a tree T isomorphic to $T_\mathcal{H}$ with the same terminal and non-terminal symbols labeling nodes, without hidden states. The probability of a skeletal tree T is a sum of the probabilities of all annotated trees that admit T as their projection.

$$p(T) = \sum_{T_\mathcal{H} \in \rho^{-1}(T)} \prod_{r \in T_\mathcal{H}} p(r) \tag{2}$$

PCFG-LA parsing amounts to, given a sequence of words, finding the most probable skeletal tree with this sequence as its yield according to a grammar G:

$$T^* = \arg\max_T \sum_{T_\mathcal{H} \in \rho^{-1}(T)} \prod_{r \in T_\mathcal{H}} p(r) \tag{3}$$

Because of this alternation of sum and products, the parsing problem is intractable. Moreover, the PCFG-LAs do not belong to the family of linear models that are assumed in the Lagrangian framework of (Rush and Collins, 2012). We now turn to approximations for the parsing problem in order to address both issues.

3.2 Variational Inference and MaxRule

Variational inference is a common technique to approximate a probability distribution p with a cruder one q, as close as possible to the original one, by minimizing the Kullback-Liebler divergence between the two – see for instance (Smith, 2011), chapter 5 for an introduction. Matsuzaki et al. (2005) showed that one can easily find such a cruder distribution for PCFG-LAs and demonstrated experimentally that this approximation gives good results. More precisely, they find a PCFG that only recognizes the input sentence where the probabilities $q(r_s)$ of the rules are set according to their marginal probabilities in the original PCFG-LA parse forest. The parameters r_s are skeletal rules with *span information*. Distribution q is defined in Figure 2.

Other approximations are possible. In particular, Petrov and Klein (2007) found that normalizing by the forest probability (in other words the inside probability of the root node) give better exper-

[1] For brevity and without loss of generality, we omit unary and n-ary rules, as PCFG-LA admit a Chomsky normal form.

$$
\begin{aligned}
score(a \to b\,c, i, j, k) &= \sum_{x,y,z \in \mathcal{H}} P_{\text{out}}^{i,k}\big(a[x]\big) \cdot p\big(\, a[x] \to b[y]\; c[z]\,\big) \cdot P_{\text{in}}^{i,j}\big(b[y]\big) \cdot P_{\text{in}}^{j,k}\big(c[z]\big) \\
norm(a \to b\,c, i, j, k) &= \sum_{x \in \mathcal{H}} P_{\text{in}}^{i,k}\big(a[x]\big) \cdot P_{\text{out}}^{i,k}\big(a[x]\big) \\
score(a \to w, i) &= \sum_{x \in \mathcal{H}} P_{\text{out}}^{i,i}\big(a[x]\big) \cdot p\big(a[x] \to w\big) \\
norm(a \to w, i) &= \sum_{x \in \mathcal{H}} P_{\text{in}}^{i,i}\big(a[x]\big) \cdot P_{\text{out}}^{i,i}\big(a[x]\big) \\
q(r_s) &= \left[\frac{score(r_s)}{norm(r_s)} \text{ (Variational Inference)}\right] \text{ or } \left[\frac{score(r_s)}{P_{\text{in}}^{0,n}\,(S[0])} \text{ (MaxRule-Product)}\right]
\end{aligned}
$$

Figure 2: Variational Inference for PCFG-LA. P_{in} and P_{out} denote inside and outside probabilities.

imental results although its interpretation as variational inference is still unclear. This approximation is called MaxRule-Product and amounts to replacing the $norm$ function (see Figure 2).

In both cases, the probability of a skeletal tree now becomes a simple product of parameters associated with anchored skeletal rules. For our purpose, the consequence is twofold:

1. The parsing problem becomes tractable by applying standard PCFG algorithms relying on dynamic programming (CKY for example).

2. Equivalent to probability, a score σ can be defined as the logarithm of the probability. The parsing problem becomes[2]:

$$
\begin{aligned}
T^* &= \arg\max_T \prod_{r_s \in T} q(r_s) \\
&= \arg\max_T \sum_{r_s \in T} \log q(r_s) \\
&= \arg\max_T \sum_{r_s \in \mathcal{F}} w_{r_s} \cdot \mathbf{1}\{r_s \in T\} \\
&= \arg\max_T \sigma(T)
\end{aligned}
$$

Thus, from a PCFG-LA we are able to define a linear model whose parameters are the log-probabilities of the rules in distribution q.

[2]We denote the parse forest of a sentence by $\mathcal{F}$ and the characteristic function of a set by $\mathbf{1}$.

3.3 Products of PCFG-LAs

Although PCFG-LA training is beyond the scope of this paper, it is worthwhile mentioning that the most common way to learn their parameters relies on Expectation-Maximization which is not guaranteed to find the optimal estimation. Fortunately, this can be partly overcome by combining grammars that only differ on the initial parameterization of the EM algorithm. The probability of a skeletal tree is the product of the probabilities assigned by each single grammar G_i.

$$
T^* = \arg\max_T \prod_{i=1}^{n} q_{G_i}(T) \tag{4}
$$

Since grammars only differ by their numerical parameters (i.e. skeletal rules are the same), inference can be efficiently implemented using dynamic programming (Petrov, 2010).

Scoring with n such grammars now becomes:

$$
\begin{aligned}
T^* &= \arg\max_T \sum_{i=1}^{n} \sum_{r \in T} \log q_{G_i}(r) &(5)\\
&= \arg\max_T \sum_{r \in T} \sum_{i=1}^{n} \log q_{G_i}(r) &(6)
\end{aligned}
$$

The distributions q_{G_i} still have to be computed independently – and possibly in parallel – but the final decoding can be performed jointly. This is still a linear model for PCFG-LA parsing, but restricted to grammars that share the same skeletal rules.

4 Dual Decomposition

In this section, we show how we derive an algorithm to work out the best parse according to a set of n grammars that do not share the exact same skeletal rules. As such, the grammars' product cannot be easily conducted inside the parser to produce and score a same and unique best tree, and we now consider a *c(ompound)-parse* as a tuple $(T_1 \dots T_n)$ of n *compatible* trees. Each grammar G_i is responsible for scoring tree T_i, and we seek to obtain the *c-parse* that maximizes the sum of the scores of its different trees. For a *c-parse* to be consistent, we have to precisely define the parts on which the trees must agree to be compatible with each other, so that we can model these as agreement constraints.

4.1 Compound Parse Consistency

Let us suppose we have a set of phrase-structure parsers trained on different versions of the same treebank. Hence, some elements in the charts will either be the same or can be mapped to each other provided an equivalence relation and we define consensus between parsers on these elements.

When the grammar is not functionally annotated, phrase-structure trees can be decomposed into a set of anchored (syntactical) categories X_s, asserting that a category X is in the tree at position[3] s. Thus, such a tree T can be described by means of a boolean vector $z(T)$ indexed by anchored labels X_s, where $z(T)_{X_s} = 1$ if X_s is in T and 0 otherwise.

We will differentiate the set of *natural* non-terminals that occur in the treebanks from the set of *artificial* non-terminals that do not occur in the treebank and are the results of a binarization with markovization. As these artificial non-terminals disappear after reversing binarization in solution trees, they do not play any role in the consensus between parsers, and we only consider natural non-terminals in the set of anchored labels.

When the grammar is functionally annotated, each label $\bar{X}$ in a tree is a pair (X, F), where X is a syntactical category and F is a function label. In this case, in order to manage the consensus with non-functional grammars, we decompose such a tree into two sets: a set of anchored categories X_s and a set of anchored function labels F_s. Thus, a tree T can be described by means of two boolean vectors:

- $z(T)$ indexed by anchored categories X_s, $z(T)_{X_s} = 1$ if there exists a function label F so that $(X, F)_s$ is in T, and 0 otherwise;
- $\zeta(T)$ indexed by anchored function labels F_s, $\zeta(T)_{F_s} = 1$ if there exists a category X so that $(X, F)_s$ is in T, and 0 otherwise.

In the present work, a compound parse $(T_1 \dots T_n)$ is said to be consistent iff every tree shares the same set of *anchored categories*, i.e. iff:

$$\forall (i,j) \in [\![1,n]\!]^2,\ z(T_i) = z(T_j)$$

4.2 Combining Parsers through Dual Decomposition

Like previous applications, we base our reasoning on the assumption that computing the optimal score with each grammar G_i can be efficiently calculated, which is the case for approximate PCFG-LA parsing. We follow the presentation of the decomposition from (Martins et al., 2011) to explain how we can combine several PCFG-LA parsers together.

For a sentence s, we want to obtain the best consistent compound parse from a set of n parsers:

$$(P): \quad \textit{find} \quad \arg\max_{(T_1 \dots T_n) \in \mathcal{C}} \sum_{p=1}^{n} \sigma_p(T_p) \tag{7}$$

$$\textit{s.t.} \ \forall (i,j) \in [\![1,n]\!]^2,\ z(T_i) = z(T_j) \tag{8}$$

where $\mathcal{C} = \mathcal{F}_1(s) \times \dots \times \mathcal{F}_n(s)$ is the product of parse forests $\mathcal{F}_i(s)$, and $\mathcal{F}_i(s)$ is the set of trees in grammar G_i whose yields are the input sentence s.

Solving this problem with an exact algorithm is intractable. While artificial nodes could be inferred using a traditional parsing algorithm based on dynamic programming (i.e. CKY), the natural nodes require a coupling of the parsers' items to enforce the fact that natural daughter nodes must be identical (or equivalent) with the same spans for all parsers. Since the debinarization of markovized rules enables the creation of arbitrarily long n-ary rules, in the worst case the number of natural daughters to check is exponential in the size of the span to infer. Even if

[3] The anchor s of a label is composed of the span (i, j), denoting that the label covers terminals of the input sentence from index i to index j. In case the grammar contains unary non-lexical rules, the anchor also discriminates the different positions in a sequence of unary rules.

we bound the length of debinarized rules, the problem is hardly tractable.

As this problem is intractable, even for approximate PCFG-LA parsing, we apply the iterate method presented in (Komodakis et al., 2007) for MRFs, also applied for joint tasks in NLP such as combined parsing and POS tagging in (Rush et al., 2010).

First, we introduce a *witness* vector u in order to simplify constraints in (8). Problem (P) can then be written in an equivalent form :

$$(P): \quad \textit{find} \quad o_P = \max_{(T_1 \dots T_n) \in \mathcal{C}} \sum_{i=1}^{n} \sigma_i(T_i) \tag{9}$$

$$\textit{s.t.} \quad \forall i \in [\![1, n]\!],\ z(T_i) = u \tag{10}$$

Next, we proceed to a Lagrangian decomposition. This decomposition is a two-step process:

Step 1 (Relaxation): the coupling constraints (10) are removed by introducing a vector of Lagrange multipliers $\Lambda_i = (\lambda_{i,X_s})_{X_s}$ for each parser i, indexed by anchored categories X_s, and writing the equivalent problem:

$$(RP): \quad \mathrm{o}_{RP} = \max_{u, T_{1\dots n}} \min_{\Lambda} f(u, T_{1\dots n}, \Lambda)$$

where:

$$f(u, T_{1\dots n}, \Lambda) = \sum_i \sigma_i(T_i) + \sum_i (z(T_i) - u) \cdot \Lambda_i$$

Intuitively, we can see the equivalence of (RP) and (P) with the following reasoning:

- whenever all constraints (10) are met, the second sum in f is nullified and $f(u, T_{1\dots n}, \Lambda) = \sum_i \sigma_i(T_i)$, which is a finite value and precisely the objective function maximized in (P);
- if there is at least one (i, X, s) such that $z(T_i)_{X_s} \neq u_{X_s}$, then the value of $\sum_i (z(T_i) - u) \cdot \Lambda_i$ can be made arbitrarily small by an appropriate choice of λ_{i,X_s}; in this case, $\min_\Lambda f(u, T_{1\dots n}, \Lambda) = -\infty$. Thus, (RP) can not reach its maximum at a point where constraints (10) are not satisfied.

Step 2 (dualization): the dual problem (LP) is obtained by permuting $\max$ and $\min$ in (RP):

$$(LP1): \quad \mathrm{o}_{LP} = \min_{\Lambda} \max_{u, T_{1\dots n}} f(u, T_{1\dots n}, \Lambda)$$

Finally, u can be removed from $(LP1)$ by adding the constraint: $\sum_i \Lambda_i = 0$. As a matter of fact, one can see that if this constraint is not matched, $\max_{u, T_{1\dots n}} f(u, T_{1\dots n}, \Lambda) = +\infty$ and $(LP1)$ can not reach its minimum on such a point. We can now find the maximum of f by maxing each T_i independently of each other. The dual problem becomes:

$$(LP): \quad \mathrm{o}_{LP} = \min_{\Lambda} \sum_{i=1}^{n} \max_{T_i \in \mathcal{F}_i} (\sigma_i(T_i) + z(T_i) \cdot \Lambda_i)$$

$$s.t. \quad \sum_i \Lambda_i = 0$$

Minimization in (LP) can be solved iteratively using the projected subgradient method. Finding a subgradient amounts to computing the optimal solution (Rush and Collins, 2012) for each of the n subproblems (the *slave* problems in the terminology of (Martins et al., 2011) and (Komodakis et al., 2007)) which can be done efficiently, by incorporating the calculation of the penalties in the parsing algorithm, and in parallel. Until the agreement constraints are met (or a maximal number of iterations τ), the Lagrangian multipliers are updated according to the deviations from the average solutions (i.e. updates are zeros for a natural span if the parsers agree on it). This leads to Algorithm 1.

It should be noted that the DP charts are built and pruned during the first iteration only ($t = 0$); further iterations do not require recreating the DP chart, which is memory intensive and time consuming, nor recomputing the approximate distribution for variational inference. As DP on the pruned charts is a fast process, the bottleneck of the algorithm still is in the first calculation of slave solutions.

The stepsize sequence $(\alpha_t)_{0 \leq t}$ must be diminishing and non-summable, that is to say: $\forall t, \alpha_t \geq 0$, $\lim_{t \to \infty} \alpha_t = 0$ and $\sum_{t=0}^{\infty} \alpha_t = \infty$. In practice, we set $\alpha_t = \frac{1}{1 + c(t)}$ where $c(t)$ is the number of times the objective function o_P has increased since iterations began.

Solving $(\mathbf{P})$: it is easy to see that o_{LP} is an upper bound of o_P, but we do not necessarily have

Algorithm 1 Find best compound parse with constraints on natural spans

Require: n parsers $\{p_i\}_{1 \leq i \leq n}$
  for all i, syntactical category X, anchor s **do**
    $\lambda^{(0)}_{i,X_s} = 0$
  end for
  for $t = 0 \rightarrow \tau$ **do**
    for all parsers p_i **do**
    $T_i^{(t)} \leftarrow \arg\max_{T \in \mathcal{F}_i} \left(\sigma_i(T) + z(T) \cdot \Lambda_i^{(t)}\right)$
    end for
    for all parsers p_i **do**
      $\Delta_i^{(t)} \leftarrow \alpha_t \left(z\left(T_i^{(t)}\right) - \frac{\sum_{1 \leq j \leq n} z\left(T_j^{(t)}\right)}{n} \right)$
      $\Lambda_i^{(t+1)} \leftarrow \Lambda_i^{(t)} + \Delta_i^{(t)}$
    end for
    if $\Delta_i^{(t)} = 0$ for all i **then**
      Exit loop
    end if
  end for
  return $(T_1^{(\tau)}, \cdots, T_n^{(\tau)})$

strong duality (i.e. $o_{LP} = o_P$) due to the facts that parse forests are discrete sets. Furthermore, they get pruned independently of each other. Thus, the algorithm is not guaranteed to find a t such that $z(T_i^{(t)})$ is the same for every parser i. However – see (Koo et al., 2010) – if it does reach such a state, then we have the guarantee of having found an exact solution of the primal problem (P). We show in the experiments that this occurs very frequently.

5 Experiments

5.1 Experimental Setup

We perform our experiments on the WSJ sections of the PTB with the usual split: sections 2 to 21 for training, section 23 for testing, and we run benchmarks on section 22. `evalb` is used for evaluation.

We use the LORG parser modified with Algorithm 1. [4] All grammars are trained using 6 split/merge EM cycles. For the handling of unknown words, we removed all words occurring once in the training set and replaced them by their morphological signature (Attia et al., 2010). Grammars for products are obtained by training with 16 random seeds for each setting. We use the approximate algorithm MaxRule-Product (Petrov and Klein, 2007).

The basic settings are a combination of the two following parameters:

left or right binarization: we conjecture that this affects the quality of the parsers by impacting the recognition of left and right constituent frontiers. We set vertical markovization to 1 (no parent annotation) and horizontal markovization to 0 (we drop all left/right annotations).

with or without functional annotations: in particular when non-terminals are annotated with multiple functions, all are kept.

5.2 Products of Grammars

We first evaluate each setting on its own before combining them. We test the 4 different settings on the development set, using a single grammar or a product of n grammars. Results are reported on Figure 3. We can see that right binarization performs better than left binarization. Contrary to the results of Gabbard et al. (2006), function labels are detrimental for parsing performance for one grammar only. However, they do not penalize performance when using the product model with 8 grammars or more.

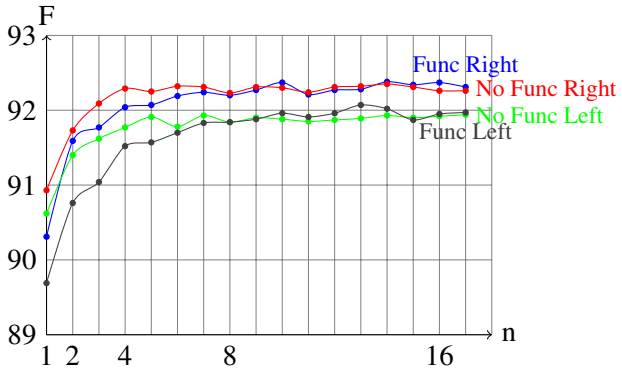


Figure 3: F1 for products of n grammars on the dev. set

EM is not guaranteed to find the optimal model and the problem is made harder by the increased number of parameters. Product models effectively alleviate this *curse of dimensionality* by letting some models compensate for the errors made by others.

On the other hand, as differences between left and right binarization settings remain over all product sizes, right binarization seems more useful on its own. The first part of Table 1 gives F-score and

[4] The LORG parser is available at `https://github.com/CNGLdlab/LORG-Release` and the modification at `https://github.com/jihelhere/LORG-Release/tree/functional_c11`.

Exact Match results of the product models with 16 grammars on the development set.

5.3 Combinations with Dual Decomposition

We now turn to a series of experiments combining product models of 16 grammars. In all these experiments, we set the maximum number of iterations in Algorithm 1 to 1000. The system then returns the first element of the c-parse. We first try to combine two settings in four different combinations:

DD Right Bin the two right-binarized systems – with and without functions – the system returns the function-labeled parse;

DD Left Bin the two left-binarized systems – with and without functions – the system returns the function-labeled parse;

DD Func the two systems with functions – left and right binarization – the system returns the right-binarized parse;

DD No Func the two systems without functions – left and right binarization – the system returns the right-binarized parse;

Results are in the second part of Table 1. Unsurprisingly, the best configuration is the one combining the two best product systems (with right binarization) but all combined systems perform better than their single components.

Setting	F	EX
No Func Right	92.26	42.97
No Func Left	91.92	42.91
Func Right	92.37	43.35
Func Left	91.95	43.15
DD Right Bin	92.71	44.44
DD Left Bin	92.23	43.97
DD Func	92.51	44.79
DD No Func	92.52	44.08
DD3	92.86	45.03
DD4	92.82	45.14

Table 1: Parse evaluation on development set.

We also combine 3 and 4 parsers to see if combining the above **DD Right Bin** setting with information that could improve the recognition of beginning of constituents can be helpful. We have 2 settings:

DD3 The 2 right-binarized parsers combined with the left binarized parser without functions,

DD4 The 4 parsers together.

In both cases the system returns the right-binarized function annotated parse. The results are shown in the last part of Table 1. These 2 new configurations give similar F-scores, better than all 2-parser configurations.

We conclude from these results that left-binarization and right-binarization capture different linguistic aspects, even in the case of heavy horizontal markovization, and that the method we propose enables a practical integration of these models.

Table 2 shows for each setting how often the systems agree before 1000 iterations of Algorithm 1. As one might expect, the more diverse the systems are, the lower the rate of agreement.

Setting	Rate
DD Right Bin	99.24
DD Left Bin	99.12
DD Func	98.53
DD No Func	99.12
DD3	96.18
DD4	94.53

Table 2: Rate of certificates of optimality on the dev set.

5.4 Evaluation of Function Labeling

We also evaluate the quality of the function labels. We compare the results obtained directly from the parser output with results obtained with Funtag, a state-of-the-art functional tagger that is applied on parser output, using a *gold model* trained on sections 02 to 21 of the WSJ (Chrupala et al., 2007).

Setting	SYSTEM FUN	FUNTAG
No Func Right	–	90.41
No Func Left	–	90.26
Func Right	89.61	90.37
Func Left	89.29	90.40
DD Right Bin	89.50	90.38
DD Left Bin	89.11	90.31
DD Func	89.54	90.49
DD No Func	–	90.36
DD3	89.48	90.42
DD4	89.57	90.45

Table 3: Function labeling F1 on development set.

The results are shown in Table 3. First, we can see that the parser output is always outperformed by Funtag. This is expected from a context-free parser

that has a limited domain of locality with strong independence constraints, compared to a voted-SVM classifier that can rely on arbitrarily rich features. Second, the quality of the Funtag prediction seems to be influenced by the fact that parser already handle functions and by the accuracy of the parser (Parseval F-score). This is because we use a model trained on the gold reference and so the closer the parser output is from the reference, the better the prediction. On the other hand, this is not the case with parser predicted functions, where the best system is the right-binarized product model with functions, with very similar performance obtained by the combinations consisting of 2 function parsers, settings **DD Func** and **DD4**. This tends to indicate that the constraints we have set to define consistencies in c-parses, focusing on syntactical categories, do not help in retrieving better function labels. This suggests some possible further improvements where parsers with functional annotations should be forced to agree on these too.

5.5 Evaluation of Dependencies

Setting	Stanford		LTH		p2m
	LAS	UAS	LAS	UAS	UAS
Func Right	92.18	94.32	89.51	93.92	94.2
No Func Right	92.03	94.47	65.31	92.22	94.2
Func Left	91.86	94.06	89.28	93.75	93.9
No Func Left	91.83	94.29	65.33	92.18	94.1
DD Right Bin	92.56	94.60	89.81	94.17	94.5
DD Left Bin	92.01	94.38	89.62	94.05	94.2
DD Func	92.19	94.36	89.67	94.06	94.2
DD No Func	92.19	94.57	65.44	92.37	94.3
DD3	92.77	94.79	90.04	94.33	94.5
DD4	92.59	94.62	89.95	94.24	94.4

Table 4: Dependency accuracies on the dev set

Dependency-based evaluation of phrase structure parser output has been used in recent years to provide a more rounded view on parser performance and to compare with direct dependency parsers (Cer et al., 2010; Petrov et al., 2010; Nivre et al., 2010; Foster et al., 2011; Petrov and McDonald, 2012). We evaluate our various parsing models on their ability to recover three types of dependencies: basic Stanford dependencies (de Marneffe and Manning, 2008)[5], LTH dependencies (Johansson and Nugues, 2007)[6] and *penn2malt* dependencies.[7] The latter are a simpler version of the LTH dependencies but are still used when reporting unlabeled attachment scores for dependency parsing.

The results, shown in Table 4, mirror the constituency evaluation results in that the dual decomposition results tend to outperform the basic product model results, and combining three or four grammars using dual decomposition yields the highest scores. The differences between the **Func** and **No Func** results highlight an important difference between the Stanford and LTH dependency schemes. The tool used to produce Stanford dependencies has been designed to work with phrase structure trees that do not contain function labels. In contrast, the LTH tool makes use of function label information in phrase structure trees. Thus, their availability results in only a moderate improvement in LAS for the Stanford dependencies and a very striking improvement for the LTH dependencies. By retaining function labels during parsing, we have shown that LTH dependencies can be recovered with a high level of accuracy without having to resort to a post-parsing function labeling step.

5.6 Test Set Results

We now evaluate our various systems on the test set (the first half of Table 5) and compare these results with state-of-the-art systems (the second half of Table 5). We present parser accuracy results, measured using Parseval F-score and *penn2malt* UAS, and, for our systems, function label accuracy for labels produced during parsing and after parsing using Funtag. We also carried out statistical significance testing[8] on the F-score differences between our various systems on the development and test sets. The results

[5]We used the latest version at the time of writing, i.e. 3.20.

[6]`nlp.cs.lth.se/software/treebank_converter`. It is recommended that LTH is used with the version of the Penn Treebank which contains the more detailed NP bracketing provided by Vadas and Curran (2007). However, to facilitate comparison with other parsers and dependency schemes, we did not use it in our experiments. We ran the converter with the *rightBranching=false* option to indicate that we are using the version without extra noun phrase bracketing.

[7]`stp.lingfil.uu.se/~nivre/research/Penn2Malt`. The English head-finding rules of Yamada and Matsumoto (2003), supplied on the website, are employed.

[8]We used Dan Bikel's `compare.pl` script which uses stratified shuffling to compute significance. We consider a p value < 0.05 to indicate a statistically significant difference.

Setting	F	UAS	Fun	Funtag
Func Right	91.73	93.9	91.02	91.88
No Func Right	91.76	93.8	–	91.80
Func Left	91.45	93.7	90.41	91.80
No Func Left	91.57	93.7	–	91.74
DD Right Bin	92.16	94.1	90.85	91.86
DD Left Bin	91.89	93.9	90.10	91.85
DD Func	92.23	94.1	91.02	91.91
DD No Func	92.09	94.0	–	91.86
DD3	92.45	94.3	90.86	91.98
DD4	92.44	94.3	90.97	92.04
(Shindo et al., 2012)	92.4			
(Zhang et al., 2009)	92.3			
(Petrov, 2010)	91.8			
(Huang, 2008)	91.7			
(Bohnet and Nivre, 2012)		93.7		

Table 5: Test Set Results: Parseval F-score, *penn2malt* UAS, Function Label Accuracy and Funtag Function Label Accuracy

are shown in Table 6.

Comparison	Dev	Test
Func Right vs. No Func Right	✗	✗
Func Left vs. No Func Left	✗	✗
Func Right vs. Func Left	✓	✗
No Func Right vs. No Func Left	✗	✗
DD Right Bin vs. Func Right	✓	✓
DD Right Bin vs. No Func Right	✓	✓
DD Left Bin vs. Func Left	✓	✓
DD Left Bin vs. No Func Left	✓	✓
DD Right Bin vs DD Left Bin	✓	✓
DD Func vs. Func Right	✗	✓
DD Func vs. Func Left	✓	✓
DD No Func vs. No Func Right	✓	✓
DD No Func vs. No Func Left	✓	✓
DD Func vs. DD No Func	✗	✗
DD3 vs. DD Right Bin	✗	✓
DD3 vs. No Func Left	✓	✓
DD3 vs. DD Func	✓	✓
DD4 vs. DD. Right Bin	✗	✓
DD4 vs. DD. Left Bin	✓	✓
DD4 vs. DD Func	✓	✓
DD4 vs. DD3	✗	✗

Table 6: Statistical Significance Testing

We measured the performance of DD4 on the test set. It is approximately 3 times slower than the slowest product model (left binarization with function labels) and 7 slower than the fastest one (right binarization without function labels). This system performs on average 85.5 iterations of the DD algorithm. If we exclude the non-converging cases (5.1% of the cases), this drops to 39.4.

Finally we compare our results with systems trained and evaluated on the PTB, see the lower half of Table 5. Our product models are not different from those presented in (Petrov, 2010) and it is not surprising to see that the F-scores are similar. More interestingly our DD4 setting improves on these results and compares favorably with systems relying on richer syntactic information, such as the discriminative parser of (Huang, 2008) that makes use of non-local features to score trees and the TSG parser of (Shindo et al., 2012) that can take into account larger tree fragments: this would indicate that by combining our parsers we extend the domain of locality, horizontally with binarization schemes and vertically with function labels. Our system also performs better than the combination system presented in (Zhang et al., 2009) that only relies on material from the PTB[9] but a more detailed comparison is difficult: this system does not use products of latent models and more generally their method is orthogonal to ours. We also include for comparison state-of-the-art dependency parsing results (Bohnet and Nivre, 2012).

6 Conclusion

We presented an algorithm and a set of experiments showing that grammar extraction strategies can be combined in an elegant way and give state-of-the-art results when applied to high-quality phrase-based parsers. As well as repeating these experiments for languages which rely more on function annotation, we also plan to apply our method to other types of annotations, e.g. more linguistically motivated binarization strategies or – of particular interest to us – annotation of empty elements.

Acknowledgments

We are grateful to the reviewers for their helpful comments. We also thank Joachim Wagner for providing feedback on an early version of the paper. This work has been partially funded by the Labex EFL (ANR/CGI).

[9] Their other system relying on the self-trained version of the BLLIP parser achieves 92.6 F1.

References

Mohammed Attia, Jennifer Foster, Deirdre Hogan, Joseph Le Roux, Lamia Tounsi, and Josef van Genabith. 2010. Handling unknown words in statistical latent-variable parsing models for Arabic, English and French. In *Proceedings of the First Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st Annual Meeting of the North American chapter of the ACL*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC*.

Grzegorz Chrupala, Nicolas Stroppa, Josef van Genabith, and Georgiana Dinu. 2007. Better training for function labeling. In *Proceedings of the 2007 Conference on Recent Advances in Natural Language Processing (RANLP)*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From news to comment: Resources and benchmarks for parsing the language of web 2.0. In *Proceedings of IJCNLP*.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the penn treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 184–191.

David Hall and Dan Klein. 2012. Training factored PCFGs with expectation propagation. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, pages 649–652.

John C. Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processing*, pages 187–194.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek, and Mare Koit, editors, *Proceedings of NODALIDA 2007*, pages 105–112.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the 1994 ARPA Speech and Natural Language Workshop*, pages 114–119.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 238–249.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82.

Paola Merlo and Gabriele Musillo. 2005. Accurate function parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 620–627.

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

Joakim Nivre, Laura Rimell, Ryan Mc Donald, and Carlos Gómez-Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of COLING*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the conference on Human Language Technologies and the conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'07)*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Working Notes of the SANCL Workshop (NAACL-HLT)*.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of EMNLP*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Proceedings of the conference on Human Language Technologies and the conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'10)*, pages 19–27.

Alexander Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 1044–1050.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 440–448.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, May.

Mark Steedman, Miles Osbourne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL*, pages 759–763.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of the conference on Human Language Technologies and the conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'10)*, pages 649–652.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of ACL*, pages 240–247.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560.

Feature Noising for Log-linear Structured Prediction

Sida I. Wang*, Mengqiu Wang*, Stefan Wager†,
Percy Liang, Christopher D. Manning
Department of Computer Science, †Department of Statistics
Stanford University, Stanford, CA 94305, USA
{sidaw, mengqiu, pliang, manning}@cs.stanford.edu
swager@stanford.edu

Abstract

NLP models have many and sparse features, and regularization is key for balancing model overfitting versus underfitting. A recently repopularized form of regularization is to generate fake training data by repeatedly adding noise to real data. We reinterpret this noising as an explicit regularizer, and approximate it with a second-order formula that can be used during training without actually generating fake data. We show how to apply this method to structured prediction using multinomial logistic regression and linear-chain CRFs. We tackle the key challenge of developing a dynamic program to compute the gradient of the regularizer efficiently. The regularizer is a sum over inputs, so we can estimate it more accurately via a semi-supervised or transductive extension. Applied to text classification and NER, our method provides a $>1\%$ absolute performance gain over use of standard L_2 regularization.

1 Introduction

NLP models often have millions of mainly sparsely attested features. As a result, balancing overfitting versus underfitting through good weight regularization remains a key issue for achieving optimal performance. Traditionally, L_2 or L_1 regularization is employed, but these simple types of regularization penalize all features in a uniform way without taking into account the properties of the actual model.

An alternative approach to regularization is to generate fake training data by adding random noise to the input features of the original training data. Intuitively, this can be thought of as simulating missing features, whether due to typos or use of a previously unseen synonym. The effectiveness of this technique is well-known in machine learning (Abu-Mostafa, 1990; Burges and Schölkopf, 1997; Simard et al., 2000; Rifai et al., 2011a; van der Maaten et al., 2013), but working directly with many corrupted copies of a dataset can be computationally prohibitive. Fortunately, feature noising ideas often lead to tractable deterministic objectives that can be optimized directly. Sometimes, training with corrupted features reduces to a special form of regularization (Matsuoka, 1992; Bishop, 1995; Rifai et al., 2011b; Wager et al., 2013). For example, Bishop (1995) showed that training with features that have been corrupted with additive Gaussian noise is equivalent to a form of L_2 regularization in the low noise limit. In other cases it is possible to develop a new objective function by marginalizing over the artificial noise (Wang and Manning, 2013; van der Maaten et al., 2013).

The central contribution of this paper is to show how to efficiently simulate training with artificially noised features in the context of log-linear structured prediction, without actually having to generate noised data. We focus on dropout noise (Hinton et al., 2012), a recently popularized form of artificial feature noise where a random subset of features is omitted independently for each training example. Dropout and its variants have been shown to outperform L_2 regularization on various tasks (Hinton et al., 2012; Wang and Manning, 2013; Wan et al., 2013). Dropout is is similar in spirit to feature bagging in the deliberate removal of features, but performs the removal in a preset way rather than randomly (Bryll et al., 2003; Sutton et al., 2005; Smith et al., 2005).

*Both authors contributed equally to the paper

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1170–1179,
Seattle, Washington, USA, 18-21 October 2013.

Our approach is based on a second-order approximation to feature noising developed among others by Bishop (1995) and Wager et al. (2013), which allows us to convert dropout noise into a form of adaptive regularization. This method is suitable for structured prediction in log-linear models where second derivatives are computable. In particular, it can be used for multiclass classification with maximum entropy models (a.k.a., softmax or multinomial logistic regression) and for the sequence models that are ubiquitous in NLP, via linear chain Conditional Random Fields (CRFs).

For linear chain CRFs, we additionally show how we can use a noising scheme that takes advantage of the clique structure so that the resulting noising regularizer can be computed in terms of the pairwise marginals. A simple forward-backward-type dynamic program can then be used to compute the gradient tractably. For ease of implementation and scalability to semi-supervised learning, we also outline an even faster approximation to the regularizer. The general approach also works in other clique structures in addition to the linear chain when the clique marginals can be computed efficiently.

Finally, we extend feature noising for structured prediction to a transductive or semi-supervised setting. The regularizer induced by feature noising is label-independent for log-linear models, and so we can use unlabeled data to learn a better regularizer. NLP sequence labeling tasks are especially well suited to a semi-supervised approach, as input features are numerous but sparse, and labeled data is expensive to obtain but unlabeled data is abundant (Li and McCallum, 2005; Jiao et al., 2006).

Wager et al. (2013) showed that semi-supervised dropout training for logistic regression captures a similar intuition to techniques such as entropy regularization (Grandvalet and Bengio, 2005) and transductive SVMs (Joachims, 1999), which encourage confident predictions on the unlabeled data. Semi-supervised dropout has the advantage of only using the predicted label probabilities on the unlabeled data to modulate an L_2 regularizer, rather than requiring more heavy-handed modeling of the unlabeled data as in entropy regularization or expectation regularization (Mann and McCallum, 2007).

In experimental results, we show that simulated feature noising gives more than a 1% absolute boost in performance over L_2 regularization, on both text classification and an NER sequence labeling task.

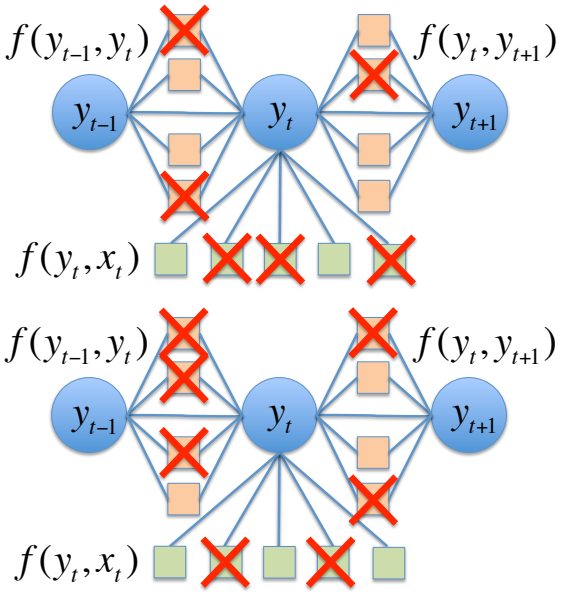


Figure 1: An illustration of dropout feature noising in linear-chain CRFs with only transition features and node features. The green squares are node features $f(y_t, x_t)$, and the orange squares are edge features $f(y_{t-1}, y_t)$. Conceptually, given a training example, we sample some features to ignore (generate fake data) and make a parameter update. Our goal is to train with a roughly equivalent objective, without actually sampling.

2 Feature Noising Log-linear Models

Consider the standard structured prediction problem of mapping some input $x \in \mathcal{X}$ (e.g., a sentence) to an output $\mathbf{y} \in \mathcal{Y}$ (e.g., a tag sequence). Let $f(\mathbf{y}, x) \in \mathbb{R}^d$ be the feature vector, $\theta \in \mathbb{R}^d$ be the weight vector, and $\mathbf{s} = (s_1, \ldots, s_{|\mathcal{Y}|})$ be a vector of scores for each output, with $s_{\mathbf{y}} = f(\mathbf{y}, x) \cdot \theta$. Now define a log-linear model:

$$p(\mathbf{y} \mid x; \theta) = \exp\{s_{\mathbf{y}} - A(\mathbf{s})\}, \qquad (1)$$

where $A(\mathbf{s}) = \log \sum_{\mathbf{y}} \exp\{s_{\mathbf{y}}\}$ is the log-partition function. Given an example $(x, \mathbf{y})$, parameter estimation corresponds to choosing θ to maximize $p(\mathbf{y} \mid x; \theta)$.

The key idea behind feature noising is to artificially corrupt the feature vector $f(\mathbf{y}, x)$ randomly

into some $\tilde{f}(\mathbf{y}, x)$ and then maximize the average log-likelihood of $\mathbf{y}$ given these corrupted features—the motivation is to choose predictors θ that are robust to noise (missing words for example). Let $\tilde{\mathbf{s}}$, $\tilde{p}(\mathbf{y} \mid x; \theta)$ be the *randomly* perturbed versions corresponding to $\tilde{f}(\mathbf{y}, x)$. We will also assume the feature noising preserves the mean: $\mathbb{E}[\tilde{f}(\mathbf{y}, x)] = f(\mathbf{y}, x)$, so that $\mathbb{E}[\tilde{\mathbf{s}}] = \mathbf{s}$. This can always be done by scaling the noised features as described in the list of noising schemes.

It is useful to view feature noising as a form of regularization. Since feature noising preserves the mean, the feature noising objective can be written as the sum of the original log-likelihood plus the difference in log-normalization constants:

$$\mathbb{E}[\log \tilde{p}(\mathbf{y} \mid x; \theta)] = \mathbb{E}[\tilde{\mathbf{s}}_{\mathbf{y}} - A(\tilde{\mathbf{s}})] \quad (2)$$

$$= \log p(\mathbf{y} \mid x; \theta) - R(\theta, x), \quad (3)$$

$$R(\theta, x) \stackrel{\text{def}}{=} \mathbb{E}[A(\tilde{\mathbf{s}})] - A(\mathbf{s}). \quad (4)$$

Since $A(\cdot)$ is convex, $R(\theta, x)$ is always positive by Jensen's inequality and can therefore be interpreted as a regularizer. Note that $R(\theta, x)$ is in general non-convex.

Computing the regularizer (4) requires summing over all possible noised feature vectors, which can imply exponential effort in the number of features. This is intractable even for flat classification. Following Bishop (1995) and Wager et al. (2013), we approximate $R(\theta, x)$ using a second-order Taylor expansion, which will allow us to work with only means and covariances of the noised features. We take a quadratic approximation of the log-partition function $A(\cdot)$ of the noised score vector $\tilde{\mathbf{s}}$ around the the unnoised score vector $\mathbf{s}$:

$$A(\tilde{\mathbf{s}}) \approxeq A(\mathbf{s}) + \nabla A(\mathbf{s})^\top (\tilde{\mathbf{s}} - \mathbf{s}) \quad (5)$$

$$+ \frac{1}{2}(\tilde{\mathbf{s}} - \mathbf{s})^\top \nabla^2 A(\mathbf{s})(\tilde{\mathbf{s}} - \mathbf{s}).$$

Plugging (5) into (4), we obtain a new regularizer $R^{\mathrm{q}}(\theta, x)$, which we will use as an approximation to $R(\theta, x)$:

$$R^{\mathrm{q}}(\theta, x) = \frac{1}{2}\mathbb{E}[(\tilde{\mathbf{s}} - \mathbf{s})^\top \nabla^2 A(\mathbf{s})(\tilde{\mathbf{s}} - \mathbf{s})] \quad (6)$$

$$= \frac{1}{2} \operatorname{tr}(\nabla^2 A(\mathbf{s}) \operatorname{Cov}(\tilde{\mathbf{s}})). \quad (7)$$

This expression still has two sources of potential intractability, a sum over an exponential number of noised score vectors $\tilde{\mathbf{s}}$ and a sum over the $|\mathcal{Y}|$ components of $\tilde{\mathbf{s}}$.

Multiclass classification If we assume that the components of $\tilde{\mathbf{s}}$ are independent, then $\operatorname{Cov}(\tilde{\mathbf{s}}) \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{Y}|}$ is diagonal, and we have

$$R^{\mathrm{q}}(\theta, x) = \frac{1}{2} \sum_{\mathbf{y} \in \mathcal{Y}} \mu_{\mathbf{y}}(1 - \mu_{\mathbf{y}}) \operatorname{Var}[\tilde{s}_{\mathbf{y}}], \quad (8)$$

where the mean $\mu_{\mathbf{y}} \stackrel{\text{def}}{=} p_\theta(\mathbf{y} \mid x)$ is the model probability, the variance $\mu_{\mathbf{y}}(1 - \mu_{\mathbf{y}})$ measures model uncertainty, and

$$\operatorname{Var}[\tilde{s}_{\mathbf{y}}] = \theta^\top \operatorname{Cov}[\tilde{f}(\mathbf{y}, x)]\theta \quad (9)$$

measures the uncertainty caused by feature noising.[1] The regularizer $R^{\mathrm{q}}(\theta, x)$ involves the product of two variance terms, the first is non-convex in θ and the second is quadratic in θ. Note that to reduce the regularization, we will favor models that (i) predict confidently and (ii) have stable scores in the presence of feature noise.

For multiclass classification, we can explicitly sum over each $\mathbf{y} \in \mathcal{Y}$ to compute the regularizer, but this will be intractable for structured prediction.

To specialize to multiclass classification for the moment, let us assume that we have a separate weight vector for each output $\mathbf{y}$ applied to the same feature vector $g(x)$; that is, the score $s_{\mathbf{y}} = \theta_{\mathbf{y}} \cdot g(x)$. Further, assume that the components of the noised feature vector $\tilde{g}(x)$ are independent. Then we can simplify (9) to the following:

$$\operatorname{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j \operatorname{Var}[g_j(x)]\theta_{yj}^2. \quad (10)$$

Noising schemes We now give some examples of possible noise schemes for generating $\tilde{f}(\mathbf{y}, x)$ given the original features $f(\mathbf{y}, x)$. This distribution affects the regularization through the variance term $\operatorname{Var}[\tilde{s}_{\mathbf{y}}]$.

- *Additive Gaussian*: $\tilde{f}(\mathbf{y}, x) = f(\mathbf{y}, x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_{d \times d})$.

[1] Here, we are using the fact that first and second derivatives of the log-partition function are the mean and variance.

In this case, the contribution to the regularizer from noising is $\mathrm{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j \sigma^2 \theta_{yj}^2$.

- *Dropout*:

  $\tilde{f}(\mathbf{y}, x) = f(\mathbf{y}, x) \odot z$, where $\odot$ takes the elementwise product of two vectors. Here, z is a vector with independent components which has $z_i = 0$ with probability δ, $z_i = \frac{1}{1-\delta}$ with probability $1 - \delta$. In this case, $\mathrm{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j \frac{g_j(x)^2 \delta}{1-\delta} \theta_{yj}^2$.

- *Multiplicative Gaussian*:

  $\tilde{f}(\mathbf{y}, x) = f(\mathbf{y}, x) \odot (1 + \varepsilon)$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_{d \times d})$. Here, $\mathrm{Var}[\tilde{s}_{\mathbf{y}}] = \sum_j g_j(x)^2 \sigma^2 \theta_{yj}^2$. Note that under our second-order approximation $R^{\mathrm{q}}(\theta, x)$, the multiplicative Gaussian and dropout schemes are equivalent, but they differ under the original regularizer $R(\theta, x)$.

2.1 Semi-supervised learning

A key observation (Wager et al., 2013) is that the noising regularizer R (8), while involving a sum over examples, is independent of the output $\mathbf{y}$. This suggests estimating R using unlabeled data. Specifically, if we have n labeled examples $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ and m unlabeled examples $\mathcal{D}_{\text{unlabeled}} = \{u_1, u_2, \ldots, u_n\}$, then we can define a regularizer that is a linear combination the regularizer estimated on both datasets, with α tuning the tradeoff between the two:

$$R_*(\theta, \mathcal{D}, \mathcal{D}_{\text{unlabeled}}) \tag{11}$$
$$\stackrel{\text{def}}{=} \frac{n}{n + \alpha m} \Big(\sum_{i=1}^{n} R(\theta, x_i) + \alpha \sum_{i=1}^{m} R(\theta, u_i) \Big).$$

3 Feature Noising in Linear-Chain CRFs

So far, we have developed a regularizer that works for all log-linear models, but—in its current form—is only practical for multiclass classification. We now exploit the decomposable structure in CRFs to define a new noising scheme which does not require us to explicitly sum over all possible outputs $\mathbf{y} \in \mathcal{Y}$. The key idea will be to noise each local feature vector (which implicitly affects many $\mathbf{y}$) rather than noise each $\mathbf{y}$ independently.

Assume that the output $\mathbf{y} = (y_1, \ldots, y_T)$ is a sequence of T tags. In linear chain CRFs, the feature vector f decomposes into a sum of local feature vectors g_t:

$$f(\mathbf{y}, x) = \sum_{t=1}^{T} g_t(y_{t-1}, y_t, x), \tag{12}$$

where $g_t(a, b, x)$ is defined on a pair of consecutive tags a, b for positions $t - 1$ and t.

Rather than working with a score $s_{\mathbf{y}}$ for each $\mathbf{y} \in \mathcal{Y}$, we define a collection of *local scores* $\mathbf{s} = \{s_{a,b,t}\}$, for each tag pair (a, b) and position $t = 1, \ldots, T$. We consider noising schemes which independently set $\tilde{g}_t(a, b, x)$ for each a, b, t. Let $\tilde{\mathbf{s}} = \{\tilde{s}_{a,b,t}\}$ be the corresponding collection of noised scores.

We can write the log-partition function of these local scores as follows:

$$A(\mathbf{s}) = \log \sum_{y \in \mathcal{Y}} \exp \left\{ \sum_{t=1}^{T} s_{y_{t-1}, y_t, t} \right\}. \tag{13}$$

The first derivative yields the edge marginals under the model, $\mu_{a,b,t} = p_\theta(y_{t-1} = a, y_t = b \mid x)$, and the diagonal elements of the Hessian $\nabla^2 A(\mathbf{s})$ yield the marginal variances.

Now, following (7) and (8), we obtain the following regularizer:

$$R^{\mathrm{q}}(\theta, x) = \frac{1}{2} \sum_{a,b,t} \mu_{a,b,t}(1 - \mu_{a,b,t}) \, \mathrm{Var}[\tilde{s}_{a,b,t}], \tag{14}$$

where $\mu_{a,b,t}(1 - \mu_{a,b,t})$ measures model uncertainty about edge marginals, and $\mathrm{Var}[\tilde{s}_{a,b,t}]$ is simply the uncertainty due to noising. Again, minimizing the regularizer means making confident predictions and having stable scores under feature noise.

Computing partial derivatives So far, we have defined the regularizer $R^{\mathrm{q}}(\theta, x)$ based on feature noising. In order to minimize $R^{\mathrm{q}}(\theta, x)$, we need to take its derivative.

First, note that $\log \mu_{a,b,t}$ is the difference of a restricted log-partition function and the log-partition function. So again by properties of its first derivative, we have:

$$\nabla \log \mu_{a,b,t} = \mathbb{E}_{p_\theta(\mathbf{y} \mid x, y_{t-1}=a, y_t=b)}[f(\mathbf{y}, x)] - \mathbb{E}_{p_\theta(\mathbf{y} \mid x)}[f(\mathbf{y}, x)]. \tag{15}$$

Using the fact that $\nabla \mu_{a,b,t} = \mu_{a,b,t} \nabla \log \mu_{a,b,t}$ and the fact that $\mathrm{Var}[\tilde{s}_{a,b,t}]$ is a quadratic function in θ, we can simply apply the product rule to derive the final gradient $\nabla R^{\mathrm{q}}(\theta, x)$.

3.1 A Dynamic Program for the Conditional Expectation

A naive computation of the gradient $\nabla R^{\mathrm{q}}(\theta, x)$ requires a full forward-backward pass to compute $\mathbb{E}_{p_\theta(\mathbf{y} | y_{t-1}=a, y_t=b, x)}[f(\mathbf{y}, x)]$ for each tag pair (a, b) and position t, resulting in a $\mathcal{O}(K^4 T^2)$ time algorithm.

In this section, we reduce the running time to $\mathcal{O}(K^2 T)$ using a more intricate dynamic program. By the Markov property of the CRF, $\mathbf{y}_{1:t-2}$ only depends on (y_{t-1}, y_t) through y_{t-1} and $\mathbf{y}_{t+1:T}$ only depends on (y_{t-1}, y_t) through y_t.

First, it will be convenient to define the partial sum of the local feature vector from positions i to j as follows:

$$G_{i:j} = \sum_{t=i}^{j} g_t(y_{t-1}, y_t, x). \tag{16}$$

Consider the task of computing the feature expectation $\mathbb{E}_{p_\theta(\mathbf{y} | y_{t-1}=a, y_t=b)}[f(\mathbf{y}, x)]$ for a fixed (a, b, t). We can expand this quantity into

$$\sum_{\mathbf{y}: y_{t-1}=a, y_t=b} p_\theta(\mathbf{y}_{-(t-1:t)} \mid y_{t-1} = a, y_t = b) G_{1:T}.$$

Conditioning on y_{t-1}, y_t decomposes the sum into three pieces:

$$\sum_{\mathbf{y}: y_{t-1}=a, y_t=b} [g_t(y_{t-1} = a, y_t = b, x) + F_t^a + B_t^b],$$

where

$$F_t^a = \sum_{\mathbf{y}_{1:t-2}} p_\theta(\mathbf{y}_{1:t-2} \mid y_{t-1} = a) G_{1:t-1}, \tag{17}$$

$$B_t^b = \sum_{\mathbf{y}_{t+1:T}} p_\theta(\mathbf{y}_{t+1:T} \mid y_t = b) G_{t+1:T}, \tag{18}$$

are the expected feature vectors summed over the prefix and suffix of the tag sequence, respectively. Note that F_t^a and B_t^b are analogous to the forward and backward messages of standard CRF inference, with the exception that they are vectors rather than scalars.

We can compute these messages recursively in the standard way. The forward recurrence is

$$F_t^a = \sum_b p_\theta(y_{t-2} = b \mid y_{t-1} = a) \left[g_t(y_{t-2} = b, y_{t-1} = a, x) + F_{t-1}^b \right],$$

and a similar recurrence holds for the backward messages B_t^b.

Running the resulting dynamic program takes $\mathcal{O}(K^2 T q)$ time and requires $\mathcal{O}(KTq)$ storage, where K is the number of tags, T is the sequence length and q is the number of active features. Note that this is the same order of dependence as normal CRF training, but there is an additional dependence on the number of active features q, which makes training slower.

4 Fast Gradient Computations

In this section, we provide two ways to further improve the efficiency of the gradient calculation based on ignoring long-range interactions and based on exploiting feature sparsity.

4.1 Exploiting Feature Sparsity and Co-occurrence

In each forward-backward pass over a training example, we need to compute the conditional expectations for all features active in that example. Naively applying the dynamic program in Section 3 is $\mathcal{O}(K^2 T)$ for each active feature. The total complexity has to factor in the number of active features, q. Although q only scales linearly with sentence length, in practice this number could get large pretty quickly. For example, in the NER tagging experiments (*cf.* Section 5), the average number of active features per token is about 20, which means $q \simeq 20T$; this term quickly dominates the computational costs. Fortunately, in sequence tagging and other NLP tasks, the majority of features are sparse and they often co-occur. That is, some of the active features would fire and only fire at the same locations in a given sequence. This happens when a particular token triggers multiple rare features.

We observe that all indicator features that only fired once at position t have the same conditional expectations (and model expectations). As a result, we can collapse such a group of features into a single

feature as a preprocessing step to avoid computing identical expectations for each of the features. Doing so on the same NER tagging experiments cuts down q/T from 20 to less than 5, and gives us a 4 times speed up at no loss of accuracy. The exact same trick is applicable to the general CRF gradient computation as well and gives similar speedup.

4.2 Short-range interactions

It is also possible to speed up the method by resorting to approximate gradients. In our case, the dynamic program from Section 3 together with the trick described above ran in a manageable amount of time. The techniques developed here, however, could prove to be useful on larger tasks.

Let us rewrite the quantity we want to compute slightly differently (again, for all a, b, t):

$$\sum_{i=1}^{T} \mathbb{E}_{p_\theta(\mathbf{y}|x, y_{t-1}=a, y_t=b)}[g_i(y_{i-1}, y_i, x)]. \quad (19)$$

The intuition is that conditioned on y_{t-1}, y_t, the terms $g_i(y_{i-1}, y_i, x)$ where i is far from t will be close to $\mathbb{E}_{p_\theta(\mathbf{y}|x)}[g_i(y_{i-1}, y_i, x)]$.

This motivates replacing the former with the latter whenever $|i-k| \geq r$ where r is some window size. This approximation results in an expression which only has to consider the sum of the local feature vectors from $i-r$ to $i+r$, which is captured by $G_{i-r:i+r}$:

$$\begin{aligned} &\mathbb{E}_{p_\theta(\mathbf{y}|y_{t-1}=a, y_t=b, x)}[f(\mathbf{y}, x)] - \mathbb{E}_{p_\theta(\mathbf{y}|x)}[f(\mathbf{y}, x)] \\ &\quad \approx \mathbb{E}_{p_\theta(\mathbf{y}|y_{t-1}=a, y_t=b, x)}[G_{t-r:t+r}] \\ &\qquad - \mathbb{E}_{p_\theta(\mathbf{y}|x)}[G_{t-r:t+r}]. \end{aligned} \quad (20)$$

We can further approximate this last expression by letting $r = 0$, obtaining:

$$g_t(a, b, x) - \mathbb{E}_{p_\theta(\mathbf{y}|x)}[g_t(y_{t-1}, y_t, x)]. \quad (21)$$

The second expectation can be computed from the edge marginals.

The accuracy of this approximation hinges on the lack of long range dependencies. Equation (21) shows the case of $r = 0$; this takes almost no additional effort to compute. However, for some of our experiments, we observed a 20% difference with the real derivative. For $r > 0$, the computational savings are more limited, but the bounded-window method is easier to implement.

Dataset	q	d	K	N_{train}	N_{test}
CoNLL	20	437906	5	204567	46666
SANCL	5	679959	12	761738	82405
20news	81	62061	20	15935	3993
RCV1_4	76	29992	4	9625/2	9625/2
R21578	47	18933	65	5946	2347
TDT2	130	36771	30	9394/2	9394/2

Table 1: Description of datasets. q: average number of non-zero features per example, d: total number of features, K: number of classes to predict, N_{train}: number of training examples, N_{test}: number of test examples.

5 Experiments

We show experimental results on the CoNLL-2003 Named Entity Recognition (NER) task, the SANCL Part-of-speech (POS) tagging task, and several document classification tasks.[2] The datasets used are described in Table 1. We used standard splits whenever available; otherwise we split the data at random into a test set and a train set of equal sizes (RCV1_4, TDT2). CoNLL has a development set of size 51578, which we used to tune regularization parameters. The SANCL test set is divided into 3 genres, namely `answers`, `newsgroups`, and `reviews`, each of which has a corresponding development set.[3]

5.1 Multiclass Classification

We begin by testing our regularizer in the simple case of classification where $\mathcal{Y} = \{1, 2, \ldots, K\}$ for K classes. We examine the performance of the noising regularizer in both the fully supervised setting as well as the transductive learning setting.

In the transductive learning setting, the learner is allowed to inspect the test features at train time (without the labels). We used the method described in Section 2.1 for transductive dropout.

[2]The document classification data are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets` and `http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html`

[3]The SANCL dataset has two additional genres—emails and weblogs—that we did not use, as we did not have access to development sets for these genres.

Dataset	K	None	L_2	Drop	+Test
CoNLL	5	78.03	80.12	80.90	81.66
20news	20	81.44	82.19	83.37	84.71
RCV1$_4$	4	95.76	95.90	96.03	96.11
R21578	65	92.24	92.24	92.24	92.58
TDT2	30	97.74	97.91	98.00	98.12

Table 2: Classification performance and transductive learning results on some standard datasets. None: use no regularization, Drop: quadratic approximation to the dropout noise (8), +Test: also use the test set to estimate the noising regularizer (11).

Dataset	K	L_2	Drop	+Unlabeled
CoNLL	5	91.46	91.81	92.02
20news	20	76.55	79.07	80.47
RCV1$_4$	4	94.76	94.79	95.16
R21578	65	90.67	91.24	90.30
TDT2	30	97.34	97.54	97.89

Table 3: Semisupervised learning results on some standard datasets. A third (33%) of the full dataset was used for training, a third for testing, and the rest as unlabeled.

5.1.1 Semi-supervised Learning with Feature Noising

In the transductive setting, we used test data (without labels) to learn a better regularizer. As an alternative, we could also use unlabeled data in place of the test data to accomplish a similar goal; this leads to a semi-supervised setting.

To test the semi-supervised idea, we use the same datasets as above. We split each dataset evenly into 3 thirds that we use as a training set, a test set and an unlabeled dataset. Results are given in Table 3.

In most cases, our semi-supervised accuracies are lower than the transductive accuracies given in Table 2; this is normal in our setup, because we used less labeled data to train the semi-supervised classifier than the transductive one.[4]

5.1.2 The Second-Order Approximation

The results reported above all rely on the approximate dropout regularizer (8) that is based on a second-order Taylor expansion. To test the validity of this approximation we compare it to the Gaussian method developed by Wang and Manning (2013) on a two-class classification task.

We use the 20-newsgroups `alt.atheism` vs `soc.religion.christian` classification task; results are shown in Figure 2. There are 1427 examples with 22178 features, split evenly and randomly into a training set and a test set.

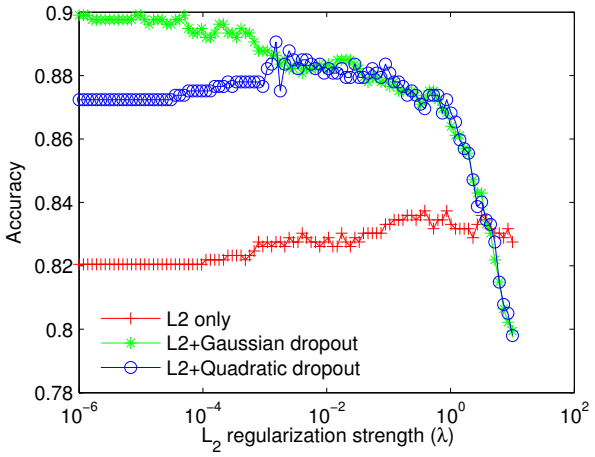


Figure 2: Effect of λ in $\lambda\|\theta\|_2^2$ on the testset performance. Plotted is the test set accuracy with logistic regression as a function of λ for the L_2 regularizer, Gaussian dropout (Wang and Manning, 2013) + additional L_2, and quadratic dropout (8) + L_2 described in this paper. The default noising regularizer is quite good, and additional L_2 does not help. Notice that no choice of λ in L_2 can help us combat overfitting as effectively as (8) without underfitting.

Over a broad range of λ values, we find that dropout plus L_2 regularization performs far better than using just L_2 regularization for any value of λ. We see that Gaussian dropout appears to perform slightly better than the quadratic approximation discussed in this paper. However, our quadratic approximation extends easily to the multiclass case and to structured prediction in general, while Gaussian dropout does not. Thus, it appears that our approximation presents a reasonable trade-off between

[4] The CoNNL results look somewhat surprising, as the semi-supervised results are better than the transductive ones. The reason for this is that the original CoNLL test set came from a different distributions than the training set, and this made the task more difficult. Meanwhile, in our semi-supervised experiment, the test and train sets are drawn from the same distribution and so our semi-supervised task is actually easier than the original one.

computational efficiency and prediction accuracy.

5.2 CRF Experiments

We evaluate the quadratic dropout regularizer in linear-chain CRFs on two sequence tagging tasks: the CoNLL 2003 NER shared task (Tjong Kim Sang and De Meulder, 2003) and the SANCL 2012 POS tagging task (Petrov and McDonald, 2012) .

The standard CoNLL-2003 English shared task benchmark dataset (Tjong Kim Sang and De Meulder, 2003) is a collection of documents from Reuters newswire articles, annotated with four entity types: *Person*, *Location*, *Organization*, and *Miscellaneous*. We predicted the label sequence $\mathcal{Y} = \{\text{LOC, MISC, ORG, PER, O}\}^T$ without considering the BIO tags.

For training the CRF model, we used a comprehensive set of features from Finkel et al. (2005) that gives state-of-the-art results on this task. A total number of 437906 features were generated on the CoNLL-2003 training dataset. The most important features are:

- The word, word shape, and letter n-grams (up to 6gram) at current position
- The prediction, word, and word shape of the previous and next position
- Previous word shape in conjunction with current word shape
- Disjunctive word set of the previous and next 4 positions
- Capitalization pattern in a 3 word window
- Previous two words in conjunction with the word shape of the previous word
- The current word matched against a list of name titles (e.g., Mr., Mrs.)

The $F_{\beta=1}$ results are summarized in Table 4. We obtain a 1.6% and 1.1% absolute gain on the test and dev set, respectively. Detailed results are broken down by precision and recall for each tag and are shown in Table 6. These improvements are significant at the 0.1% level according to the paired bootstrap resampling method of 2000 iterations (Efron and Tibshirani, 1993).

For the SANCL (Petrov and McDonald, 2012) POS tagging task, we used the same CRF framework with a much simpler set of features

- word unigrams: w_{-1}, w_0, w_1
- word bigram: (w_{-1}, w_0) and (w_0, w_1)

$F_{\beta=1}$	None	L_2	Drop
Dev	89.40	90.73	91.86
Test	84.67	85.82	87.42

Table 4: CoNLL summary of results. None: no regularization, Drop: quadratic dropout regularization (14) described in this paper.

$F_{\beta=1}$	None	L_2	Drop
	`newsgroups`		
Dev	91.34	91.34	91.47
Test	91.44	91.44	91.81
	`reviews`		
Dev	91.97	91.95	92.10
Test	90.70	90.67	91.07
	`answers`		
Dev	90.78	90.79	90.70
Test	91.00	90.99	91.09

Table 5: SANCL POS tagging $F_{\beta=1}$ scores for the 3 official evaluation sets.

We obtained a small but consistent improvement using the quadratic dropout regularizer in (14) over the L_2-regularized CRFs baseline.

Although the difference on SANCL is small, the performance differences on the test sets of `reviews` and `newsgroups` are statistically significant at the 0.1% level. This is also interesting because here is a situation where the features are extremely sparse, L_2 regularization gave no improvement, and where regularization overall matters less.

6 Conclusion

We have presented a new regularizer for learning log-linear models such as multiclass logistic regression and conditional random fields. This regularizer is based on a second-order approximation of feature noising schemes, and attempts to favor models that predict confidently and are robust to noise in the data. In order to apply our method to CRFs, we tackle the key challenge of dealing with feature correlations that arise in the structured prediction setting in several ways. In addition, we show that the regularizer can be applied naturally in the semi-supervised setting. Finally, we applied our method to a range of different datasets and demonstrate consistent gains over standard L_2 regularization. Inves-

	Precision	Recall	$F_{\beta=1}$
LOC	91.47%	91.12%	91.29
MISC	88.77%	81.07%	84.75
ORG	85.22%	84.08%	84.65
PER	92.12%	93.97%	93.04
Overall	89.84%	88.97%	89.40

(a) CoNLL dev. set with no regularization

	Precision	Recall	$F_{\beta=1}$
LOC	92.05%	92.84%	92.44
MISC	90.51%	83.52%	86.87
ORG	88.35%	85.23%	86.76
PER	93.12%	94.19%	93.65
Overall	91.36%	90.11%	90.73

(b) CoNLL dev. set with L_2 regularization

	Precision	Recall	$F_{\beta=1}$
LOC	93.59%	92.69%	93.14
MISC	93.99%	81.47%	87.28
ORG	92.48%	84.61%	88.37
PER	94.81%	95.11%	94.96
Overall	93.85%	89.96%	91.86

(c) CoNLL dev. set with dropout regularization

Tag	Precision	Recall	$F_{\beta=1}$
LOC	87.33%	84.47%	85.87
MISC	78.93%	77.12%	78.02
ORG	78.70%	79.49%	79.09
PER	88.82%	93.11%	90.92
Overall	84.28%	85.06%	84.67

(d) CoNLL test set with no regularization

	Precision	Recall	$F_{\beta=1}$
LOC	87.96%	86.13%	87.03
MISC	77.53%	79.30%	78.41
ORG	81.30%	80.49%	80.89
PER	90.30%	93.33%	91.79
Overall	85.57%	86.08%	85.82

(e) CoNLL test set with L_2 regularization

	Precision	Recall	$F_{\beta=1}$
LOC	86.26%	87.74%	86.99
MISC	81.52%	77.34%	79.37
ORG	88.29%	81.89%	84.97
PER	92.15%	92.68%	92.41
Overall	88.40%	86.45%	87.42

(f) CoNLL test set with dropout regularization

Table 6: CoNLL NER results broken down by tags and by precision, recall, and $F_{\beta=1}$. Top: development set, bottom: test set performance.

tigating how to better optimize this non-convex regularizer online and convincingly scale it to the semi-supervised setting seem to be promising future directions.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, or the US government. S. Wager is supported by a BC and EJ Eaves SGF Fellowship.

References

Yaser S. Abu-Mostafa. 1990. Learning from hints in neural networks. *Journal of Complexity*, 6(2):192–198.

Chris M. Bishop. 1995. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7(1):108–116.

Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek. 2003. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern recognition*, 36(6):1291–1302.

Chris J.C. Burges and Bernhard Schölkopf. 1997. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems*, pages 375–381.

Brad Efron and Robert Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics*, pages 363–370.

Yves Grandvalet and Yoshua Bengio. 2005. Entropy regularization. In *Semi-Supervised Learning*, United Kingdom. Springer.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 209–216.

Thorsten Joachims. 1999. Transductive inference for

text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209.

Wei Li and Andrew McCallum. 2005. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the 20th national conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 813–818.

Gideon S. Mann and Andrew McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the International Conference on Machine Learning*.

Kiyotoshi Matsuoka. 1992. Noise injection into inputs in back-propagation learning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(3):436–440.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Salah Rifai, Yann Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. 2011a. The manifold tangent classifier. *Advances in Neural Information Processing Systems*, 24:2294–2302.

Salah Rifai, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. 2011b. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*.

Patrice Y. Simard, Yann A. Le Cun, John S. Denker, and Bernard Victorri. 2000. Transformation invariance in pattern recognition: Tangent distance and propagation. *International Journal of Imaging Systems and Technology*, 11(3):181–197.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 18–25. Association for Computational Linguistics.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2005. Feature bagging: Preventing weight undertraining in structured discriminative learning. *Center for Intelligent Information Retrieval, U. of Massachusetts*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147.

Laurens van der Maaten, Minmin Chen, Stephen Tyree, and Kilian Q. Weinberger. 2013. Learning with marginalized corrupted features. In *Proceedings of the International Conference on Machine Learning*.

Stefan Wager, Sida Wang, and Percy Liang. 2013. Dropout training as adaptive regularization. *arXiv preprint:1307.1493*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine learning*.

Sida Wang and Christopher D. Manning. 2013. Fast dropout training. In *Proceedings of the International Conference on Machine Learning*.

Improvements to the Bayesian Topic N-gram Models

Hiroshi Noji†‡
noji@nii.ac.jp

Daichi Mochihashi†∗
daichi@ism.ac.jp

Yusuke Miyao†‡
yusuke@nii.ac.jp

†Graduate University for Advanced Studies
‡National Institute of Informatics, Tokyo, Japan
∗The Institute of Statistical Mathematics, Tokyo, Japan

Abstract

One of the language phenomena that n-gram language model fails to capture is the topic information of a given situation. We advance the previous study of the Bayesian topic language model by Wallach (2006) in two directions: one, investigating new priors to alleviate the sparseness problem caused by dividing all n-grams into exclusive topics, and two, developing a novel Gibbs sampler that enables moving multiple n-grams across different documents to another topic. Our blocked sampler can efficiently search for higher probability space even with higher order n-grams. In terms of modeling assumption, we found it is effective to assign a topic to only some parts of a document.

1 Introduction

N-gram language model is still ubiquitous in NLP, but due to its simplicity it fails to capture some important aspects of language, such as difference of word usage in different situations, sentence level syntactic correctness, and so on. Toward language model that can consider such a more global context, many extensions have been proposed from lexical pattern adaptation, e.g., adding cache (Jelinek et al., 1991) or topic information (Gildea and Hofmann, 1999; Wallach, 2006), to grammaticality aware models (Pauls and Klein, 2012).

Topic language models are important for use in e.g., *unsupervised language model adaptation*: we want a language model that can adapt to the domain or topic of the current situation (e.g., a document in SMT or a conversation in ASR) automatically and select the appropriate words using both topic and syntactic context. Wallach (2006) is one such model, which generate each word based on local context and global topic information to capture the difference of lexical usage among different topics.

However, Wallach's experiments were limited to bigrams, a toy setting for language models, and experiments with higher-order n-grams have not yet been sufficiently studied, which we investigate in this paper. In particular, we point out the two fundamental problems caused when extending Wallach's model to a higher-order: *sparseness* caused by dividing all n-grams into exclusive topics, and *local minima* caused by the deep hierarchy of the model. On resolving these problems, we make several contributions to both computational linguistics and machine learning.

To address the first problem, we investigate incorporating a global language model for ease of sparseness, along with some priors on a suffix tree to capture the difference of *topicality* for each context, which include an *unsupervised* extension of the doubly hierarchical Pitman-Yor language model (Wood and Teh, 2009), a Bayesian generative model for *supervised* language model adaptation. For the second inference problem, we develop a novel blocked Gibbs sampler. When the number of topics is K and vocabulary size is V, n-gram topic model has $O(KV^n)$ parameters, which grow exponentially to n, making the local minima problem even more severe. Our sampler resolves this problem by moving many customers in the hierarchical Chinese restaurant process at a time.

We evaluate various models by incremental calculation of test document perplexity on 3 types of corpora having different size and diversity. By combining the proposed prior and the sampling method, our Bayesian model achieve much higher accuracies than the naive extension of Wallach (2006) and shows results competitive with the unigram rescaling (Gildea and Hofmann, 1999), which require

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1180–1190,
Seattle, Washington, USA, 18-21 October 2013.

huge computational cost at prediction, with much faster prediction time.

2 Basic Models

All models presented in this paper are based on the Bayesian n-gram language model, the hierarchical Pitman-Yor process language model (HPYLM). In the following, we first introduce the HPYLM, and then discuss the topic model extension of Wallach (2006) with HPYLM.

2.1 HPYLM

Let us first define some notations. W is a vocabulary set, $V = |W|$ is the size of that set, and $u, v, w \in W$ represent the word type.

The HPYLM is a Bayesian treatment of the n-gram language model. The generative story starts with the unigram word distribution G_ϕ, which is a V-dimensional multinomial where $G_\phi(w)$ represents the probability of word w. The model first generates this distribution from the PYP as $G_\phi \sim \mathrm{PYP}(a, b, G_0)$, where G_0 is a V-dimensional uniform distribution ($G_0(u) = \frac{1}{V}; \forall u \in W$) and acts as a prior for G_ϕ and a, b are hyperparameters called discount and concentration, respectively. It then generates all bigram distributions $\{G_u\}_{u \in W}$ as $G_u \sim \mathrm{PYP}(a, b, G_\phi)$. Given this distributions, it successively generates 3-gram distributions $G_{uv} \sim \mathrm{PYP}(a, b, G_u)$ for all $(u, v) \in W^2$ pairs, which encode a natural assumption that contexts having common suffix have similar word distributions. For example, two contexts "he is" and "she is", which share the suffix "is", are generated from the same (bigram) distribution G_{is}, so they would have similar word distributions. This process continues until the context length reaches $n-1$ where n is a prespecified n-gram order (if $n = 3$, the above example is a complete process). We often generalize this process using two contexts h and h' as

$$G_h \sim \mathrm{PYP}(a, b, G_{h'}), \tag{1}$$

where $h = ah'$, in which a is a leftmost word of h.

We are interested in the posterior word distribution following a context h. Our training corpus $\mathbf{w}$ is a collection of n-grams, from which we can calculate the posterior $p(w|h, \mathbf{w})$, which is often explained with the Chinese restaurant process (CRP):

$$p(w|h, \mathbf{w}) = \frac{c_{hw} - a t_{hw}}{c_{h\cdot} + b} + \frac{a t_{h\cdot} + b}{c_{h\cdot} + b} p(w|h', \mathbf{w}), \tag{2}$$

where c_{hw} is an observed count of n-gram hw called customers, while t_{hw} is a hidden variable called tables. $c_{h\cdot}$ and $t_{h\cdot}$ represents marginal counts: $c_{h\cdot} = \sum_w c_{hw}$ and $t_{h\cdot} = \sum_w t_{hw}$. This form is very similar to the well-known Kneser-Ney smoothing, and actually the Kneser-Ney can be understood as a heuristic approximation of the HPYLM. This characteristic enables us to build the state-of-the-art language model into a more complex generative model.

2.2 Wallach (2006) with HPYLM

Wallach (2006) is a generative model for a document collection that combines the topic model with a Bayesian n-gram language model. The latent Dirichlet allocation (LDA) (Blei et al., 2003) is the most basic topic model, which generates each word in a document based on a unigram word distribution defined by a topic allocated to that word. The bigram topic model of Wallach (2006) simply replaces this unigram word distribution (a multinomial) for each topic with a bigram word distribution [1]. In other words, ordinary LDA generates word conditioning *only* on the latent topic, whereas the bigram topic model generates conditioning on *both* the latent topic and the previous word, as in the bigram language model. Extending this model with a higher order n-gram is trivial; all we have to do is to replace the bigram language model for each topic with an n-gram language model.

The formal description of the generative story of this n-gram topic model is as follows. First, for each topic $k \in 1, \cdots, K$, where K is the number of topics, the model generates an n-gram language model G_h^k.[2] These n-gram models are generated by the PYP, so $G_h^k \sim \mathrm{PYP}(a, b, G_{h'}^k)$ holds. The model then generate a document collection. For each document $j \in 1, \cdots, D$, it generates a K-

[1] This is the model called *prior 2* in Wallach (2006); it consistently outperformed the other prior. Wallach used the Dirichlet language model as each topic, but we only explore the model with HPYLM because its superiority to the Dirichlet language model has been well studied (Teh, 2006b).

[2] We sometimes denote G_h^k to represent a language model of topic k, not a specific multinomial for some context h, depending on the context.

dimensional topic distribution θ_j by a Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_K)$ is a prior. Finally, for each word position $i \in 1, \cdots, N_j$ where N_j is the number of words in document j, i-th word's topic assignment z_{ji} is chosen according to θ_j, then a word type w_{ji} is generated from $G^{z_{ji}}_{h_{ji}}$ where h_{ji} is the last $n-1$ words preceding w_{ji}. We can summarize this process as follows:

1. Generate topics:
   For each $h \in \phi, \{W\}, \cdots, \{W\}^{n-1}$:
     For each $k \in 1, \cdots, K$:
       $G^k_h \sim \mathrm{PYP}(a, b, G^k_{h'})$

2. Generate corpora:
   For each document $j \in 1, \cdots D$:
     $\theta_j \sim \mathrm{Dir}(\boldsymbol{\alpha})$
     For each word position $i \in 1, \cdots, N_j$:
       $z_{ji} \sim \theta_j$
       $w_{ji} \sim G^{z_{ji}}_{h_{ji}}$

3 Extended Models

One serious drawback of the n-gram topic model presented in the previous section is *sparseness*. At inference, as in LDA, we assign each n-gram a topic, resulting in an exclusive clustering of n-grams in the corpora. Roughly speaking, when the number of topics is K and the number of all n-grams in the training corpus is N, a language model of topic k, G^k_h is learned using only about $O(N/K)$ instances of the n-grams assigned the topic k, making each G^k_h much sparser and unreliable distribution.

One way to alleviate this problem is to place another n-gram model, say G^0_h, which is shared with all topic-specific n-gram models $\{G^k_h\}^K_{k=1}$. However, what is the best way to use this special distribution? We explore two different approaches to incorporate this distribution in the model presented in the previous section. In one model, the HIERARCHICAL model, G^0_h is used as a prior for all other n-gram models, where G^0_h exploits global statistics across all topics $\{G^k_h\}$. In the other model, the SWITCHING model, no statistics are shared across G^0_h and $\{G^k_h\}$, but some words are directly generated from G^0_h regardless of the topic distribution.

3.1 HIERARCHICAL Model

Informally, what we want to do is to establish hierarchies among the global G^0_h and other topics $\{G^k_h\}$. In Bayesian formalism, we can explain this using an abstract distribution $\mathcal{F}$ as $G^k_h \sim \mathcal{F}(G^0_h)$. The problem here is making the appropriate choice for the distribution $\mathcal{F}$. Each topic word distribution already has hierarchies among $n-1$-gram and n-gram contexts as $G^k_h \sim \mathrm{PYP}(a, b, G^k_{h'})$. A natural solution to this problem is the doubly hierarchical Pitman-Yor process (DHPYP) proposed in Wood and Teh (2009). Using this distribution, the new generative process of G^k_h is

$$G^k_h \sim \mathrm{PYP}(a, b, \lambda G^k_{h'} + (1-\lambda) G^0_h), \qquad (3)$$

where λ is a new hyperparameter that determines mixture weight. The dependencies among G^0_h and $\{G^k_h\}$ are shown in Figure 1. Note that the generative process of G^0_h is the same as the HPYLM (1).

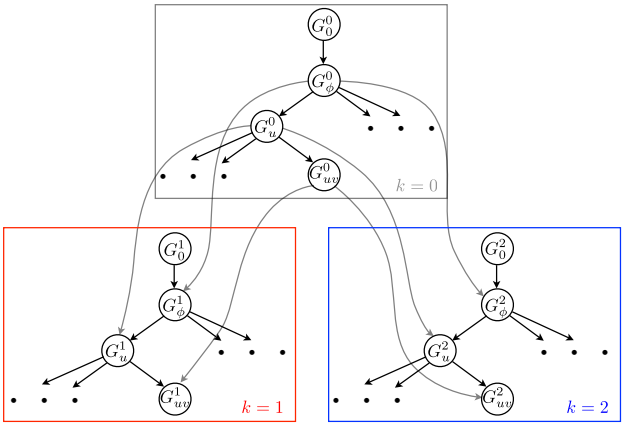


Figure 1: Variable dependencies of the HIERARCHICAL model. $\{u, v\}$ are word types, k is a topic and each G^k_h is a multinomial word distribution. For example, G^2_{uv} represents a word distribution following the context uv in topic 2.

Let us clarify the DHPYP usage differences between our model and the previous work of Wood and Teh (2009). A key difference is the problem setting: Wood and Teh (2009) is aimed at the *supervised* adaptation of a language model for a specific domain, whereas our goal is *unsupervised* adaptation. In Wood and Teh (2009), each G^k_h for $k \in 1, 2, \cdots$ corresponds to a language model of a specific domain and the training corpus for each k is prespecified and fixed. For ease of data sparseness of domain-specific corpora, latent model G^0_h exploits shared statistics among G^k_h for $k = 1, 2, \cdots$. In contrast, with our model, each G^k_h is a topic, so it must perform the clustering of n-grams in addition to ex-

ploiting the latent G_h^0. This makes inference harder and requires more careful design of λ.

Modeling of λ We can better understand the role of λ in (3) by considering the posterior predictive form corresponds to (2), which is written as

$$p(w|h,k,\mathbf{w}) = \frac{c_{hw}^k - at_{hw}^k}{c_{h\cdot}^k + b} + \frac{at_{h\cdot}^k + b}{c_{h\cdot}^k + b} q(w|h,k,\mathbf{w}), \quad (4)$$

$$q(w|h,k,\mathbf{w}) = \lambda p(w|h',k,\mathbf{w}) + (1-\lambda)p(w|h,0,\mathbf{w}),$$

where c, t with superscript k corresponds to the count existing in topic k. This shows us that λ determines the back-off behavior: which probability we should take into account: the shorter context of the same topic $G_{h'}^k$ or the full context of the global model G_h^0. Wood and Teh (2009) shares this variable across all contexts of the same length, for each k, but this assumption may not be the best. For example, after the context "in order", we can predict the word "to" or "that", and this tendency is unaffected by the topic. We call this property of context the *topicality* and say that "in order" has *weak topicality*. Therefore, we place λ as a distinct value for each context h, which we share across all topics. We designate this λ determined by h λ_h in the following. Moreover, similar contexts may have similar values of λ_h. For example, the two contexts "of the" and "in the", which share the suffix "the", both have a *strong topicality*[3]. We encode this assumption by placing hierarchical Beta distributions on the suffix tree across all topics:

$$\lambda_h \sim \mathrm{Beta}(\gamma\lambda_{h'}, \gamma(1-\lambda_{h'})) = \mathrm{DP}(\gamma, \lambda_{h'}), \quad (5)$$

where DP is the hierarchical Dirichlet process (Teh et al., 2006), which has only two atoms in {0,1} and γ is a concentration parameter. As in HPYLM, we place a uniform prior $\lambda_0 = 1/2$ on the base distribution of the top node ($\lambda_\phi \sim \mathrm{DP}(\gamma, \lambda_0)$).

Having generated the topic component of the model, the corpus generating process is the same as the previous model because we only change the generating process of G_h^k for $k = 1, \cdots, K$.

[3]These words can be used very differently depending on the context. For example, in a teen story, "in the room" or "in the school" seems more dominant than "in the corpora" or "in the topic", which is likely to appear in this paper.

3.2 SWITCHING Model

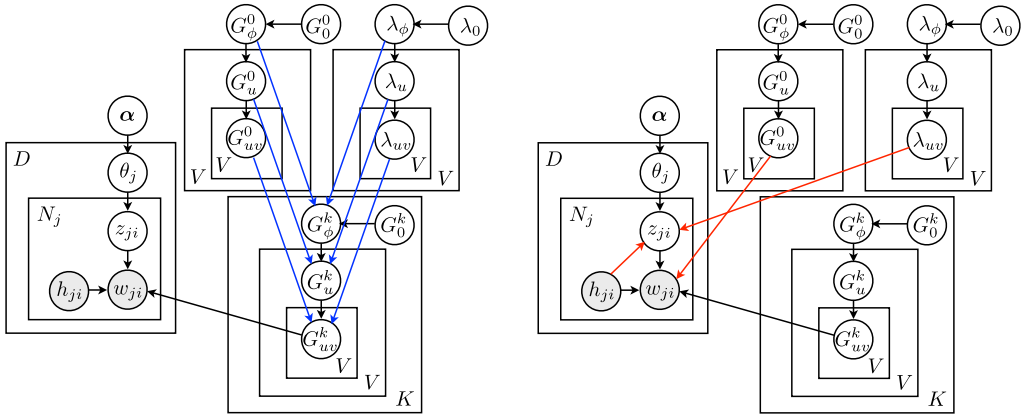
Our second extension also exploits the global G_h^0, albeit differently than the HIERARCHICAL model. In this model, the relationship of G_h^0 to the other $\{G_h^k\}$ is *flat*, not *hierarchical*: G_h^0 is a special topic that can generate a word. The model first generates each language model of $k = 0, 1, 2, \cdots, K$ independently as $G_h^k \sim \mathrm{PYP}(a, b, G_{h'}^k)$. When generating a word, it first determines whether to use global model G_h^0 or topic model $\{G_h^k\}_{k=1}^K$. Here, we use the λ_h introduced above in a similar way: the probability of selecting $k = 0$ for the next word is determined by the previous context. This assumption seems natural; we expect the G_h^0 to mainly generate common n-grams, and the topicality of each context determines how common that n-gram might be. The complete generative process of this model is written as follows:

1. Generate topics:
   For each $h \in \phi, \{V\}, \cdots, \{V\}^{n-1}$:
   $\lambda_h \sim \mathrm{DP}(\gamma, \lambda'_h)$
   For each $k \in 0, \cdots, K$:
   $G_h^k \sim \mathrm{PYP}(a, b, G_{h'}^k)$
2. Generate corpora:
   For each document $j \in 1, \cdots D$:
   $\theta_j \sim \mathrm{Dir}(\boldsymbol{\alpha})$
   For each word position $i \in 1, \cdots, N_j$:
   $l_{ji} \sim \mathrm{Bern}(\lambda_{h_{ji}})$
   If $l_{ji} = 0$: $z_{ji} = 0$
   If $l_{ji} = 1$: $z_{ji} \sim \theta_j$
   $w_{ji} \sim G_{h_{ji}}^{z_{ji}}$

The difference between the two models is their usage of the global model G_h^0. For a better understanding of this, we provide a comparison of their graphical models in Figure 2.

4 Inference

For posterior inference, we use the collapsed Gibbs sampler. In our models, all the latent variables are $\{G_h^k, \lambda_h, \theta_j, \mathbf{z}, \Theta\}$, where $\mathbf{z}$ is the set of topic assignments and $\Theta = \{a, b, \gamma, \boldsymbol{\alpha}\}$ are hyperparameters, which are treated later. We collapse all multinomials in the model, i.e., $\{G_h^k, \lambda_h, \theta_j\}$, in which G_h^k and λ_h are replaced with the Chinese restaurant process of PYP and DP respectively. Given the training corpus $\mathbf{w}$, the target posterior distribution is $p(\mathbf{z}, \mathbf{S}|\mathbf{w}, \Theta)$, where $\mathbf{S}$ is the set of seating arrangements of all restaurants. To distinguish the two types of restaurant, in the following, we refer the *restaurant* to indi-



(a) HIERARCHICAL (b) SWITCHING

Figure 2: Graphical model representations of our two models in the case of a 3-gram model. Edges that only exist in one model are colored.

cate the collapsed state of G_h^k (PYP), while we refer the *restaurant of* λ_h to indicates the collapsed state of λ_h (DP). We present two different types of sampler: a *token-based sampler* and a *table-based sampler*. For both samplers, we first explain in the case of our basic model (Section 2.2), and later discuss some notes on our extended models.

4.1 Token-based Sampler

The token-based sampler is almost identical to the collapsed sampler of the LDA (Griffiths and Steyvers, 2004). At each iteration, we consider the following conditional distribution of z_{ji} given all other topic assignments $\mathbf{z}^{-ji}$ and $\mathbf{S}^{-ji}$, which is the set of seating arrangements with a customer corresponds to w_{ji} removed, as

$$p(z_{ji}|\mathbf{z}^{-ji}, \mathbf{S}^{-ji}) \propto p(z_{ji}|\mathbf{z}^{-ji}) p(w_{ji}|z_{ji}, h_{ji}, \mathbf{S}^{-ji}), \tag{6}$$

where $p(w_{ji}|z_{ji}, h_{ji}, \mathbf{S}^{-ji}) =$

$$\frac{c_{hw}^k - at_{hw}^k}{c_{h\cdot}^k + b} + \frac{at_{h\cdot}^k + b}{c_{h\cdot}^k + b} p(w_{ji}|z_{ji}, h_{ji}, \mathbf{S}^{-ji}) \tag{7}$$

is a predictive word probability under the topic z_{ji}, and

$$p(z_{ji}|\mathbf{z}^{-ji}) = \frac{n_{jk}^{-ji} + \alpha_k}{N_j - 1 + \sum_{k'} \alpha_{k'}}, \tag{8}$$

where n_{jk}^{-ji} is the number of words that is assigned topic k in document j excluding w_{ji}, which is the same as the LDA. Given the sampled topic z_{ji}, we update the language model of topic z_{ji}, by adding customer w_{ji} to the restaurant specified by z_{ji} and context h_{ji}. See Teh (2006a) for details of these customer operations.

HIERARCHICAL Adding customer operation is slightly changed: When a new table is added to a restaurant, we must track the label $l \in \{0, 1\}$ indicating the parent restaurant of that table, and add the customer corresponding to l to the restaurant of λ_h. See Wood and Teh (2009) for details of this operation.

SWITCHING We replace $p(z_{ji}|\mathbf{z}^{-ji})$ with

$$p(z_{ji}|\mathbf{z}^{-ji}) = \begin{cases} p(l_{ji} = 0|h_{ji}) & (z_{ji} = 0) \\ p(l_{ji} = 1|h_{ji}) \cdot \frac{n_{jk}^{-ji} + \alpha_k}{\sum_{k \neq 0} n_{jk}^{-ji} + \sum_{k'} \alpha_{k'}} & (z_{ji} \neq 0), \end{cases} \tag{9}$$

where $p(l_{ji}|h_{ji})$ is a predictive of l_{ji} given by the CRP of $\lambda_{h_{ji}}$. We need not assign a label to a new table, but rather we always add a customer to the restaurant of λ_h according to whether the sampled topic is 0 or not.

4.2 Table-based Sampler

One problem with the token-based sampler is that the seating arrangement of the internal restaurant would never be changed unless a new table is created (or an old table is removed) in its child restaurant. This probability is very low, particularly in the restaurants of shallow depth (e.g., unigram or

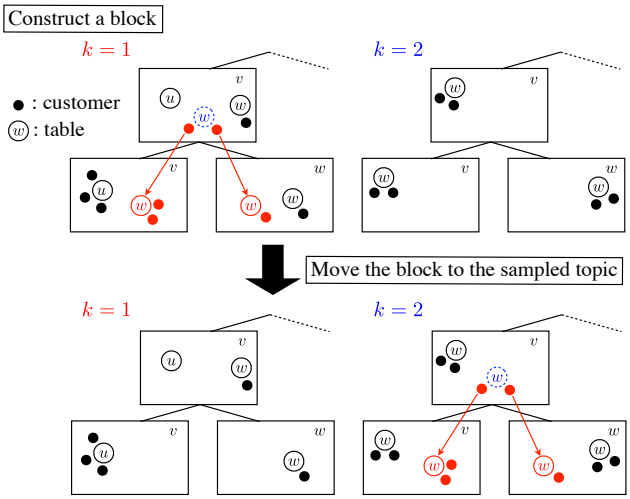


Figure 3: Transition of the state of restaurants in the table-based sampler when the number of topics is 2. $\{u, v, w\}$ are word types. Each box represents a restaurant where the type in the upper-right corner indicates the context. In this case, we can change the topic of the three 3-grams (vvw, vvw, uvw) in some documents from 1 to 2 at the same time.

bigram restaurants) because these restaurants have a larger number of customers and tables than those of deep depth, leading to get stack in undesirable local minima. For example, imagine a table in the restaurant of context "hidden" (depth is 2) and some topic, served "unit". This table is connected to tables in its child restaurants corresponding to some 3-grams (e.g., "of hidden unit" or "train hidden unit"), whereas similar n-grams, such as those of "of hidden units" or "train hidden units" might be gathered in another topic, but collecting these n-grams into the same topic might be difficult under the token-based sampler. The *table-based sampler* moves those different n-grams having common suffixes jointly into another topic.

Figure 3 shows a transition of state by the table-based sampler and Algorithm 4.2 depicts a high-level description of one iteration. First, we select a table in a restaurant, which is shown with a dotted line in the figure. Next, we descend the tree to collect the tables connected to the selected table, which are pointed by arrows. Because this connection cannot be preserved in common data structures for a restaurant described in Teh (2006a) or Blunsom et al. (2009), we select the child tables randomly. This is correct because customers in CRP are exchangeable, so we can restore the parent-child relations arbitrarily. We continue this process recursively until reaching the leaf nodes, obtaining a block of seating arrangement S. After calculating the conditional distribution, we sample new topic assignment for this block. Finally, we move this block to the sampled topic, which potentially changes the topic of many words across different documents, which are connected to customers in a block at leaf nodes (this connection is also arbitrary).

Algorithm 1 Table-based sampler

for all table **in** all restaurants **do**
  Remove a customer from the parent restaurant.
  Construct a block of seating arrangement S by descending the tree recursively.
  Sample topic assignment $z_S \sim p(z_S|S, \mathbf{S}^{-S}, \mathbf{z}^{-S})$.
  Move S to sampled topic, and add a customer to the parent restaurant of the first selected table.
end for

Conditional distribution Let $\mathbf{z}_S$ be the block of topic assignments connected to S and z_S be a variable indicating the topic assignment. Thanks to the exchangeability of all customers and tables in one restaurant (Teh, 2006a), we can imagine that customers and tables in S have been added to the restaurants last. We are interested in the following conditional distribution: (conditioning Θ is omitted)

$$p(z_S = k'|S, \mathbf{S}^{-S}, \mathbf{z}^{-S}) \propto p(S|\mathbf{S}^{-S}, k')p(z_S = k'|\mathbf{z}^{-S}),$$

where $p(S|\mathbf{S}^{-S}, k')$ is a product of customers' actions moving to another topic, which can be decomposed as:

$$p(S|\mathbf{S}^{-S}, k') = p(w|k', h) \prod_{s \in S} p(s|k') \quad (10)$$

$$p(s|k') = \frac{\prod_{i=0}^{t_s - 1}(b + a(t_{h_s w}^{k'(-s)} + i)) \prod_{j=1}^{c_{si}}(j - a)}{(b + c_{h_s w\cdot}^{k'(-s)})^{\overline{c_{s\cdot}}}} \quad (11)$$

$$\propto \frac{\prod_{i=0}^{t_s - 1}(b + a(t_{h_s w}^{k'(-s)} + i))}{(b + c_{h_s w\cdot}^{k'(-s)})^{\overline{c_{s\cdot}}}}. \quad (12)$$

Let us define some notations used above. Each $s \in S$ is a part of seating arrangements in a restaurant, there being t_s tables, i-th of which with c_{si} customers, with h_s as the corresponding context. A restaurant of context h and topic k has t_{hw}^k tables served dish w, i-th of which with c_{hwi}^k customers. Superscripts $-s$ indicate excluding the contribution

of customers in s, and $x^{\overline{n}} = x(x+1)\cdots(x+n-1)$ is the ascending factorial. In (10) $p(w|k', h)$ is the parent distribution of the first selected table, and the other $p(s|k')$ is the seating arrangement of customers. The likelihood for changing topic assignments across documents must also be considered, which is $p(z_S = k'|\mathbf{z}^{-S})$ and decomposed as:

$$p(z_S = k'|\mathbf{z}^{-S}) = \prod_j \frac{(n^{-S}_{jk'} + \alpha_{k'})^{\overline{n_j(S)}}}{(N^{-S}_j + \sum_k \alpha_k)^{\overline{n_j(S)}}}, \quad (13)$$

where $n_j(S)$ is the number of word tokens connected with S in document j.

HIERARCHICAL We skip tables on restaurants of $k = 0$, because these tables are all from other topics and we cannot construct a block. The effects of λ can be ignored because these are shared by all topics.

SWITCHING In the SWITCHING, $p(z_S = k'|\mathbf{z}^{-S})$ cannot be calculated in a closed form because $p(l_{ji}|h_{ji})$ in (9) would be changed dynamically when adding customers. This problem is the same one addressed by Blunsom and Cohn (2011), and we follow the same approximation in which, when we calculate the probability, we fractionally add tables and customers recursively.

4.3 Inference of Hyperparameters

We also place a prior on each hyperparameter and sample value from the posterior distribution for every iteration. As in Teh (2006a), we set different values of a and b for each depth of PYP, but share across all topics and sample values with an auxiliary variable method. We also set different value of γ for each depth, on which we place $\mathrm{Gamma}(1, 1)$. We make the topic prior $\boldsymbol{\alpha}$ asymmetric: $\boldsymbol{\alpha} = \beta\boldsymbol{\alpha}_0; \beta \sim \mathrm{Gamma}(1, 1), \boldsymbol{\alpha}_0 \sim \mathrm{Dir}(\mathbf{1})$.

5 Related Work

HMM-LDA (Griffiths et al., 2005) is a composite model of HMM and LDA that assumes the words in a document are generated by HMM, where only one state has a document-specific topic distribution. Our SWITCHING model can be understood as a lexical extension of HMM-LDA. It models the topicality by context-specific binary random variables, not by hidden states. Other n-gram topic models have focused mainly on information retrieval. Wang et al. (2007) is a topic model on automatically segmented chunks. Lindsey et al. (2012) extended this model with the hierarchical Pitman-Yor prior. They also used switching variables, but for a different purpose: to determine the segmenting points. They treat these variables completely independently, while our model employs a hierarchical prior to share statistical strength among similar contexts.

Our primary interest is language model adaptation, which has been studied mainly in the area of speech processing. Conventionally, this adaptation has relied on a heuristic combination of two separately trained models: an n-gram model $p(w|h)$ and a topic model $p(w|d)$. The unigram rescaling, which is a product model of these two models, perform better than more simpler models such as linear interpolation (Gildea and Hofmann, 1999). There are also some extensions to this method (Tam and Schultz, 2009; Huang and Renals, 2008), but these methods have one major drawback: at prediction, the rescaling-based method requires normalization across vocabulary at each word, which prohibits use on applications requiring dynamic (incremental) adaptation, e.g., settings where we have to update the topic distribution as new inputs come in. Tam and Schultz (2005) studied on this incremental settings, but they employ an interpolation. The practical interest here is whether our Bayesian models can rival the rescaling-based method in terms of prediction power. We evaluate this in the next section.

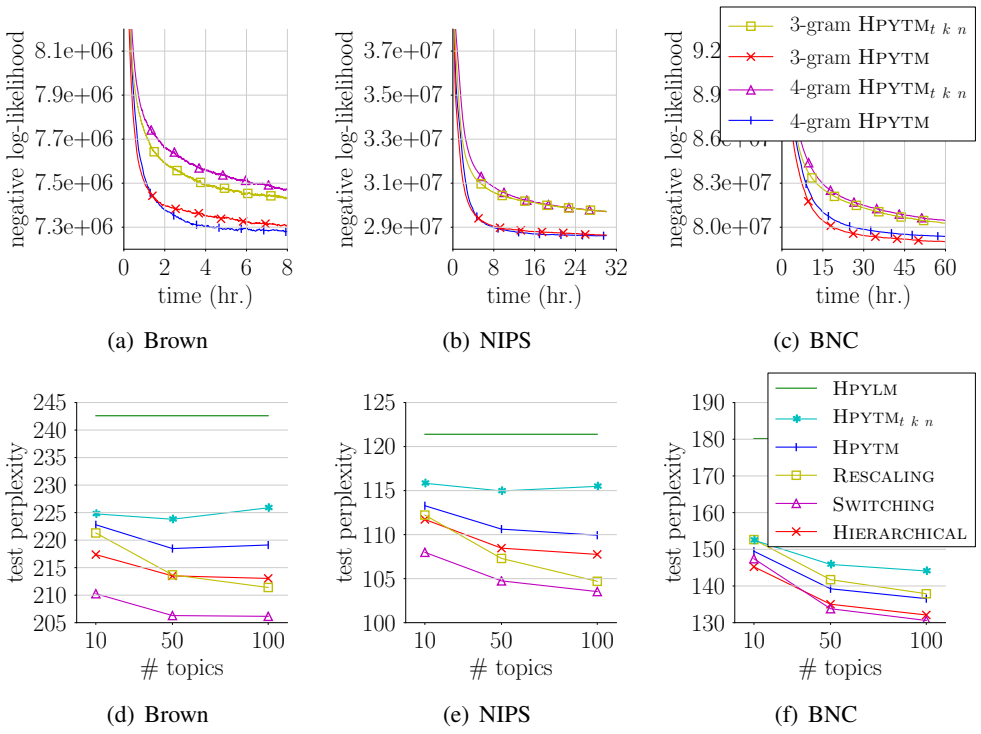
Corpus	min. appear	# types	training set # docs	training set # tokens	test set # docs	test set # tokens
Brown	4	19,759	470	1,157,225	30	70,795
NIPS	4	22,705	1500	5,088,786	50	167,730
BNC	10	33,071	6,162	12,783,130	100	202,994

Table 1: Corpus statistics after the pre-processing: We replace words appearing less than min.appear times in training + test documents, or appearing only in a test set with an unknown token. All numbers are replaced with #, while punctuations are remained.

6 Experiments

6.1 Settings

We test the effectiveness of presented models and the blocked sampling method on unsupervised language model adaptation settings. Specifically we



(a) Brown (b) NIPS (c) BNC

(d) Brown (e) NIPS (f) BNC

Figure 4: (a)–(c): Comparison of negative log-likelihoods at training of HPYTM ($K = 50$). Lower is better. HPYTM is trained on both token- and table-based samplers, while HPYTM$_{token}$ is trained only on the token-based sampler. (d)–(f): Test perplexity of various 3-gram models as a function of number of topics on each corpus.

concentrate on the *dynamic* adaptation: We update the posterior of language model given previously observed contexts, which might be decoded transcripts at that point in ASR or MT.

We use three corpora: the Brown, BNC and NIPS. The Brown and BNC are balanced corpora that consist of documents of several genres from news to romance. The Brown corpus comprises 15 categories. We selected two documents from each category for the test set, and use other 470 documents for the training set. For the NIPS, we randomly select 1,500 papers for training and 50 papers for testing. For BNC, we first randomly selected 400 documents from a written corpus and then split each document into smaller documents every 100 sentences, leading to 6,262 documents, from which we randomly selected 100 documents for testing, and other are used for training. See Table 1 for the pre-processing of unknown types and the resulting corpus statistics.

For comparison, besides our proposed HIERARCHICAL and SWITCHING models, we prepare various models for baseline. HPYLM is a n-gram language model without any topics. We call the model without the global G_h^0 introduced in Section 2.2 HPYTM. To see the effect of the table-based sampler, we also prepare HPYTM$_{token}$, which is trained only on the token-based sampler. RESCALING is the unigram rescaling. This is a product model of an n-gram model $p(w|h)$ and a topic model $p(w|d)$, where we learn each model separately and then combine them by:

$$p(w|h,d) \propto \left(\frac{p(w|d)}{p(w)}\right)^{\beta} p(w|h). \qquad (14)$$

We set β in (14) to 0.7, which we tuned with the Brown corpus.

6.2 Effects of Table-based Sampler

We first evaluate the effects of our blocked sampler at training. For simplicity, we concentrate on the HPYTM with $K = 50$. Table 4(a)–(c) shows negative likelihoods of the model during training. On all corpora, the model with the table-based sampler reached the higher probability space with much faster speed on both 3-gram and 4-gram models.

6.3 Perplexity Results

Training For burn-in, we ran the sampler as follows: For HPYLM, we ran 100 Gibbs iterations. For RESCALING, we ran 900 iterations on LDA and 100 iterations on HPYLM. For all other models, we ran 500 iterations of the Gibbs; HPYTM$_{token}$ is trained only on the token-based sampler, while for other models, the table-based sampler is performed after the token-based sampler.

Evaluation We have to adapt to the topic distribution of unseen documents incrementally. Although previous works have employed incremental EM (Gildea and Hofmann, 1999; Tam and Schultz, 2005) because their inference is EM/VB-based, we use the left-to-right method (Wallach et al., 2009), which is a kind of particle filter updating the posterior topic distribution of a test document. We set the number of particles to 10 and resampled each particle every 10 words for all experiments. To get the final perplexity, after burn-in, we sampled 10 samples every 10 iterations of Gibbs, calculated a test perplexity for each sample, and averaged the results.

Comparison of 3-grams Figure 4(d)–(f) shows perplexities when varying the number of topics. Generally, compared to the HPYTM$_{token}$, the HPYTM got much perplexity gains, which again confirm the effectiveness of our blocked sampler. Both our proposed models, the HIERARCHICAL and the SWITCHING, got better performances than the HPYTM, which does not place the global model G_h^0. Our SWITCHING model consistently performed the best. The HIERARCHICAL performed somewhat worse than the RESCALING when K become large, but the SWITCHING outperformed that.

Comparison of 4-grams and beyond We summarize the results with higher order n-grams in Table 2, where we also show the time for prediction. We fixed the number of topics $K = 100$ because we saw that all models but HPYTM$_{token}$ performed best at $K = 100$ when $n = 3$. Generally, the results are consistent with those of $n = 3$. The models with $n = \infty$ indicate a model extension using the Bayesian variable-order language model (Mochihashi and Sumita, 2008), which can naturally be integrated with our generative models. By this extension, we can prune unnecessary nodes stochastically during training. We can see that this ∞-gram did not hurt performances, but the sampled model get much more compact; in BNC, the number of nodes of the SWITCHING with 4-gram is about 7.9M, while the one with ∞-gram is about 3.9M. Note that our models require no explicit normalization, thereby drastically reducing the time for prediction compared to the RESCALING. This difference is especially remarkable when the vocabulary size becomes large.

		NIPS		BNC	
Model	n	PPL	time	PPL	time
HPYLM	4	117.2	59	169.2	74
HPYLM	∞	117.9	61	173.1	59
RESCALING	4	101.4	19009	130.3	36323
HPYTM	4	107.0	1004	133.1	980
HPYTM	∞	107.2	1346	133.6	1232
HIERARCHICAL	4	106.3	1038	129.0	993
HIERARCHICAL	∞	105.7	1337	129.3	1001
SWITCHING	4	**100.0**	1059	**125.5**	991
SWITCHING	∞	100.4	1369	125.7	1006

Table 2: Comparison of perplexity and the time require for prediction (in seconds). The number of topics is fixed to 100 on all topic-based models.

We can see that our SWITCHING performed consistently better than the HIERARCHICAL. One reason for this result might be the mismatch of prediction of the topic distribution in the HIERARCHICAL. The HIERARCHICAL must allocate some (not global) topics to every word in a document, so even the words to which the SWITCHING might allocate the global topic (mainly function words; see below) must be allocated to some other topics, causing a mismatch of allocations of topic.

6.4 Qualitative Results

To observe the behavior in which the SWITCHING allocates some words to the global topic, in Figure 5, we show the posterior of allocating the topic 0 or not at each word in a part of the NIPS training corpus. We can see that the model elegantly identified content and function words, learning the topic distribution appropriately using only semantic contexts. These same results in the HIERARCHICAL are presented in Table 3, where we show some relations between λ_h and context h. Contexts that might be likely to precede nouns have a higher value of λ_h,

there has been much recent work on measuring image statistics and on learning probability distributions on images . we observe that the mapping from images to statistics is many-to-one and show it can be quantified by a phase space factor .

Figure 5: The posterior for assigning topic 0 or not in NIPS by the ∞-gram SWITCHING. Darker words indicate a higher probability of not being assigned topic 0.

λ_h	h
0.0–0.1	in spite, were unable, a sort, on behalf, . regardless
0.5–0.6	assumed it, rand mines, plans was, other excersises
0.9–1.0	that the, the existing, the new, their own, and spatial

Table 3: Some contexts h for various values of λ_h induced by the 3-gram HIERARCHICAL in BNC.

0	46
according to	ˆ support vectors
ˆ (#)	in high dimentional
ˆ section #	as decision function
techniques such as	set of # observations
ˆ (b)	original data set
83	89
the hierarchical mixtures	ˆ linear discriminant
the rbf units	images per class
the gating networks	multi-class classification
grown hme	ˆ decision boundaries
the modular architecture	references per class

Table 4: Topical phrases from NIPS induced by the ∞-gram SWITCHING model. ˆ is a symbol for the beginning of a sentence and # represents a number.

while prefixes of idioms have a lower value. The ∞-gram extension gives us the posterior of n-gram order $p(n|h)$, which can be used to calculate the probability of a word ordering composing a phrase in topic k as $p(w, n|k, h) \propto p(n|h)p(w|k, n, h)$. In Table 4, we show some higher probability topic-specific phrases from the model trained on the NIPS.

7 Conclusion

We have presented modeling and algorithmic contributions to the existing Bayesian n-gram topic model. We explored two different priors to incorporate a global model, and found the effectiveness of the flat structured model. We developed a novel blocked Gibbs move for these types of models to accelerate inference. We believe that this Gibbs operation can be incorporated with other models having a similar hierarchical structure. Empirically, we demonstrate that by a careful model design and efficient inference, a well-defined Bayesian model can rival the conventional heuristics.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Phil Blunsom and Trevor Cohn. 2011. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA, June. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, Sharon Goldwater, and Mark Johnson. 2009. A note on the implementation of hierarchical dirichlet processes. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 337–340, Suntec, Singapore, August. Association for Computational Linguistics.

Daniel Gildea and Thomas Hofmann. 1999. Topic-based language models using em. In *In Proceedings of EUROSPEECH*, pages 2167–2170.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press.

Songfang Huang and Steve Renals. 2008. Unsupervised language model adaptation based on topic and role information in multiparty meetings. In *in Proc. Interspeech08*, pages 833–836.

F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. 1991. A dynamic language model for speech recognition. In *Proceedings of the workshop on Speech and Natural Language*, HLT '91, pages 293–295, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Lindsey, William Headden, and Michael Stipicevic. 2012. A phrase-discovering topic model using hierarchical pitman-yor processes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 214–222, Jeju Island, Korea, July. Association for Computational Linguistics.

Daichi Mochihashi and Eiichiro Sumita. 2008. The infinite markov model. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1017–1024. MIT Press, Cambridge, MA.

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 959–968. Association for Computational Linguistics.

Yik-Cheung Tam and Tanja Schultz. 2005. Dynamic language model adaptation using variational bayes inference. In *INTERSPEECH*, pages 5–8.

Yik-Cheung Tam and Tanja Schultz. 2009. Correlated bigram lsa for unsupervised language model adaptation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1633–1640.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Yee Whye Teh. 2006a. *A Bayesian Interpretation of Interpolated Kneser-Ney*. NUS School of Computing Technical Report TRA2/06.

Yee Whye Teh. 2006b. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, Sydney, Australia, July. Association for Computational Linguistics.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1105–1112, New York, NY, USA. ACM.

Hanna M. Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 977–984.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 697–702, Washington, DC, USA. IEEE Computer Society.

Frank Wood and Yee Whye Teh. 2009. A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 12.

An Empirical Study Of Semi-Supervised Chinese Word Segmentation Using Co-Training

Fan Yang
Nuance Communications, Inc.
fan.yang@nuance.com

Paul Vozila
Nuance Communications, Inc.
paul.vozila@nuance.com

Abstract

In this paper we report an empirical study on semi-supervised Chinese word segmentation using co-training. We utilize two segmenters: 1) a word-based segmenter leveraging a word-level language model, and 2) a character-based segmenter using character-level features within a CRF-based sequence labeler. These two segmenters are initially trained with a small amount of segmented data, and then iteratively improve each other using the large amount of unlabelled data. Our experimental results show that co-training captures 20% and 31% of the performance improvement achieved by supervised training with an order of magnitude more data for the SIGHAN Bakeoff 2005 PKU and CU corpora respectively.

1 Introduction

In the literature there exist two general models for supervised Chinese word segmentation, the word-based approach and the character-based approach. The word-based approach searches for all possible segmentations, usually created using a dictionary, for the optimal one that maximizes a certain utility. The character-based approach treats segmentation as a character sequence labeling problem, indicating whether a character is located at the boundary of a word. Typically the word-based approach uses word level features, such as word n-grams and word length; while the character-based approach uses character level information, such as character n-grams. Both approaches have their own advantages and disadvantages, and there has been some research in combining the two approaches to improve the performance of supervised word segmentation.

In this research we are trying to take advantage of the word-based and the character-based approaches in the semi-supervised setting for Chinese word segmentation, where there is only a limited amount of human-segmented data available, but there exists a relatively large amount of in-domain unsegmented data. The goal is to make use of the in-domain unsegmented data to improve the ultimate performance of word segmentation. According to Sun et al. (2009), "the two approaches [word-based and character-based approaches] are either based on a particular view of segmentation." This naturally motivates the use of co-training, which utilizes two models trained on different views of the input labeled data which then iteratively *educate* each other with the unlabelled data. At the end of the co-training iterations, the initially weak models achieve improved performance. Co-training has been successfully applied in many natural language processing tasks. In this paper we describe an empirical study of applying co-training to semi-supervised Chinese word segmentation. Our experimental results show that co-training captures 20% and 31% of the performance improvement achieved by supervised training with an order of magnitude more data for the SIGHAN Bakeoff 2005 PKU and CU corpora respectively.

In section 2 we review the two supervised approaches and co-training algorithm in more detail. In section 3 we describe our implementation of the co-training word segmentation. In section 4 we de-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1191–1200,
Seattle, Washington, USA, 18-21 October 2013.

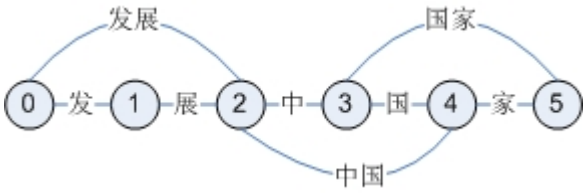


Figure 1: A search space for word segmenter

scribe our co-training experiments. In section 5 we conclude the paper.

2 Related Work

In this section, we first review the related research on the word-based and the character-based approaches for Chinese word segmentation, and comparatively analyze these two supervised approaches. We then review the related research on co-training.

2.1 Supervised Word Segmentation

2.1.1 Word-Based Segmenter

Given a character sequence $c_1c_2...c_n$, the word-based approach searches in all possible segmentations for one that maximizes a pre-defined utility function, formally represented as in Equation 1. The search space, $GEN(c_1c_2...c_n)$, can be represented as a lattice, where each vertex represents a character boundary index and each arc represents a word candidate which is the sequence of characters within the index range. A dictionary[1] can be used to generate such a lattice. For example, given the character sequence "发展中国家" and a dictionary that contains the words {发展，中国，国家} and all single Chinese characters, the search space is illustrated in Figure 1.

$$\hat{W} = arg \max_{W \in GEN(c_1c_2...c_n)} Util(W) \quad (1)$$

Dynamic programming such as Viterbi decoding is usually used to search for the optimized segmentation. The utility can be as simple as the negation of number of words (i.e. $Util(W) = -\mid W \mid$), which gives a reasonable performance if the dictionary used for generating the search space has a good coverage. Alternatively one can search for the segmentation that maximizes the word sequence probability $P(W)$ (i.e. $Util(W) = P(W)$). With a Markov assumption, $P(W)$ can be calculated using a language model as in Equation 2.

$$\begin{aligned} P(W) &= P(w_1w_2...w_m) \\ &= P(w_1).P(w_2|w_1)...P(w_n|w_1w_2...w_{n_1}) \\ &= P(w_1)P(w_2|w_1)...P(w_n|w_{n-1}) \end{aligned} \quad (2)$$

More generally, the utility can be formulated as a semi-Markov linear model, defined as Equation 3, in which Φ is the feature function vector, and Θ is the parameter vector that can be learned from training data using different techniques: Liang (2005), Gao et al. (2005), and Zhang and Clark (2007) use averaged perceptron; Nakagawa (2004) uses general iterative scaling; Andrew (2006) uses semi-Markov CRF; and Sun (2010) uses a passive-aggressive learning algorithm.

$$Util(W) = \Theta^T \Phi(c_1c_2...c_n, W) \quad (3)$$

2.1.2 Character-Based Segmenter

The character-based approach treats word segmentation as a character sequence labeling problem, to label each character with its location in a word, first proposed by Xue (2003).[2] The basic labeling scheme is to use two tags: 'B' for the beginning character of a word and 'O' for other characters (Peng et al., 2004). Xue (2003) use a four-tag scheme based on some linguistic intuitions: 'B' for the beginning character, 'I' for the internal characters, 'E' for the ending character, and 'S' for single-character word. For example, the word sequence "洽谈会 很 成功" can be labelled as 洽\B 谈\I 会\E 很\S 成\B 功\E. Zhao et al. (2010) further extend this scheme by using six tags.

Training and decoding of the character labeling problem is similar to part-of-speech tagging, which

[1] A dictionary is not a must to create the search space but it could shrink the search space and also lead to improved segmentation performance.

[2] Teahan et al. (2000) use a character language model to determine whether a word boundary should be inserted after each character, which can also be considered as a character-based approach as well.

is also generally formulated as a linear model. Many machine learning techniques have been explored: Xue (2003) use a maximum entropy model; Peng et al. (2004) use linear-chain CRF; Liang (2005) uses averaged perceptron; Sun et al. (2009) use a discriminative latent variable approach.

2.1.3 Comparison and Combination

It is more natural to use word-level information, such as word n-grams and word length, in a word-based segmenter; while it is more natural to use character-level information, such as character n-grams, in a character-based segmenter. Sun (2010) gives a detailed comparison of the two approaches from both the theoretical and empirical perspectives. Word-level information has greater representational power in terms of contextual dependency, while character-level information is better at morphological analysis in terms of word internal structures.

On one hand, features in a character-based model are usually defined in the neighboring n-character window; and an order-K CRF can only look at the labels of the previous K characters. Given that many words contain more than one character, a word-based model can examine a wider context. Thus the contextual dependency information encoded in a character-based model is generally weaker than in a word-based model. Andrew (2006) also shows that semi-Markov CRF makes strictly weaker independence assumptions than linear CRF and so a word-based segmenter using an order-K semi-Markov model is more expressive than a character-based model using an order-K CRF.

On the other hand, Chinese words have internal structures. Chinese characters can serve some morphological functions in a word. For example, the character 们 usually works as a suffix to signal plural; the character 者 can also be a suffix meaning a group of people; and 阿 generally works as a prefix before a person's nickname that has one character. Such morphological information is extremely useful for identifying unknown words. For example, a character-based model can learn that 阿 is usually tagged as 'B' and the next character is usually tagged as 'E'. Thus even when 阿甘 is not an existing word in the training data, a character-based model might still be able to correctly label it as 阿\B 甘\E.

Recent advanced Chinese word segmenters, either word-based or character-based, have been trying to make use of both word-level and character-level information. For example, Nakagawa (2004) integrates the search space of a character-based model into a word-based model; Andrew (2006) converts CRF-type features into semi-CRF features in his semi-Markov CRF segmenter; Sun et al. (2009) add word identify information into their character-based model; and Sun (2010) combine the two approaches at the system level using bootstrap aggregating.

2.2 Co-Training

The co-training approach was first introduced by Blum and Mitchell (1998). Theoretical analysis of its effectiveness is given in (Blum and Mitchell, 1998; Dasgupta et al., 2001; Abney, 2002). Co-training works by partitioning the feature set into two conditionally independent views (given the true output). On each view a statistical model can be trained. The presence of multiple distinct views of the data can be used to train separate models, and then each model's predictions on the unlabeled data are used to augment the training set of the other model.

Figure 2 depicts a general co-training framework. The inputs are two sets of data, a labelled set S and an unlabelled set U. Generally S is small and U is large. Two statistical models M1 and M2 are used, which are built on two sets of data L1 and L2 initialized as S but then incrementally increased in each iteration. C is a cache holding a small subset of U to be labelled by both models (Blum and Mitchell, 1998; Abney, 2002). In some applications, C is not used and both models label the whole set of U (i.e. C==U) (Collins and Singer, 1999; Nigam and Ghani, 2000; Pierce and Cardie, 2001). The stopping criteria can be, for example, when U is empty, or when a certain number of iterations are executed.

In step 5 and 6 during each iteration, some data labelled by M1 are selected and added to the training set L2, and vice versa. Several selection algorithms have been proposed. Dasgupta et al. (2001) and Abney (2002) use a selection algorithm that tries to maximize the agreement rate between the two models. The more popular selection algorithm is to choose the K examples that have the highest con-

```
Input:
    S is the labelled data
    U is the unlabelled data
Variables:
    L1 is the training data for View One
    L2 is the training data for View Two
    C is a cache holding a small subset of U
Initialization:
    L1 <- S
    L2 <- S
    C <- randomly sample a subset of U
    U <- U - C
REPEAT:
  1. Train M1 using L1
  2. Train M2 using L2
  3. Use M1 to label C
  4. Use M2 to label C
  5. Select examples labelled by M1, add to L2
  6. Select examples labelled by M2, add to L1
  7. Randomly move samples from U to C
     so that C maintains its size
UNTIL stopping criteria
```

Figure 2: A generic co-training framework

fidence score (Nigam and Ghani, 2000; Pierce and Cardie, 2001). In order to balance the class distributions in the training data L1 and L2, Blum and Mitchell (1998) select P positive examples and Q negative examples that have the highest confidence scores respectively. Wang et al. (2007) and Guz et al. (2007) use disagreement-based selection, which adds to L2, data that is labeled by M1 and M2 with high and low confidence respectively, with the intuition that such data are more useful and compensatory to M2. Finally, instead of adding the selected data to the training data, Tur (2009) propose the co-adaptation approach which linearly interpolates the existing model with the new model built with the new selected data.

3 Segmentation With Co-Training

3.1 Design of Two Segmenters

The use of co-training needs two statistical models that satisfy the following three conditions. First, in theory these two models need to be built on two conditionally independent views. However this is a very strong assumption and many large-scale NLP problems do not have a natural split of features to satisfy this assumption. In practice it has been shown that co-training can still achieve improved performance when this assumption is violated, but conforming to the conditionally independent assumption leads to a bigger gain (Nigam and Ghani, 2000; Pierce and Cardie, 2001). Thus we should strive to have the two models less correlated. Second, the two models both need to be effective for the task, that is, each of the models itself can perform the task reasonably well. Third, the decoding and training of the two models need to be efficient, as in co-training we need to segment the unlabelled data and re-train the models in each iteration. In the following we describe our design of the two segmenters.

Word-based segmenter In the word-based segmenter, we utilize a statistical n-gram language model and try to optimize the language modeling score together with a word insertion penalty, as show in Equation 4. K is a per-word penalty that is pre-determined with 10 fold cross-validation on the SIGhan PKU training set. We train a Kneser-Ney backoff language model from the training data, and extract a dictionary of words from the training data for generating the search space. Our pilot study suggested that a bigram language model is sufficient for this task.

$$Util(W) = \ln(P(W)) - |W| * K \quad (4)$$

Character-based segmenter We use an order-1 linear conditional random field to label a character sequence. Following Xue (2003), we use the four-tag scheme "BIES". We use the tool CRF++[3]. The features that we use are character n-grams within the neighboring 5-character window and tag bigrams. Given a character c_0 in the character sequence $c_{-2}c_{-1}c_0c_1c_2$, we extract the following features: character unigrams c_{-2}, c_{-1}, c_0, c_1, c_2, bigrams $c_{-1}c_0$ and c_0c_1. L2 regularization is applied in learning.

As can be seen, we build a word-based segmenter that uses only word level features, and a character-based segmenter that uses only character level features. These two segmenters by no means satisfy the conditionally independence assumption, but we have the hope that they are not too correlated as they use different levels of information and these

[3] http://crfpp.googlecode.com/svn/trunk/doc/index.html

different levels of information have been shown to be complementary in literature. Also the effectiveness of these two segmenters has been demonstrated in literature and will be shown again in our results in Section 4. Finally, both segmenters can decode and be trained pretty quickly. In our implementation, running on a Xeon 2.93GHz CPU with 4G of memory, it takes less than 30 seconds to build a word-based segmenter and less than 1 hour to build a character-based segmenter with the SIGhan PKU training data, and it takes less than 20 seconds to apply the word-based segmenter or less than 5 seconds to apply the character-based segmenter to the PKU testing data.

3.2 Co-Training

We follow the framework in Figure 2 for the co-training setup. We do not use the cache C, but directly label the whole unlabelled data set U, because in our experiment setup (see Section 4) U is not huge and computationally we can afford to label the whole set. The stopping criteria we use is when U is empty. Following Wang et al. (2007) and Guz et al. (2007), we use disagreement-based data selection. In every iteration, we pick some sentences that are segmented by the character-based model with high confidence but are segmented by the word-based model with low confidence to add to the training data of the word-based model, and vice versa. Confidence score is normalized with regard to the length of the sentence (i.e. number of characters) to avoid biasing towards short sentences. Confidence scores between the two segmenters, however, are not directly comparable. Thus we rank the sentences by their confidence scores in each segmenter respectively, and calculate the rank difference between the two segmenters. This rank difference is used as the indication of the gap of the confidence between the two segmenters. The sentences of highest rank difference are assigned to the training data of the word-based segmenter, with the segmentations from the character-based model; and the sentences of lowest rank difference are assigned to the training data of the character-based model, with segmentations from the word-based model.

4 Experiments

4.1 Data and Experiment Setup

We conduct a set of experiments to evaluate the performance of our co-training on semi-supervised Chinese word segmentation. Two corpora, the PKU corpus and the CU corpus, from the SIGhan Bakeoff 2005 are used. The PKU corpus contains texts of simplified Chinese characters, which include 19056 sentences in the training data and 1945 sentences in the testing data. The CU corpus contains texts of traditional Chinese characters, which include 53019 sentences in the training data and 1493 sentences in the testing data. The training data in each corpus is randomly split into 10 subsets. In each run one set is used as the labelled data S, and the other nine sets are combined and used as the unlabelled data U with segmentations removed. That is, 10% of the training data is used as segmented data, and 90% are used as unsegmented data in our semi-supervised training. This setup resembles our semi-supervised application, where there is only a small limited amount of segmented data but a relatively large amount of in-domain unsegmented data available. The final trained character-based and word-based segmenters from co-training are then evaluated on the testing data. Results we report in this paper are the average of the 10 runs. F-measure is used as the performance measurement. A 99% confidence interval is calculated as $\pm 2.56\sqrt{p(1-F)/N}$ for statistical significance evaluation, where F is the F-measure and N is the number of words. Subsequent assertions in this paper about statistical significance indicate whether or not the p-value in question exceeds 1%.

4.2 Co-Training Results

For comparison, we measure the baseline as the performance of a model trained with the 10% segmented data only (referred to as *BASIC* baselines). The *BASIC* baselines, both for the word-based model and the character-based model, however, use only the segmented data but leave out the large amount of available unsegmented data. We thus measure another baseline (referred to as *FOLD-IN*), which naively uses the unsegmented data. In the *FOLD-IN* baseline, a model is first trained with the 10% segmented data, and then this model is used

Table 1: Co-training results

	PKU		CU	
	char	word	char	word
BASIC	90.4	84.2	89.2	78.4
FOLD-IN	90.5	84.2	89.3	78.5
CEILING	94.5	93.0	94.2	88.9
CO-TRAINING	91.2	90.3	90.2	86.2

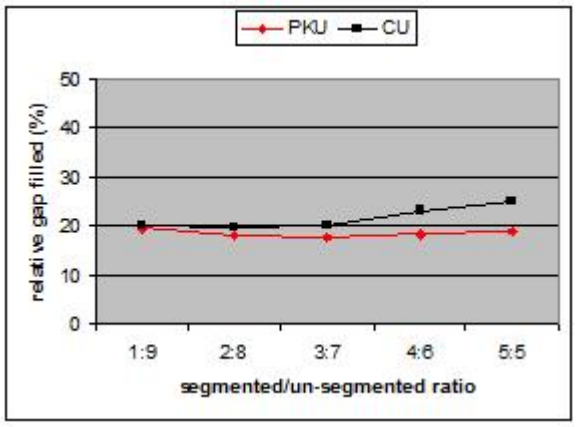


Figure 3: Gap filling with different split ratio

to label the unsegmented data. The automatic segmentation is then combined with the segmented data to build a new model. We also measure the *CEILING* as the performance of a model trained with all the training data available, i.e. we use the true segmentations of the 90% unsegmented data together with the 10% segmented data to train a model. The *CEILING* tells us the oracle performance when we have all segmented data for training, while the *BASIC* shows how much performance is dropped when we only have 10% of the segmented data. The performance of co-training will tell us how much we can fill the gap by taking advantage of the other 90% as unsegmented data in the semi-supervised training. The *FOLD-IN* baseline further verifies the effectiveness of co-training, i.e. co-training should perform better than naively folding in the unsegmented data.

Table 1 presents the results. First, we see that both the word-based and character-based models are doing a decent job under the *CEILING* condition. This confirms the effectiveness of each individual model, which is generally a requirement for running co-training. The character-based segmenter, although simple and with character-level features only, achieves the performance that is close to the state-of-the-art technologies that are much more complicated (The best performance is 95.2% for the PKU corpus and 95.1% for the CU corpus, see (Sun et al., 2009)). Second, we see that under all four conditions, the character-based segmenter performs better than the word-based model. This is not too surprising as these results are consistent with those reported in the literature. The word-based segmenter implemented in this work is less powerful, and it needs a good dictionary to achieve good performance. In our implementation, a dictionary is extracted from the segmented training set. Thus the word-based model suffers a lot when the training data is small. Third, we see that both the word-based model and the character-based model are improved by co-training, and the improvements are all statistically significant. It is not surprising for the word-based model to learn from the more accurate character-based model, which can also identify new words to add to the dictionary. More interestingly, the character-based segmenter is able to benefit from the less powerful word-based segmenter. For the character-based model, about 20% of the gap between *BASIC* and *CEILING* is filled by co-training, consistently in both the PKU and CU corpora. Finally, comparing *FOLD-IN* and *BASIC*, we see that naively using the unsegmented data does not lead to a significant improvement. This suggests that co-training provides a process that effectively makes use of the unsegmented data.

For completeness, in Figure 3 we also show the relative gap filling with different splits of the segmented vs unsegmented data. With more data moving to the segmented set, the absolute improvement of co-training over *BASIC* gets smaller, while the gap between the *BASIC* and *CEILING* also becomes smaller. The relative gap filled, i.e. the improvement relative to the difference between *BASIC* and *CEILING*, as can be seen, consistently falls inside the section of 15% and 25%.

4.3 Further Analysis

It is not surprising that the word-based segmenter benefits from co-training since it learns from the more accurate character-based segmenter. Our focus, however, is to better understand what benefit the character-based segmenter gains from the co-training procedure. The character-based segmenter treats word segmentation as a character sequence labelling problem with four tags "B I E S". Assuming that segmentation accuracy is proportional to tag accuracy, we examine the tag accuracy of the character-based segmenter before and after co-training.

If a character is labelled with tag *T0* initially before co-training and with tag *T1* after co-training, with the tag *T1* different from *T0*, there can be one of three cases: 1) *T0* is correct; 2) *T1* is correct; or 3) neither is correct. The absolute gain from co-training of switching from tag *T0* to *T1* is defined as the number of case 2 instances less case 1 instances. Absolute gain indicates the gain of tag accuracy where co-training learns to switch from *T0* to *T1*, and it contributes to the overall tag accuracy improvement. We also define *relative gain* of switching from tag *T0* to *T1* as the absolute gain divided by the total number of cases switching from tag *T0* to *T1*. Relative gain indicates how well co-training learns to switch from *T0* to *T1*.

Results are shown in Table 2. For both absolute gain and relative gain, 12 ordered switching pairs can be divided into two pools, a positive pool that has higher gain including $B \rightarrow E$, $E \rightarrow B$, $S \rightarrow B$, $S \rightarrow E$, $E \rightarrow I$, $B \rightarrow I$, $B \rightarrow S$, and a neutral pool that has lower or even negative gain including $I \rightarrow E$, $I \rightarrow S$, $I \rightarrow B$, $E \rightarrow S$, $S \rightarrow I$. The $S \rightarrow B$, $S \rightarrow E$, $B \rightarrow I$, $E \rightarrow I$ in the positive pool actually suggest that the character-based segmenter learns from co-training to combine a single-character word with it's neighbour to create a new longer word; whereas the $I \rightarrow E$, $I \rightarrow S$, $I \rightarrow B$ in the neutral pool suggest that it does not really learn how to separate a longer words into smaller units.

Table 2: Absolute Gain and Relative Gain

		Absolute Gain		Relative Gain	
T0	T1	PKU	CU	PKU	CU
B	I	678	681	0.28	0.59
B	E	2331	1727	0.41	0.46
B	S	1025	686	0.07	0.08
I	B	458	-283	0.07	-0.08
I	E	61	-1117	0.01	-0.23
I	S	323	-338	0.09	-0.34
E	B	2163	1601	0.41	0.46
E	I	963	819	0.36	0.62
E	S	520	-13	0.03	0.00
S	B	1847	892	0.27	0.30
S	I	104	47	0.22	0.28
S	E	1438	846	0.26	0.55

4.4 Feature Combination

We split the features into two sets, a character-level feature set used by the character-based segmenter and a word-level feature set used by the word-based segmenter. We have shown that these two segmenters improve each other via co-training. However, as reviewed in Section 2.1, there is active research in combining the character-level and word-level features in a segmenter. When training with the whole set of data (i.e. under the *CEILING* condition), a segmenter with combined features tends to perform better than only using one set of features. Thus we need to address two problems. First, we want to understand whether co-training, which splits the features, can actually beat the *BASIC* and *FOLD-IN* baselines of a segmenter with combined features. Second, we want to explore whether we can further improve the final co-training performance by feature combination.

To address these two problems, we adopt Weiwei Sun's character-based segmenter[4] in (Sun, 2010). We use this segmenter because it is publicly available and it performs well on both the PKU corpus and CU corpus. It models word segmentation as a character labelling problem, and solves it with a *passive-aggressive* optimization algorithm. It uses the same feature set as in (Sun et al., 2009), including both character-level features and word-level features. Character-level features include character uni-grams and bi-grams in the five character window, and whether the current character is the same as the next or the one after the next character. Word-

[4] Available at http://www.coli.uni-saarland.de/ wsun/ccws.tgz

Table 3: Sun-Segmenter's performance

	PKU	CU
BASIC	90.3	89.2
FOLD-IN	90.6	89.7
CEILING	94.8	95.0

Table 4: Results of feature combination

	PKU	CU
data combination	91.2	90.9
relabelling	91.2	91.0

level features include what word uni-grams or bi-grams are anchored at the current character. Word uni-grams and bi-grams are extracted from the labeled training data. For more details, please refer to (Sun et al., 2009) and (Sun, 2010). For ease of description, we will refer to Weiwei Sun's segmenter with combined features as *Sun-Segmenter*, and the character-based segmenter used in our co-training which uses character-level features as *Char-Segmenter*.

Table 3 shows the performance of the Sun-Segmenter under the three conditions: *BASIC*, *FOLD-IN*, and *CEILING*. We see that under the *CEILING* condition, the Sun-Segmenter outperforms the Char-Segmenter by 0.3% in the PKU corpus and 0.8% in the CU corpus. However, under the *BASIC* condition when there is only 10% of training data available, the Sun-Segmenter gives no gain. This probably is due to the fact that the Sun-Segmenter uses a much larger feature set and thus correspondingly a larger training set is needed to avoid under-fitting. The Sun-Segmenter has more gain when folding in the unsegmented data than the Char-Segmenter, further suggesting that the Sun-Segmenter is benefiting from the size of data. For both corpora, however, the Char-Segmenter after co-training beats the *FOLD-IN* baseline of the Sun-Segmenter by at least 0.5%, and the improvement is statistically significant. When there is only a small amount of segmented data available, using a more advanced segmenter with combined features still under-performs compared to co-training. These results justify the split of features for running co-training.

Next we would like to explore whether we could further improve the co-training performance, given that we have a more advanced segmenter using combined features. We try two approaches. In the first approach, after all the iterations of co-training, the data are split into two sets, one set for training the word-based segmenter L1 and the other set for training the character-based segmenter L2. The segmentations of these two sets of data are probably better than the segmentations under the *FOLD-IN* condition. We thus combine the two sets of data, and use the combined data to train a new model with the Sun-Segmenter. In the second approach, we use the character-based segmenter after co-training, which has an improved performance, to relabel the set of unsegmented data U, and then combine it with the segmented data set S. We then use the combined data to train a new model with the Sun-Segmenter.

Results are shown in Table 4. In the PKU corpus, we do not see a gain using either the data combination approach or the relabelling approach compared to the performance of the Char-Segmenter after co-training, probably because the Sun-Segmenter just modestly improves over the Char-Segmenter under the *CEILING* condition. However, in the CU corpus, where under the *CEILING* condition the Sun-Segmenter has a much bigger gain over the Char-Segmenter, there is 0.7% improvement by using the data combination approach and 0.8% by using the relabelling approach, and the improvement is statistically significant. Overall, using co-training with feature combination we are able to cut the gap between the *BASIC* baseline and *CEILING* of the Sun-Segmenter by 20% in the PKU corpus and 31% in the CU corpus.

5 Discussion

There has been some research on semi-supervised Chinese word segmentation. For example, Liang (2005) derive word cluster features and mutual information features from unlabelled data, and add them to supervised discriminative training; Li and Sun (2009) use punctuation as implicit annotations of a character starting a word (the character after a punctation) or ending a word (the character before a punctuation) in a large unlabelled data set to augment supervised data; Sun and Xu (2011) derive a large set of features from unlabelled data, includ-

ing mutual information, accessor variety and punctuation variety to augment the character and word features derived from labelled data. These research works aim to use huge amount of unsegmented data to further improve the performance of an already well-trained supervised model.

In this paper, we assume a much limited amount of segmented data available, and try to boost up the performance by using in-domain unsegmented data. Chinese word segmentation is domain-sensitive or application sensitive. For example, a CRF segmenter trained on the SIGhan MSR training data, which achieves an F-measure of 96.5% in the MSR testing data, only has 83.8% when applied to the PKU testing data; and the same CRF segmenter trained on the PKU training data achieves 94.5% on the PKU testing data. When one starts a new application that requires word segmentation in a new domain, it is likely that there is only a very small amount of segmented data available.

We propose the approach of co-training for Chinese word segmentation for the semi-supervised setting where there is only a limited amount of human-segmented data available, but there exists a relatively large amount of in-domain unsegmented data. We split the feature set into character-level features and word-level features, and then build a character-based segmenter with character-level features and a word-based segmenter with word-level features, using the limited amount of available segmented data. These two segmenters then iteratively educate and improve each other by making use of the large amount of unsegmented data. Finally we combine the word-level and character-level features with an advanced segmenter to further improve the co-training performance. Our experiments show that using 10% data as segmented data and the other 90% data as unsegmented data, co-training reaches 20% performance improvement achieved by supervised training with all data in the SIGHAN 2005 PKU corpus and 31% in the CU corpus.

Acknowledgments

The authors thank Weiwei Sun for helping with data setup and technical consultation of the SunSegmenter. The authors also thank Christian Monson and Nicola Ueffing for helpful discussions.

References

Steven Abney. 2002. Bootstrapping. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 360–367.

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 465–472.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

Sanjoy Dasgupta, Michael L. Littman, and David Mcallester. 2001. Pac generalization bounds for co-training. In *Proceedings of Advances in Neural Information Processing Systems*, pages 375–382.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computationl Linguistics*, 31(4):574.

Umit Guz, Sebastien Cuendet, Dilek Hakkani-Tur, and Gokhan Tur. 2007. Co-training using prosodic and lexical information for sentence segmentation. In *proceedings of INTERSPEECH*, pages 2597–2600.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Comput. Linguist.*, 35:505–512, December.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, May.

Tetsuji Nakagawa. 2004. Chinese and japanese word segmentation using word-level and character-level information. In *Proceedings of the 20th international conference on Computational Linguistics*.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *In Proceedings of the 2001 Conference on*

Empirical Methods in Natural Language Processing, pages 1–9.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK., July.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: comparison and combination. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1211–1219.

W. J. Teahan, Rodger McNab, Yingying Wen, and Ian H. Witten. 2000. A compression-based algorithm for chinese word segmentation. *Computational Linguistics*, 26(3):375–393, September.

Gokhan Tur. 2009. Co-adaptation: Adaptive co-training for semi-supervised learning. In *proceedings of ICASSP*, pages 3721–3724.

Wen Wang, Zhongqiang Huang, and Mary Harper. 2007. Semi-supervised learning for part-of-speech tagging of mandarin transcribed speech. In *In ICASSP*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, pages 29–48.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech Republic, June. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for chinese word segmentation. *ACM Transactions on Asian Language Information Processing*, 9(2):5:1–5:32, June.

Ubertagging: Joint segmentation and supertagging for English

Rebecca Dridan
Institutt for Informatikk
Universitetet i Oslo
rdridan@ifi.uio.no

Abstract

A precise syntacto-semantic analysis of English requires a large detailed lexicon with the possibility of treating multiple tokens as a single meaning-bearing unit, a *word-with-spaces*. However parsing with such a lexicon, as included in the English Resource Grammar, can be very slow. We show that we can apply supertagging techniques over an ambiguous token lattice without resorting to previously used heuristics, a process we call ubertagging. Our model achieves an ubertagging accuracy that can lead to a four to eight fold speed up while improving parser accuracy.

1 Introduction and Motivation

Over the last decade or so, supertagging has become a standard method for increasing parser efficiency for heavily lexicalised grammar formalisms such as LTAG (Bangalore and Joshi, 1999), CCG (Clark and Curran, 2007) and HPSG (Matsuzaki et al., 2007). In each of these systems, fine-grained lexical categories, known as supertags, are used to prune the parser search space prior to full syntactic parsing, leading to faster parsing at the risk of removing necessary lexical items. Various methods are used to configure the degree of pruning in order to balance this trade-off.

The English Resource Grammar (ERG; Flickinger (2000)) is a large hand-written HPSG-based grammar of English that produces fine-grained syntacto-semantic analyses. Given the high level of lexical ambiguity in its lexicon, parsing with the ERG should therefore also benefit from supertagging, but while various attempts have shown possibilities (Blunsom, 2007; Dridan et al., 2008; Dridan, 2009), supertagging is still not a standard element in the ERG parsing pipeline. There are two main reasons for this. The first is that the ERG lexicon does not assign simple atomic categories to words, but instead builds complex structured *signs* from information about lemmas and lexical rules, and hence the shape and integration of the supertags is not straightforward. Bangalore and Joshi (2010) define a supertag as a primitive structure that contains all the information about a lexical item, including argument structure, and where the arguments should be found. Within the ERG, that information is not all contained in the lexicon, but comes from different places. The choice, therefore, of what information may be predicted prior to parsing and how it should be integrated into parsing is an open question.

The second reason that supertagging is not standard with ERG processing is one that is rarely considered when processing English, namely ambiguous segmentation. In most mainstream English parsing, the segmentation of parser input into tokens that will become the leaves of the parse tree is considered a fixed, unambiguous process. While recent work (Dridan and Oepen, 2012) has shown that producing even these tokens is not a solved problem, the issue we focus on here is the ambiguous mapping from these tokens to meaning-bearing units that we might call *words*. Within the ERG lexicon are many multi-token lexical entries that are sometimes referred to as *words-with-spaces*. These multi-token entries are added to the lexicon where the grammarian finds that the semantics of a fixed expression is non-compositional and has the distributional properties of other single word entries. Some examples include an adverb-like *all of a sudden*, a preposition-like *for example* and an adjective-like *over and done with*. Each of these entries create an segmentation ambiguity between treating the whole expression as a single unit, or allowing analyses comprising en-

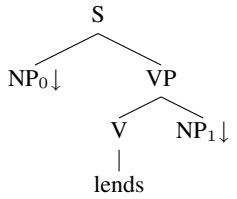
Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1201–1212,
Seattle, Washington, USA, 18-21 October 2013.

tries triggered by the individual tokens. Previous supertagging research using the ERG has either used the gold standard tokenisation, hence making the task artificially easier, or else tagged the individual tokens, using various heuristics to apply multi-token tags to single tokens. Neither approach has been wholly satisfactory.

In this work we avoid the heuristic approaches and learn a sequential classification model that can simultaneously determine the most likely segmentation and supertag sequences, a process we dub ubertagging. We also experiment with more fine-grained tag sets than have been previously used, and find that it is possible to achieve a level of ubertagging accuracy that can improve both parser speed and accuracy for a precise semantic parser.

2 Previous Work

As stated above, supertagging has become a standard tool for particular parsing paradigms, but the definitions of a supertag, the methods used to learn them, and the way they are used in parsing varies across formalisms. The original supertags were 300 LTAG elementary trees, predicted using a fairly simple trigram tagger that provided a configurable number of tags per token, since the tagger was not accurate enough to make assigning a single tree viable parser input (Bangalore and Joshi, 1999). The C&C CCG parser uses a more complex Maximum Entropy tagger to assign tags from a set of 425 CCG lexical categories (Clark and Curran, 2007). They also found it necessary to supply more than one tag per token, and hence assign all tags that have a probability within a percentage β of the most likely tag for each token. Their standard parser configuration uses a very restrictive β value initially, relaxing it when no parse can be found. Matsuzaki et al. (2007) use a supertagger similar to the C&C tagger alongside a CFG filter to improve the speed of their HPSG parser, feeding sequences of single tags to the parser until a parse is possible. As in the ERG, category and inflectional information are separate in the automatically-extracted ENJU grammar: their supertag set consists of 1361 tags constructed by combining lexical categories and lexical rules. Figure 1 shows examples of supertags from these three tag sets, all describing the simple transitive use of *lends*.



(a) LTAG

(S[dcl]\NP)/NP

(b) CCG

[NP.nom<V.bse>NP.acc]-singular3rd_verb_rule

(c) ENJU HPSG

Figure 1: Examples of supertags from LTAG, CCG and ENJU HPSG, for the word *lends*.

The ALPINO system for parsing Dutch is the closest in spirit to our ERG parsing setup, since it also uses a hand-written HPSG-based grammar, including multi-token entries in its lexicon. Prins and van Noord (2003) use a trigram HMM tagger to calculate the likelihood of up to 2392 supertags, and discard those that are not within τ of the most likely tag. For their multi-token entries, they assign a constructed category to each token, so that instead of assigning `preposition` to the expression *met betrekking tot* ("with respect to"), they use `(1,preposition)`, `(2,preposition)`, `(3,preposition)`. Without these constructed categories, they would only have 1365 supertags.

Most previous supertagging attempts with the ERG have used the grammar's lexical types, which describe the coarse-grained part of speech, and the subcategorisation of a word, but not the inflection. Hence both *lends* and *lent* have a possible lexical type `v_np*_pp*_to_le`, which indicates a verb, with optional noun phrase and prepositional phrase arguments, where the preposition has the form *to*. The number of lexical types changes as the grammar grows, and is currently just over 1000. Dridan (2009) and Fares (2013) experimented with other tag types, but both found lexical types to be the optimal balance between predictability and efficiency. Both used a multi-tagging approach dubbed selective tagging to integrate the supertags into the parser. This involved only applying the supertag filter when the tag probability is above a configurable threshold, and not pruning otherwise.

For multi-token entries, both Blunsom (2007) and

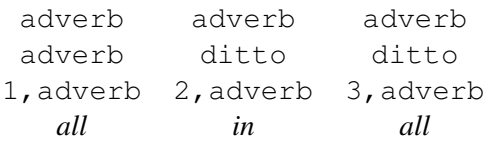


Figure 2: Options for tagging parts of the multi-token adverb *all in all* separately.

Dridan (2009) assigned separate tags to each token, with Blunsom (2007) assigning a special `ditto` tag all but the initial token of a multi-token entry, while Dridan (2009) just assigned the same tag to each token (leading to *example* in the expression *for example* receiving `p_np_i_le`, a preposition-type category). Both of these solutions (demonstrated in Figure 2), as well as that of Prins and van Noord (2003), in some ways defeat one of the purposes of treating these expressions as fixed units. The grammarian, by assigning the same category to, for example, *all of a sudden* and *suddenly*, is declaring that these two expressions have the same distributional properties, the properties that a sequential classifier is trying to exploit. Separating the tokens loses that information, and introduces extra noise into the sequence model.

Ytrestøl (2012) and Fares (2013) treat the multi-entry tokens as single expressions for tagging, but with no ambiguity. Ytrestøl (2012) manages this by using gold standard tokenisation, which is, as he states, the standard practice for statistical parsing, but is an artificially simplified setup. Fares (2013) is the only work we know about that has tried to predict the final segmentation that the ERG produces. We compare segmentation accuracy between our joint model and his stand-alone tokeniser in Section 6.

Looking at other instances of joint segmentation and tagging leads to work in non-whitespace separated languages such as Chinese (Zhang and Clark, 2010) and Japanese (Kudo et al., 2004). While at a high level, this work is solving the same problem, the shape of the problems are quite different from a data point of view. Regular joint morphological analysis and segmentation has much greater ambiguity in terms of possible segmentations but, in most cases, less ambiguity in terms of labelling than our situation. This also holds for other lemmatisation and morphological research, such as Toutanova and Cherry (2009). While we drew inspiration from this

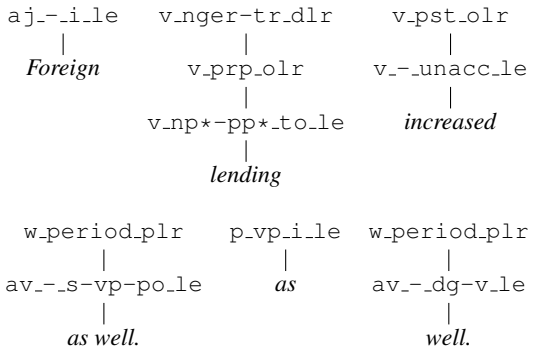


Figure 3: A selection from the 70 **lexitems** instantiated for *Foreign lending increased as well.*

related area, as well as from the speech recognition field, differences in the relative frequency of observations and labels, as well as in segmentation ambiguity mean that conclusions found in these areas did not always hold true in our problem space.

3 The Parser

The parsing environment we work with is the PET parser (Callmeier, 2000), a unification-based chart parser that has been engineered for efficiency with precision grammars, and incorporates subsumption-based ambiguity packing (Oepen and Carroll, 2000) and statistical model driven selective unpacking (Zhang et al., 2007). Parsing in PET is divided in two stages. The first stage, lexical parsing, covers everything from tokenising the raw input string to populating the base of the parse chart with the appropriate lexical items, ready for the second — syntactic parsing — stage. In this work, we embed our ubertagging model between the two stages. By this point, the input has been segmented into what we call `internal tokens`, which broadly means splitting at whitespace and hyphens, and making *'s* a separate token. These tokens are subject to a morphological analysis component which proposes possible inflectional and derivational rules based on word form, and then are used in retrieving possible lexical entries from the lexicon. The results of applying the appropriate lexical rules, plus affixation rules triggered by punctuation, to the lexical entries form a lexical item object, that for this work we dub a **lexitem**.

Figure 3 shows some examples of **lexitems** instantiated after the lexical parsing stage when

analysing *Foreign lending increased as well.* The pre-terminal labels on these subtrees are the lexical types that have previously been used as supertags for the ERG. For uninflected words, with no punctuation affixed, the lexical type is the only element in the **lexitem**, other than the word form (e.g. *Foreign*, *as*). In this example, we also see **lexitems** with inflectional rules (`v_prp_olr`, `v_pst_olr`), derivational rules (`v_nger-tr_dlr`) and punctuation affixation rules (`w_period_plr`).

These **lexitems** are put in to a chart, forming a lexical lattice, and it is over this lattice that we apply our ubertagging model, removing unlikely **lexitems** before they are seen by the syntactic parsing stage.

4 The Data

The primary data sets we use in these experiments are from the 1.0 version of DeepBank (Flickinger et al., 2012), an HPSG annotation of the Wall Street Journal text used for the Penn Treebank (PTB; Marcus et al. (1993)). The current version has gold standard annotations for approximately 85% of the first 22 sections. We follow the recommendations of the DeepBank developers in using Sections 00–19 for training, Section 20 (WSJ_{20}) for development and Section 21 (WSJ_{21}) as test data.

In addition, we use two further sources of training data: the training portions of the LinGO Redwoods Treebank (Oepen et al., 2004), a steadily growing collection of gold standard HPSG annotations in a variety of domains; and the Wall Street Journal section of the North American News Corpus (NANC), which has been parsed, but *not* manually annotated. This builds on observations by Prins and van Noord (2003), Dridan (2009) and Ytrestøl (2012) that even uncorrected parser output makes very good training data for a supertagger, since the constraints in the parser lead to viable, if not entirely correct sequences. This allows us to use much larger training sets than would be possible if we required manually annotated data.

In final testing, we also include two further data sets to observe how domain affects the contribution of the ubertagging. These are both from the test portion of the Redwoods Treebank: CatB, an essay about open-source software;[1] and WeScience_{13}, text from Wikipedia articles about Natural Language Processing from the WeScience project (Ytrestøl et al., 2009). Table 1 summarises the vital statistics of the data we use.

With the focus on multi-token **lexitems**, it is instructive to see just how frequent they are. In terms of **type** frequency, almost 10% of the approximately 38500 lexical entries in the current ERG lexicon have more than one token in their canonical form.[2] However, while this is a significant percentage of the lexicon, they do not account for the same percentage of **tokens** during parsing. An analysis of $\text{WSJ}_{00:19}$ shows that approximately one third of the sentences had at least one multi-token **lexitem** in the unpruned lexical lattice, and in just under half of those, the gold standard analysis included a multi-word entry. That gives the multi-token **lexitems** the awkward property of being rare enough to be difficult for a statistical classifier to accurately detect (just under 1% of the leaves of gold parse trees contain multiple tokens), but too frequent to ignore. In addition, since these multi-token expressions have often been distinguished because they are non-compositional, failing to detect the multi-word usage can lead to a disproportionately adverse effect on the semantic analysis of the text.

5 Ubertagging Model

Our ubertagging model is very similar to a standard trigram Hidden Markov Model (HMM), except that the states are not all of the same length. Our states are based on the **lexitems** in the lexical lattice produced by the lexical parsing stage of PET, and as such, can be partially overlapping. We formalise this be defining each state by its start position, end position, and tag. This turns out to make our model equivalent to a type of Hidden semi-Markov Model called a segmental HMM in Murphy (2002). In a segmental HMM, the states are segments with a tag (t) and a length in frames (l). In our setup, the frames are the ERG internal tokens and the segments are the **lexitems**, which are the potential candidates

[1] `http://catb.org/esr/writings/cathedral-bazaar/` by Eric S. Raymond

[2] While the parser has mechanisms for handling words unknown to the lexicon, with the current grammar these mechanisms will never propose a multi-token **lexitem**, and so only the multi-token entries explicitly in the lexicon will be recognised as such.

Data Set	Source	Use	Gold?	Trees	Lexitems All	Lexitems M-T
$WSJ_{00:19}$	DeepBank 1.0 §00–19	train	yes	33783	661451	6309
Redwoods	Redwoods Treebank	train	yes	39478	432873	6568
NANC	LDC2008T15	train	no	2185323	42376523	399936
WSJ_{20}	DeepBank 1.0 §20	dev	yes	1721	34063	312
WSJ_{21}	DeepBank 1.0 §21	test	yes	1414	27515	253
$WeScience_{13}$	Redwoods Treebank	test	yes	802	11844	153
CatB	Redwoods Treebank	test	yes	608	11653	115

Table 1: Test, development and training data used in these experiments. The final two columns show the total number of **lexitems** used for training (**All**), as well as how many of those were multi-token **lexitems** (**M-T**).

to become leaves of the parse tree. As indicated above, the majority of segments (over 99%) will be one frame long, but segments of up to four frames are regularly seen in the training data.

A standard trigram HMM has a transition probability matrix A, where the elements A_{ijk} represent the probability $P(k|ij)$, and an emission probability matrix B whose elements B_{jo} record the probabilities $P(o|j)$. Given these matrices and a vector of observed frames, O, the posterior probabilities of each state at frame v are calculated as:[3]

$$P(q_v = q_y|O) = \frac{\alpha_v(q_y)\beta_v(q_y)}{P(O)} \quad (1)$$

where $\alpha_v(q_y)$ is the forward probability at frame v, given a current state q_y (i.e. the probability of the observation up to v, given the state):

$$\alpha_v(q_y) \equiv P(O_{0:v}|q_v = q_y) \quad (2)$$

$$= \sum_{q_x} \alpha_v(q_x q_y) \quad (3)$$

$$\alpha_v(q_x q_y) = B_{q_y o_v} \sum_{q_w} \alpha_{v-1}(q_w q_x) A_{q_w q_x q_y} \quad (4)$$

$\beta_v(q_y)$ is the backwards probability at frame v, given a current state q_y (the probability of the observation from v, given the state):

$$\beta_v(q_y) \equiv P(O_{v+1:V}|q_v = q_y) \quad (5)$$

$$= \sum_{q_x} \beta_v(q_x q_y) \quad (6)$$

$$\beta_v(q_x q_y) = \sum_{q_z} \beta_{v+1}(q_y q_z) A_{q_x q_y q_z} B_{q_z o_{v+1}} \quad (7)$$

and the probability of the full observation sequence is equal to the forward probability at the end of the sequence, or the backwards probability at the start of the sequence:

$$P(O) = \alpha_V(\langle \mathcal{E} \rangle) = \beta_0(\langle \mathcal{S} \rangle) \quad (8)$$

In implementation, our model varies only in what we consider the previous or next states. While v still indexes frames, q_v now indicates a state that *ends* with frame v, and we look forwards and backwards to adjacent states, not frames, formally designated in terms of l, the length of the state. Hence, we modify equation (4):

$$\alpha_v(q_x q_y) = B_{q_y O_{v-l+1:v}} \sum_{q_w} \alpha_{v-l}(q_w q_x) A_{q_w q_x q_y} \quad (9)$$

where $v-l$ indexes the frame before the current state *starts*, and hence we are summing over all states that lead directly to our current state. An equivalent modification to equation (7) gives:

$$\beta_v(q_x q_y) = \sum_{\substack{q_z \\ \in Q_n}} \sum_{l(q_z)} \beta_{v+l}(q_y q_z) A_{q_x q_y q_z} B_{q_z O_{v+1:v+l}} \quad (10)$$

[3] Since we will require per-state probabilities for integration to the parser, we focus on the calculation of posterior probabilities, rather than determing the single best path.

Type	Example	#Tags
LTYPE	v_np-pp*_to_le	1028
INFL	v_np-pp*_to_le:v_pas_odlr	3626
FULL	v_np-pp*_to_le:v_pas_odlr:w_period_plr	21866

Figure 4: Possible tag types and their tag set size, with examples derived from the **lexitem** on the right.

where Q_n is the set of states that start at $v + 1$ (i.e., the states immediately following the current state), and $l(q_z)$ is the length of state q_z.

We construct the transition and emission probability matrices using relative frequencies directly observed from the training data, where we make the simplifying assumption that $P(q_k|q_iq_j) \equiv P(t(q_k)|t(q_i)t(q_k))$. Which is to say, while **lexitems** with the same tag, but different length will trigger distinct states with distinct emission probabilities, they will have the same transition probabilities, given the same proceeding tag.[4] Even with our large training set, some tag trigrams are rare or unseen. To smooth these probabilities, we use deleted interpolation to calculate a weighted sum of the trigram, bigram and unigram probabilities, since it has been successfully used in effective PoS taggers like the TnT tagger (Brants, 2000). Future work will look more closely at the effects of different smoothing methods.

6 Intrinsic Ubertag Evaluation

In order to develop and tune the ubertagging model, we first looked at segmentation and tagging performance in isolation over the development set. We looked at three tag granularities: lexical types (LTYPE) which have previously been shown to be the optimal granularity for supertagging with the ERG, inflected types (INFL) which encompass inflectional and derivational rules applied to the lexical type, and the full lexical item (FULL), which also includes affixation rules used for punctuation handling. Examples of each tag type are shown in Figure 4, along with the number of tags of each type seen in the training data.

[4]Since the multi-token lexical entries are defined because they have the same properties as the single token variants, there is no reason to think the length of a state should influence the tag sequence probability.

	Segmentation		Tagging	
Tag Type	F1	Sent.	F1	Sent.
FULL	99.55	94.48	93.92	42.13
INFL	99.45	93.55	93.74	41.49
LTYPE	99.40	93.03	93.27	38.12

Table 2: Segmentation and tagging performance of the best path found for each model, measured per segment in terms of F_1, and also as complete sentence accuracy.

Single sequence results Table 2 shows the results when considering the best path through the lattice. In terms of segmentation, our sentence accuracy is comparable to that of the stand-alone segmentation performance reported by Fares et al. (2013) over similar data.[5] In that work, the authors used a binary CRF classifier to label points between objects they called micro-tokens as either SPLIT or NOSPLIT. The CRF classifier used a less informed input (since it was external to the parser), but a much more complex model, to produce a best single path sentence accuracy of 94.06%. Encouragingly, this level of segmentation performance was shown in later work to produce a viable parser input (Fares, 2013).

Switching to the tagging results, we see that the F_1 numbers are quite good for tag sets of this size.[6] The best tag accuracy seen for ERG LTYPE-style tags was 95.55 in Ytrestøl (2012), using gold standard segmentation on a different data set. Dridan (2009) experimented with a tag granularity similar to our INFL (letype+morph) and saw a tag accuracy of 91.51, but with much less training data. From other formalisms, Kummerfeld et al. (2010)

[5]Fares et al. (2013) used a different section of an earlier version of DeepBank, but with the same style of annotation.

[6]We need to measure F_1 rather than tag accuracy here, since the number of tokens tagged will vary according to the segmentation.

report a single tag accuracy of 95.91, with the smaller CCG supertag set. Despite the promising tag F_1 numbers however, the sentence level accuracy still indicates a performance level unacceptable for parser input. Comparing between tag types, we see that, possibly surprisingly, the more fine-grained tags are more accurately assigned, although the differences are small. While instinctively a larger tag set should present a more difficult problem, we find that this is mitigated both by the sparse lexical lattice provided by the parser, and by the extra constraints provided by the more informative tags.

Multi-tagging results The multi-tagging methods from previous supertagging work becomes more complicated when dealing with ambiguous tokenisation. Where, in other setups, one can compare tag probabilities for all tags for a particular token, that no longer holds directly when tokens can partially overlap. Since ultimately, the parser uses **lexitems** which encompass segmentation and tagging information, we decided to use a simple integration method, where we *remove* any **lexitem** which our model assigns a probability below a certain threshold (ρ). The effect of the different tag granularities is now mediated by the relationship between the states in the ubertagging lattice and the **lexitems** in the parser's lattice: for the FULL model, this is a one-to-one relationship, but states from the models that use coarser-grained tags may affect multiple **lexitems**. To illustrate this point, Figure 5 shows some **lexitems** for the token *forecast,*, where there are multiple possible analyses for the comma. A FULL tag of `v_cp_le:v_pst_olr:w_comma_plr` will select only **lexitem** (b), whereas an INFL tag `v_cp_le:v_pst_olr` will select (b) and (c) and the LTYPE tag `v_cp_le` picks out (a), (b) and (c). On the other hand, where there is no ambiguity in inflection or affixation, an LTYPE tag of `n_-_mc_le` may relate to only a single **lexitem** ((f) in this case).

Since we are using an absolute, rather than relative, threshold, the number needs to be tuned for each model[7] and comparisons between models can only be made based on the effects (accuracy or pruning power) of the threshold. Table 3 shows how a selection of threshold values affect the accuracy

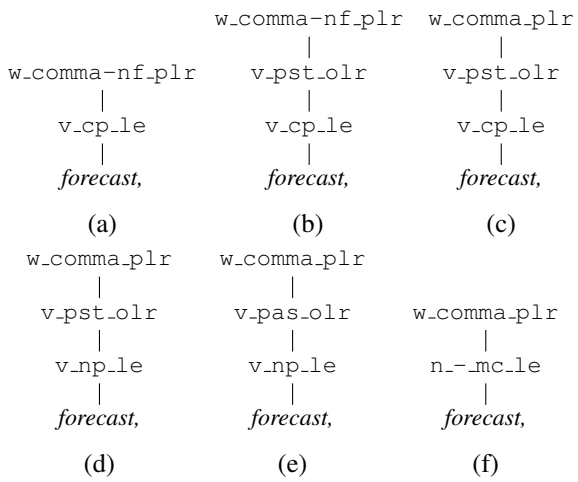


Figure 5: Some of the **lexitems** triggered by *forecast,* in *Despite the gloomy forecast, profits were up.*

Tag Type	ρ	Acc.	Lexitems Kept	Lexitems Ave.
FULL	0.00001	99.71	41.6	3.34
FULL	0.0001	99.44	33.1	2.66
FULL	0.001	98.92	25.5	2.05
FULL	0.01	97.75	19.4	1.56
INFL	0.0001	99.67	37.9	3.04
INFL	0.001	99.25	29.0	2.33
INFL	0.01	98.21	21.6	1.73
INFL	0.02	97.68	19.7	1.58
LTYPE	0.0002	99.75	66.3	5.33
LTYPE	0.002	99.43	55.0	4.42
LTYPE	0.02	98.41	43.5	3.50
LTYPE	0.05	97.54	39.4	3.17

Table 3: Accuracy and ambiguity after pruning **lexitems** in WSJ_{20}, at a selection of thresholds ρ for each model. Accuracy is measured as the percentage of gold **lexitems** remaining after pruning, while ambiguity is presented both as a percentage of **lexitems** kept, and the average number of **lexitems** per initial token still remaining.

[7] A tag set size of 1028 will lead to higher probabilities in general than a tag set size of 21866.

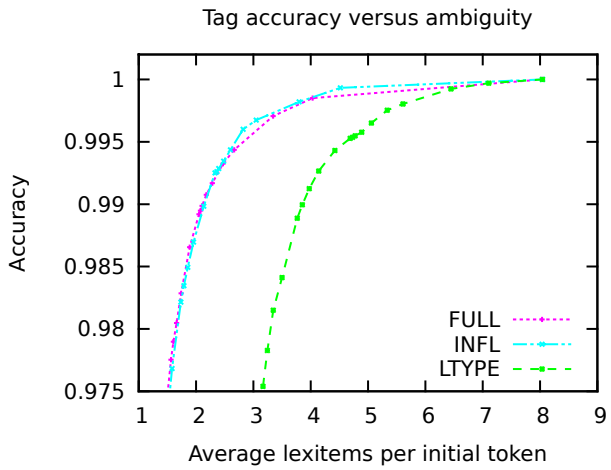


Figure 6: Accuracy over gold **lexitems** versus average **lexitems** per initial token over the development set, for each of the different ubertagging models.

and pruning impact of our different disambiguation models, where the accuracy is measured in terms of percentage of gold **lexitems** retained. The pruning effect is given both as percentage of **lexitems** retained after pruning, and average number of **lexitems** per initial token.[8] Comparison between the different models can be more easily made by examining Figure 6. Here we see clearly that the LTYPE model provides much less pruning for any given level of **lexitem** accuracy, while the performance of the other models is almost indistinguishable.

Analysis The current state-of-the-art POS tagging accuracy (using the 45 tags in the PTB) is approximately 97.5%. The most restrictive ρ value we report for each model was selected to demonstrate that level of accuracy, which we can see would lead to pruning over 80% of **lexitems** when using FULL tags, an average of 1.56 tags per token. While this level of accuracy has been sufficient for statistical treebank parsing, previous work (Dridan, 2009) has shown that tag accuracy cannot directly predict parser performance, since errors of different types can have very different effects. This is hard to quantify without parsing, but we made a qualitative analysis at the **lexitems** that were incorrectly being pruned. For all models, the most difficult **lexitems** to get correct were proper nouns, particular those that are also used as common nouns (e.g. *Bank*, *Airline*, *Report*). While capitalisation provides a clue here, it is not always deterministic, particularly since the treebank incorporates detailed decisions regarding the distinction between a name and a capitalised common noun that require real world knowledge, and are not necessarily always consistent. Almost two thirds of the errors made by the FULL and INFL models are related to these decisions, but only about 40% for the LTYPE model. The other errors are predominately over noun and verb type **lexitems**, as the open classes, with the only difference between models being that the FULL model seems marginally better at classifying verbs. The next section describes the end-to-end setup and results when parsing the development set.

7 Parsing

With encouraging ubertagging results, we now take the next step and evaluate the effect on end-to-end parsing. Apart from the issue of different error types having unpredictable effects, there are two other factors that make the isolated ubertagging results only an approximate indication of parsing performance. The first confounding factor is the statistical parsing disambiguation model. To show the effect of ubertagging in a realistic configuration, we only evaluate the first analysis that the parser returns. That means that when the unrestricted parser does not rank the gold analysis first, errors made by our model may not be visible, because we would never see the gold analysis in any case. On the other hand, it is possible to improve parser accuracy by pruning incorrect **lexitems** that were in a top ranked, non-gold analysis.

The second new factor that parser integration brings to the picture is the effect of resource limitations. For reasons of tractability, PET is run with per sentence time and memory limits. For treebank creation, these limits are quite high (up to four minutes), but for these experiments, we set the timeout to a more practical 60 seconds and the memory limit to 2048Mb. Without lexical pruning, this leads to approximately 3% of sentences not receiving an analysis. Since the main aim of ubertagging is to in-

[8] The average number of **lexitems** per token for the unrestricted parser is 8.03, although the actual assignment is far from uniform, with up to 70 **lexitems** per token seen for the very ambiguous tokens.

Tag Type	ρ	F_1 Lexitem	Bracket	Time
No Pruning		94.06	88.58	6.58
FULL	0.00001	95.62	89.84	3.99
FULL	0.0001	95.95	90.09	2.69
FULL	0.001	95.81	89.88	1.34
FULL	0.01	94.19	88.29	0.64
INFL	0.0001	96.10	90.37	3.45
INFL	0.001	96.14	90.33	1.78
INFL	0.01	95.07	89.27	0.84
INFL	0.02	94.32	88.49	0.64
LTYPE	0.0002	95.37	89.63	4.73
LTYPE	0.002	96.03	90.20	2.89
LTYPE	0.02	95.04	89.04	1.23
LTYPE	0.05	93.36	87.26	0.88

Table 4: **Lexitem** and bracket F_1over WSJ_{20}, with average per sentence parsing time in seconds.

crease efficiency, we would expect to regain at least some of these unanalysed sentences, even when a **lexitem** needed for the gold analysis has been removed.

Table 4 shows the parsing results at the same threshold values used in Table 3. Accuracy is calculated in terms of F_1 both over **lexitems**, and PARSEVAL-style labelled brackets (Black et al., 1991), while efficiency is represented by average parsing time per sentence. We can see here that an ubertagging F_1 of below 98 (cf. Table 3) leads to a drop in parser accuracy, but that an ubertagging performance of between 98 and 99 can improve parser F_1 while also achieving speed increases up to 8-fold.

From the table we confirm that, contrary to earlier pipeline supertagging configurations, tags of a finer granularity than LTYPE can deliver better performance, both in terms of accuracy and efficiency. Again, comparing graphically in Figure 7 gives a clearer picture. Here we have graphed labelled bracket F_1 against parsing time for the full range of threshold values explored, with the unpruned parsing results indicated by a cross.

From this figure, we see that the INFL model, despite being marginally less accurate when measured in isolation, leads to slightly more accurate parse results than the FULL model at all levels of efficiency.

Looking at the same graph for different samples of the development set (not shown) shows some

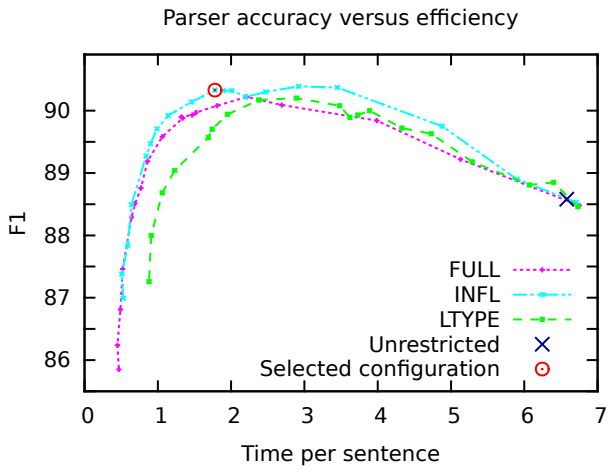


Figure 7: Labelled bracket F_1 versus parsing time per sentence over the development set, for each of the different ubertagging models. The cross indicates unpruned performance, while the circle pinpoints the configuration we chose for the final test runs.

variance in which threshold value gives the best F_1, but the relative differences and basic curve shape remains the same. From these different views, using the guideline of maximum efficiency without harming accuracy we selected our final configuration: the INFL model with a threshold value of 0.001 (marked with a circle in Figure 7). On the development set, this configuration leads to a 1.75 point improvement in F_1 in 27% of the parsing time.

8 Final Results

Table 5 shows the results obtained when parsing using the configuration selected on the development set, over our three test sets. The first, WSJ_{21} is from the same domain as the development set. Here we see that the effect over the WSJ_{21} set fairly closely mirrored that of the development set, with an F_1 increase of 1.81 in 29% of the parsing time.

The Wikipedia domain of our $WeScience_{13}$ test set, while very different to the newswire domain of the development set could still be considered in domain for the parsing and ubertagging models, since there is Wikipedia data in the training sets. With an average sentence length of 15.18 (compared to 18.86 in WSJ_{21}), the baseline parsing time is faster than for WSJ_{21}, and the speedup is not quite as large

	Baseline		Pruned	
Data Set	**F_1**	**Time**	**F_1**	**Time**
WSJ_{21}	88.12	6.06	89.93	1.77
$WeScience_{13}$	86.25	4.09	87.14	1.48
CatB	86.31	5.00	87.11	1.78

Table 5: Parsing accuracy in terms of labelled bracket F_1 and average time per sentence when parsing the test sets, without pruning, and then with lexical pruning using the INFL model with a threshold of 0.001.

but still welcome, at 36% of the baseline time. The increase is accuracy is likewise smaller (due to less issues with resource exhaustion in the baseline), but as our primary goal is to not *harm* accuracy, the results are pleasing.

The CatB test set is the standard out-of-domain test for the parser, and is also out of domain for the ubertagging model. The average sentence length is not much below that of WSJ_{21}, at 18.61, but the baseline parsing speed is still noticeably faster, which appears to be a reflection of greater structural ambiguity in the newswire text. We still achieve a reduction in parsing time to 35% of the baseline, again with a small improvement in accuracy.

The across-the-board performance improvement on all our test sets suggests that, while tuning the pruning threshold could help, it is a robust parameter that can provide good performance across a variety of domains. This means that we finally have a robust supertagging setup for use with the ERG that doesn't require heuristic shortcuts and can be reliably applied in general parsing.

9 Conclusions and Outlook

In this work we have demonstrated a lexical disambiguation process dubbed ubertagging that can assign fine-grained supertags over an ambiguous token lattice, a setup previously ignored for English. It is the first completely integrated supertagging setup for use with the English Resource Grammar, which avoids the previously necessary heuristics for dealing with ambiguous tokenisation, and can be robustly configured for improved performance without loss of accuracy. Indeed, by learning a joint segmentation and supertagging model, we have been able to achieve usefully high tagging accuracies for very fine-grained tags, which leads to potential parser speedups of between 4 and 8 fold.

Analysis of the tagging errors still being made have suggested some possibly avoidable inconsistencies in the grammar and treebank, which have been fed back to the developers, hopefully leading to even better results in the future.

In future work, we will investigate more advanced smoothing methods to try and boost the ubertagging accuracy. We also intend to more fully explore the domain adaptation potentials of the lexical model that have been seen in other parsing setups (see Rimell and Clark (2008) for example), as well as examine the limits on the effects of more training data. Finally, we would like to explore just how much the statistic properties of our data dictate the success of the model by looking at related problems like morphological analysis of unsegmented languages such as Japanese.

Acknowledgements

I am grateful to my colleagues from the Oslo Language Technology Group and the DELPH-IN consortium for many discussions on the issues involved in this work, and particularly to Stephan Oepen who inspired the initial lattice tagging idea. Thanks also to three anonymous reviewers for their very constructive feedback which improved the final version. Large-scale experimentation and engineering is made possible though access to the TITAN high-performance computing facilities at the University of Oslo, and I am grateful to the Scientific Computating staff at UiO, as well as to the Norwegian Metacenter for Computational Science and the Norwegian tax payer.

References

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

Srinavas Bangalore and Aravind Joshi, editors. 2010. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*. The MIT Press, Cambridge, US.

Ezra Black, Steve Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Phil Harrison, Don Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, S. Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Workshop on Speech and Natural Language*, page 306 – 311, Pacific Grove, USA.

Philip Blunsom. 2007. *Structured Classification for Multilingual Natural Language Processing*. Ph.D. thesis, Department of Computer Science and Software Engineering, University of Melbourne.

Thorsten Brants. 2000. TnT — a statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing ANLP-2000*, page 224 – 231, Seattle, USA.

Ulrich Callmeier. 2000. PET. A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99 – 108, March.

Stephen Clark and James R. Curran. 2007. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, page 248 – 255, Prague, Czech Republic.

Rebecca Dridan and Stephan Oepen. 2012. Tokenization. Returning to a long solved problem. A survey, contrastive experiment, recommendations, and toolkit. In *Proceedings of the 50th Meeting of the Association for Computational Linguistics*, page 378 – 382, Jeju, Republic of Korea, July.

Rebecca Dridan, Valia Kordoni, and Jeremy Nicholson. 2008. Enhancing performance of lexicalised grammars. page 613 – 621.

Rebecca Dridan. 2009. *Using lexical statistics to improve HPSG parsing*. Ph.D. thesis, Department of Computational Linguistics, Saarland University.

Murhaf Fares, Stephan Oepen, and Yi Zhang. 2013. Machine learning for high-quality tokenization. Replicating variable tokenization schemes. In *Computational Linguistics and Intelligent Text Processing*, page 231 – 244. Springer.

Murhaf Fares. 2013. ERG tokenization and lexical categorization: a sequence labeling approach. Master's thesis, Department of Informatics, University of Oslo.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank. A dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, page 85 – 96, Lisbon, Portugal. Edições Colibri.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1):15 – 28.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, page 230 – 237.

Jonathan K. Kummerfeld, Jessika Roesner, Tim Dawborn, James Haggerty, James R. Curran, and Stephen Clark. 2010. Faster parsing by supertagger adaptation. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*, page 345 – 355, Uppsala, Sweden.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpora of English: The Penn Treebank. *Computational Linguistics*, 19:313 – 330.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2007)*, page 1671 – 1676, Hyderabad, India.

Kevin P. Murphy. 2002. Hidden semi-Markov models (HSMMs).

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing. Practical results. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, page 162 – 169, Seattle, WA, USA.

Stephan Oepen, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Research on Language and Computation*, 2(4):575 – 596.

Robbert Prins and Gertjan van Noord. 2003. Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121 – 139.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. page 475 – 484.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics*, page 486 – 494, Singapore.

Gisle Ytrestøl. 2012. *Transition-based Parsing for Large-scale Head-Driven Phrase Structure Grammars*. Ph.D. thesis, Department of Informatics, University of Oslo.

Gisle Ytrestøl, Stephan Oepen, and Dan Flickinger. 2009. Extracting and annotating Wikipedia subdomains. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, page 185 – 197, Groningen, The Netherlands.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, page 843 – 852, Cambridge, MA, USA.

Yi Zhang, Stephan Oepen, and John Carroll. 2007. Efficiency in unification-based n-best parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, page 48 – 59, Prague, Czech Republic, July.

Automatic Knowledge Acquisition for Case Alternation between the Passive and Active Voices in Japanese

Ryohei Sasano[1] **Daisuke Kawahara**[2] **Sadao Kurohashi**[2] **Manabu Okumura**[1]
[1] Precision and Intelligence Laboratory, Tokyo Institute of Technology
[2] Graduate School of Informatics, Kyoto University
{sasano,oku}@pi.titech.ac.jp, {dk,kuro}@i.kyoto-u.ac.jp

Abstract

We present a method for automatically acquiring knowledge for case alternation between the passive and active voices in Japanese. By leveraging several linguistic constraints on alternation patterns and lexical case frames obtained from a large Web corpus, our method aligns a case frame in the passive voice to a corresponding case frame in the active voice and finds an alignment between their cases. We then apply the acquired knowledge to a case alternation task and prove its usefulness.

1 Introduction

Predicate-argument structure analysis is one of the fundamental techniques for many natural language applications such as recognition of textual entailment, information retrieval, and machine translation. In Japanese, the relationship between a predicate and its argument is usually represented by using case particles[1] (Kawahara and Kurohashi, 2006; Taira et al., 2008; Yoshikawa et al., 2011). However, since case particles vary depending on the voices, we have to take case alternation into account to represent predicate-argument structure. There are thus two major types of representations: one uses surface cases, and the other uses normalized-cases for the base form of predicates. For example, while the Kyoto University Text Corpus (Kawahara et al., 2004), one of the major Japanese corpora that contains annotations of predicate-argument structures, adopts the former representation, the NAIST Text Corpora (Iida et al., 2007), another major Japanese corpus, adopts the latter representation.

Examples (1) and (2) describe the same event in the passive and active voices, respectively. When we use surface cases to represent the relationship between the predicate and its argument in Example (1), the case of "女 (woman)" is *ga*[2] and the case of "男 (man)" is *ni*.[2] On the other hand, when we use the normalized-cases for the base form, the case of "女 (woman)" is *wo*[2] and the case of "男 (man)" is *ga*, which are the same as the surface cases in the active voice as in Example (2).

(1) 女が (woman-*ga*) 男に (man-*ni*) 突き落とされた. (was pushed down)
(A woman was pushed down by a man.)

(2) 男が (man-*ga*) 女を (woman-*wo*) 突き落とした. (pushed down)
(A man pushed down a woman.)

Both representations have their own advantages. Surface case analysis is easier than normalized-case analysis, especially when we consider omitted arguments, which are also called zero anaphors (Nagao and Hasida, 1998). In Japanese, zero anaphora frequently occurs, and the omitted unnormalized-case of a zero anaphor is often the same as the surface case of its antecedent (Sasano and Kurohashi, 2011). Therefore, surface case analysis suits zero anaphora resolution. On the other hand, when

[1] Japanese is a head-final language. Word order does not mark syntactic relations. Instead, postpositional case particles function as case markers.

[2] *Ga*, *wo*, and *ni* are typical Japanese postpositional case particles. In most cases, they indicate nominative, accusative, and dative, respectively.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1213–1223,
Seattle, Washington, USA, 18-21 October 2013.

we focus on the resulting predicate argument structures, the normalized-case structure is more useful. Specifically, since a normalized-case structure represents the same meaning in the same representation, normalized-case analysis is useful for recognizing textual entailment and information retrieval.

Therefore, we need a system that first analyzes surface cases and then alternates the surface cases with normalized-cases. In particular, we focus on the transformation of the passive voice into the active voice in this paper. Passive-to-active voice transformation in English can be performed systematically, which does not depend on lexical information in most cases. However, in Japanese, the method of transformation depends on lexical information. For example, while the case particle *ni* in Example (1) is alternated with *ga* in the active voice, the case particle *ni* in Example (3) is not alternated in the active voice as in Example (4) even though both their predicates are "突き落とされた (be pushed down)."

(3) 女が 海に 突き落とされた.
woman-*ga* sea-*ni* was pushed down
(A woman was pushed down into the sea.)

(4) 女を 海に 突き落とした.
woman-*wo* sea-*ni* pushed down
(ϕ pushed down a woman into the sea.)

The *ni* case in Example (1) indicates *agent*. On the other hand, the *ni* case in Example (3) indicates *direction*. To determine the difference is important for many NLP applications including machine translation. In fact, Google Translate (GT)[3] translates Examples (1) and (3) as "Woman was pushed down in the man" and "Woman was pushed down in the sea," respectively, which may be because GT cannot distinguish between the roles of *ni* in Examples (1) and (3).

(5) 賞が 男に 贈られた.
prize-*ga* man-*ni* was awarded
(A prize was awarded to a man.)

In example (5), although the *ni*-case argument "男 (man)" is the same as in Example (1), the case particle *ni* indicates *recipient* and is not alternated in the active voice. These examples show that case alternation between the passive and active voices in Japanese depends on not only predicates but also arguments, and we have to consider their combinations. Since it is impractical to manually describe the case alternation rules for all combinations of predicates and arguments, we have to acquire such knowledge automatically.

Thus, in this paper, we present a method for acquiring the knowledge for case alternation between the passive and active voices in Japanese. Our method leverages several linguistic constraints on alternation patterns and lexical case frames obtained from a large Web corpus, which are constructed for each meaning and voice of each predicate.

2 Related Work

Levin (1993) grouped English verbs into classes on the basis of their shared meaning components and syntactic behavior, defined in terms of diathesis alternations. Hence, diathesis alternations have been the topic of interest for a number of researchers in the field of automatic verb classification, which aims to induce possible verb frames from corpora (e.g., McCarthy 2000; Lapata and Brew 2004; Joanis et al. 2008; Schulte im Walde et al. 2008; Li and Brew 2008; Sun and Korhonen 2009; Theijssen et al. 2012). Baroni and Lenci (2010) used distributional slot similarity to distinguish between verbs undergoing the causative-inchoative alternations, and verbs that do not alternate.

There is some work on passive-to-active voice transformation in Japanese. Baldwin and Tanaka (2000) empirically identified the range and frequency of basic verb alternation, including active-passive alternation, in Japanese. They automatically extracted alternation types by using hand-crafted case frames but did not evaluate the quality. Kondo et al. (2001) dealt with case alternation between the passive and active voices as a subtask of paraphrasing a simple sentence. They manually introduced case alternation rules on the basis of verb types and case patterns and transformed passive sentences into active sentences.

Murata et al. (2006) developed a machine-learning-based method for Japanese case alternation. They extracted 3,576 case particles in passive sentences from the Kyoto University Text Corpus

[3] http://translate.google.com, accessed 2013-2-20.

Case particle	Grammatical function
ga	nominative
wo	accusative
ni	dative
de	locative, instrumental
kara	ablative
no	genitive

Table 1: Examples of Japanese postpositional case particles and their typical grammatical functions.

and tagged their cases in the active voice. Then, they trained SVM classifiers using the tagged corpus. Their features for training SVM were made by using several lexical resources such as IPAL (IPA, 1987), the Japanese thesaurus *Bunrui Goi Hyo* (NLRI, 1993), and the output of Kondo et al.'s method.

3 Lexicalized Case Frames

To acquire knowledge for case alternation, we exploit lexicalized case frames that are automatically constructed from 6.9 billion Web sentences by using Kawahara and Kurohashi (2002)'s method. In short, their method first parses the input sentences, and then constructs case frames by collecting reliable modifier-head relations from the resulting parses.

These case frames are constructed for each predicate like PropBank frames (Palmer et al., 2005), for each meaning of the predicate like FrameNet frames (Fillmore et al., 2003), and for each voice. However, neither pseudo-semantic role labels such as Arg1 in PropBank nor information about frames defined in FrameNet are included in these case frames. Each case frame describes surface cases that each predicate has and instances that can fill a case slot, which is fully lexicalized like the subcategorization lexicon VALEX (Korhonen et al., 2006).

We list some Japanese postpositional case particles with their typical grammatical functions in Table 1 and show examples of case frames in Table 2.[4] Ideally, one case frame is constructed for each meaning and voice of the target predicate. However, since Kawahara and Kurohashi's method is unsupervised, several case frames are actually constructed

[4] *Niyotte* in Table 2 is a Japanese functional phrase that indicates *agent* in this case. We treat *niyotte* as a case particle in this paper for the sake of simplicity.

Case Frame: "突き落とされる-4 (be pushed down-4)"
{ 女性 (woman):5, 僕 (I):2, 女 (woman):2, · · · }-*ga*
{ 海 (sea):229, 川 (bottom):115, 池 (pond):51, · · · }-*ni*
{ 継母(stepmother):2, ペガサス(Pegasus):2, · · · }-*niyotte*
· · ·

Case Frame: "突き落とされる-5 (be pushed down-5)"
{ 京子 (Kyoko):3, 監督 (manager):1, · · · }-*ga*
{ 誰か (someone):143, 何者か (somebody):85, · · · }-*ni*
{ 階段 (stair):20, 船 (ship):7, 崖 (cliff):7, · · · }-*kara*
· · ·

Case Frame: "突き落とす-2 (push down-2)"
{ 男 (man):14, 獅子 (lion):5, 虎 (tiger):3, · · · }-*ga*
{ 子(child):316, 子供(child):81, 人(person):51, · · · }-*wo*
{ 海 (sea):580, 谷 (ravine):576, 川 (river):352 · · · }-*ni*
· · ·

Case Frame: "突き落とす-4 (push down-4)"
{ 誰か (someone):14, ライオン (lion):5, · · · }-*ga*
{ 人 (person):257, 私 (I):214, 子 (child):137, · · · }-*wo*
{ 崖 (cliff):53, 階段 (stair):28, · · · }-*kara*
· · ·

Table 2: Examples of case frames for "突き落とされる (be pushed down)" and "突き落とす (push down)." Words in curly braces denote instances that can fill corresponding cases and the numbers following these words denote their frequency in the corpus.

for each meaning and voice. For example, 59 and eight case frames were respectively constructed for the predicate in the passive voice "突き落とされる (be pushed down)" and in the active voice "突き落とす (push down)" from 6.9 billion Web sentences. Table 2 shows the 4th and 5th case frames for "突き落とされる (be pushed down)" and the 2nd and 4th case frames for "突き落とす (push down)."

Table 3 shows an example of case frames for "殴る (hit)," which includes *no*-case. Here, the Japanese postpositional case particle "*no*" roughly corresponds to "of," that is, "*X no Y*" means "Y of X," and thus *no*-case is not an argument of the target predicate. While Kawahara and Kurohashi's method basically collects arguments of the target predicate, the phrase of *no*-case that modifies the direct object of the predicate is also collected as *no*-case. This is because, as we will show in the next section, this phrase can be represented as *ga*-case in the passive voice.

Case Frame: “殴る-2 (hit-2)”
{ 男 (man):51, 拳 (fist):30, 誰か (someone):23, · · · }-*ga*
{ 自分 (myself):360, 私 (I):223, · · · }-*no*
{ 頭 (head):5424, 顔 (face):3215, · · · }-*wo*
{ 拳 (fist):316, 平手 (palm):157, 拳骨 (fist):126, · · · }-*de*
· · ·

Table 3: An example of case frames for “殴る (hit).”

4 Passive-Active Transformation in Japanese

Morphologically speaking, the passive voice in Japanese is expressed by using the auxiliary verbs “れる (*reru*)” and “られる (*rareru*),” whose past forms are “れた (*reta*)” and “られた (*rareta*),” respectively. For example, the verb in the base form “突き落とす (*tsukiotosu*, push down)” is transformed into the past passive form “突き落とされた (*tsukiotosa-reta*, was pushed down).” Case alternations accompany passive-active transformation in Japanese. There are only two case alternations at most in passive-active transformation. One is the case represented as *ga* in the passive voice, and the other is the case represented as *ga* in the active voice.

Japanese passive sentences can be classified into three types in accordance with what is represented as *ga*-case in the passive voice: **direct passive**, **indirect passive**, and **possessor passive**.

In **direct passive** sentence, the object of the predicate in the active voice is represented as *ga*-case. Examples (1), (3), and (5) are all direct passive sentences. The case that is represented as *ga* in the active voice is usually represented as *ni*, *niyotte*, *kara*, or *de* in the passive sentence. In the first sentence of Examples (6) and (7),[5] *ga*-cases in the active voice are represented as *niyotte* and *kara*, respectively. On the other hand, *ga*-case in the passive sentence is alternated with *wo* or *ni* as shown with broken lines in the second sentence of Examples (6) and (7).

(6) P: 原因が 男によって 特定された.
cause-*ga* man-*niyotte* was identified
(The cause was identified by a man.)

A: 男が 原因を 特定した.
man-*ga* cause-*wo* identified
(A man identified the cause.)

[5]“P” denotes a passive sentence and “A” denotes the corresponding active sentence in these examples.

(7) P: 男が 女から 話しかけられた.
man-*ga* woman-*kara* was talked to
(A man was talked to by a woman.)

A: 女が 男に 話しかけた.
woman-*ga* man-*ni* talked to
(A woman talked to a man.)

Indirect passive is also called adversative passive, in which an indirectly influenced agent is represented with *ga*. For example, “私 (I),” the argument represented with *ga* in the first sentence of Example (8), does not appear in the active voice, i.e. the second sentence of Example (8). In the case of indirect passive, *ga*-case in the active sentence is always alternated with *ni*-case in the passive sentence as shown with solid lines in Examples (8).

(8) P: 私が 子供に 泣かれた.
I-*ga* child-*ni* was cried
(I’ve got a child crying.)

A: 子供が 泣いた. (A child cried.)
child-*ga* cried

Possessor passive is similar to indirect passive in that the argument represented with *ga*-case does not appear as an argument of the predicate in the active voice. Therefore, possessor passive is sometimes treated as a kind of indirect passive. However, in the case of possessor passive, the argument appears in the active sentence as a possessor of the direct object. For example, the *ga*-case argument “女 (woman)” in the passive sentence of Example (9) does not appear as an argument of the predicate “殴った (hit)” in the active sentence but appears in the phrase that modifies the direct object “頭 (head)” with the case particle *no*, which indicates that “女 (woman)” is the possessor of “頭 (head).”

(9) P: 女が 男に 頭を 殴られた.
woman-*ga* man-*ni* head-*wo* was hit
(A woman was hit on the head by a man.)

A: 男が 女の 頭を 殴った.
man-*ga* woman-*no* head-*wo* hit
(A man hit the head of a woman.)

In conclusion, the number of case alternation patterns accompanying passive-active transformation in Japanese is limited. *Ga*-case in the passive voice can

be alternated only with either *wo*, *ni*, or *no*, or does not appear in the active voice. *Ga*-case in the active voice can be represented only by *ni*, *niyotte*, *kara*, or *de* in the passive voice. Hence, it is sufficient to consider only their combinations.

5 Knowledge Acquisition for Case Alternation

5.1 Task Definition

Our objective is to acquire knowledge for case alternation between the passive and active voices in Japanese. We leverage lexical case frames obtained from a large Web corpus by using Kawahara and Kurohashi (2002)'s method and align cases of a case frame in the passive voice and cases of a case frame in the active voice. As described in Section 2, several case frames are constructed for each voice of each predicate. Our task consists of the following two subtasks:

1. Identify a corresponding case frame in the active voice.
2. Find an alignment between cases of case frames in the passive and active voice.

Figure 1 shows the overview of our task. If a case frame in the passive voice is input, we identify a corresponding case frame in the active voice, and find an alignment between cases by using the algorithm described in Section 5.3. In this example, an active case frame "突き落とす-4 (push down-4)" is identified as a corresponding case frame for the input passive case frame "突き落とされる-5 (be pushed down-5)" and *ga*, *ni*, and *kara*-cases in the passive case frame are aligned to *wo*, *ga*, and *kara*-cases in the active case frame, respectively.

5.2 Clues for Knowledge Acquisition

We exploit three clues for corresponding case frame identification and case alignment as follows:

1. Semantic similarity between the instances of the aligned cases: sim_{SEM}.
2. Case distribution similarity between the corresponding case frames: sim_{DIST}.
3. Preference of alternation patterns: f_{PP}.

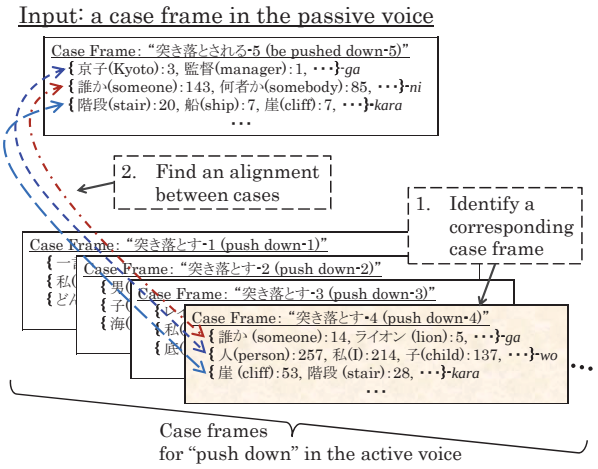


Figure 1: The overview of our task.

Semantic similarity The instances of the aligned cases should be similar. For example, the instances of the *ga*-case of the case frame "突き落とされる-5 (be pushed down-5)" and the *wo*-case of the case frame "突き落とす-4 (push down-4)," which are considered to be aligned and represent *patient*, are similar. Thus, we exploit semantic similarity sim_{SEM} between the instances of the corresponding cases.

We first define an asymmetric similarity measure between C_1 and C_2, each of which is a set of case slot instances, as follows:

$$\mathrm{sim}_a(C_1, C_2) = \frac{1}{|C_1|} \sum_{i_1 \in C_1} \max_{i_2 \in C_2} (\mathrm{sim}(i_1, i_2)),$$

where $\mathrm{sim}(i_1, i_2)$ is the similarity between instances. In this study, we apply a distributional similarity measure (Lin, 1998), which was computed from the Web corpus used to construct the case frames. We next define a symmetric similarity measure between C_1 and C_2 as an average of $\mathrm{sim}_a(C_1, C_2)$ and $\mathrm{sim}_a(C_2, C_1)$.

$$\mathrm{sim}_s(C_1, C_2) = \frac{1}{2}(\mathrm{sim}_a(C_1, C_2) + \mathrm{sim}_a(C_2, C_1)).$$

Then we define semantic similarity of a case alignment A between case frames CF_1 and CF_2.

$$\mathrm{sim}_{SEM}(A) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{sim}_s(C_{1,i}, C_{2,a(i)}),$$

where N denotes the number of case slots of CF_1, $C_{1,i}$ denotes a set of instances of the i-th case slot of CF_1, and $C_{2,a(i)}$ denotes the set of the aligned case instances of CF_2. A denotes the alignment $\{c_{1,1} \rightarrow c_{2,a(1)}, c_{1,2} \rightarrow c_{2,a(2)}, \ldots, c_{1,N} \rightarrow c_{2,a(N)}\}$ where $c_{n,i}$ denotes the case name that corresponds to $C_{n,i}$.

Case distribution similarity Although arguments are often omitted in Japanese, arguments that are usually mentioned explicitly in the passive voice will be also explicitly mentioned in the active voice. Hence, the frequency distribution of cases can be a clue for case alignment. In this study, we exploit the following cosine similarity of frequency distribution as case distribution similarity:

$$\mathrm{sim}_{DIST}(A) = \cos((|C_{1,1}|, \ldots, |C_{1,N}|), (|C_{2,a(1)}|, \ldots, |C_{2,a(N)}|)).$$

As an example, consider the alignment between a passive case[6] "選ばれる-1 (be selected-1)" and the corresponding active case frame "選ぶ-13 (select-13)" in Table 4. The alignment $A_1 = \{ga \rightarrow wo, ni \rightarrow ni, \mathrm{NIL} \rightarrow ga\}$ is considered to be correct. However, if we consider only the semantic similarity, an alignment $A_2 = \{ga \rightarrow ni, ni \rightarrow ga, wo \rightarrow wo\}$ is selected, because the alignment A_2 has the highest semantic similarity. On the other hand, the case distribution similarity

$$\mathrm{sim}_{DIST}(A_1) = \cos((17722, 122273, 0), (33338, 800, 382)) \approx 0.167$$

is much larger than

$$\mathrm{sim}_{DIST}(A_2) = \cos((17722, 122273, 96), (800, 382, 33338)) \approx 0.016.$$

Thus, the alignment A_1 would be selected by considering the case distribution similarity.

Preference of alternation patterns Some alternation patterns often appear, and others do not. For example, as Murata et al. (2006) reported, whereas 96.47% of *ga*-case is alternated with *wo*-case in passive-active transformation in Japanese, only 27.38% of *ni*-case is alternated with *ga*-case. Therefore, when we can use development data, we exploit a weighting factor $f_{PP}(A)$ that is determined on the development data and takes into account the preference of alternation patterns. We define $f_{PP}(A)$ as follows:

$$f_{PP}(A) = w(ga \rightarrow c_{ga\_to}) \times w(c_{to\_ga} \rightarrow ga), \quad \text{(i)}$$

where c_{ga_to} is the case in the active voice to which *ga*-case in the passive voice is aligned, c_{to_ga} is the case in the passive voice which is aligned to *ga*-case in the active voice, and $w(c_1 \rightarrow c_2)$ denotes the weight of the case alternation "$c_1 \rightarrow c_2$."

Case Frame: "選ばれる-1 (be selected-1)"
{ 選手 (player):1119, 作品 (work):983, $\cdots$ }-*ga*:17722
{ 代表 (representative):18295, $\cdots$ }-*ni*:122273
{ 作品 (work):5, 市長 (mayor):3, $\cdots$ }-*wo*:96
$\cdots$

Case Frame: "選ぶ-13 (select-13)"
{ 私 (I):14, 先生 (teacher):18, $\cdots$ }-*ga*:382
{ 優秀賞 (award):42, シングル (single):17, $\cdots$ }-*ni*:800
{ 曲 (tune):16666, 作品 (work):9967, $\cdots$ }-*wo*:33338
$\cdots$

Table 4: Case frames "選ばれる-1 (be selected-1)" and "選ぶ-13 (select-13)." The numbers following case names denote the total numbers of case slot instances.

5.3 Algorithm

Algorithm 1 presents our algorithm for identifying a corresponding case frame and finding an alignment between cases in pseudo-code. Our algorithm first makes all possible combinations of a case frame in the active voice (cf_{active}), a case in the active voice to which *ga*-case in the passive voice is aligned (c_{ga_to}), and a case in the passive voice which is aligned to *ga*-case in the active voice (c_{to_ga}) on the basis of the linguistic constraints, and then evaluates the score for the combinations $\{cf_{active}, c_{ga\_to}, c_{to\_ga}\}$ by the following equation:

$$score = \mathrm{sim}_{SEM}(A) \times \mathrm{sim}_{DIST}(A)^{\alpha} \times f_{PP}(A), \quad \text{(ii)}$$

where α is a parameter that controls the impact of the case distribution similarity.[7] When we can use

[6] This case frame should not have *wo*-case. However, since we constructed case frames automatically, some case frames have improper cases.

[7] Since $f_{PP}(A)$ is defined with a set of weights of case alternation patterns, $f_{PP}(A)$ contains these weights implicitly, and thus there is only a single explicit weight in equation (ii).

Algorithm 1: Identifying a corresponding case frame and finding an alignment between cases.

Input: a case frame in the passive voice: $cf_{passive}$, and a set of case frames in the active voice: CFS_{active}
Output: a case frame and an alignment between cases: A

1: $max\_score = 0$, $A = ()$
2: **for each** $cf_{active} \in CFS_{active}$
3: **for each** $c_{ga\_to} \in$ {*wo*, *ni*, *no*, NIL}
4: **for each** $c_{to\_ga} \in$ {*ni*, *niyotte*, *kara*, *de*, NIL}
5: **if** (!occur(c_{ga_to}, c_{to_ga})) **then continue**
6: $A' = (cf_{active}, c_{ga\_to}, c_{to\_ga})$
7: $score = \text{sim}_{SEM}(A') \times \text{sim}_{DIST}(A')^{\alpha} \times f_{PP}(A')$
8: **if** ($score > max\_score$) **then**
9: $(max\_score, A) = (score, A')$
10: **end for**
11: **end for**
12: **end for**

development data, we tune α on the development data; otherwise we set $\alpha = 1$. Since some combinations of c_{ga_to} and c_{to_ga} never occur, our algorithm filters them out in line 5 of the algorithm. After checking all combinations, the combination with the highest score is output.

6 Evaluation of the Acquired Knowledge

We applied our algorithm to the case frames that are automatically constructed from a corpus consisting of about 6.9 billion Japanese sentences from the Web. Of course, these case frames contain improper ones, that is, several frames mix several meanings or usages of the predicates. Thus, it is difficult to evaluate the acquired knowledge itself. Instead, we evaluate the usefulness of the acquired knowledge on a case alternation task between the passive and active voices.

6.1 Setting and Algorithm for Case Alternation

We basically used the same data as Murata et al. (2006). As mentioned in Section 2, they extracted 3,576 case particles in passive sentences from the Kyoto University Text Corpus, and tagged their cases in the active voice. Since they treated possessor passive as a kind of indirect passive, they did not adopt the case alternation between *ga* and *no*. In addition, their data included some annotation errors. We thus modified 21 annotations,[8] five of which were changed to the case alternation between *ga* and *no*. Note that there were some cases where multiple possible case particles were tagged to one instance. We adopted evaluation metrics called "Eval. B" by Murata et al., that is, we judged the output to be correct when the output was included in possible answers. We performed experiments on the following three types of data settings.

1. Experiments without either development or training data.

2. Experiments with development data.

3. Experiments with training data.

Experiments without either development or training data In the first setting, we aligned the input passive case frame to one of the active case frames of the same predicate only by using sim_{SEM} and sim_{DIST} with the parameter $\alpha = 1$. Therefore, this setting is fully unsupervised. In this setting, the input surface cases are alternated as follows:

1. If a passive sentence is input, perform syntactic and surface case structure analysis by using Kawahara and Kurohashi (2006)'s model.[9] Their model identified a proper case frame for each predicate, and assigned arguments in the input sentence to case slots of the case frame.

2. By using the acquired knowledge for case alternation, alternate input surface cases with cases in the active voice.

We call this model Model 1. For example, if Example (10) is input, the *ga*-case argument is assigned to the *ga*-case of the case frame "突き落とされる-5 (be pushed down-5)." Since this case is aligned to the *wo*-case of the case frame "突き落とす-4 (push down-4)" as shown in Figure 1, this *ga*-case is alternated with *wo*-case.

(10) 女が　突き落とされた.
woman-*ga* was pushed down
(A woman was pushed down.)

[8] The modified version of the data is publicly available at http://alaginrc.nict.go.jp/case/src/kaku1.1.tar.gz.

[9] KNP: http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP

Algorithm 2: Pseudo-code of the hill-climbing algorithm for tuning the parameter vector $\mathbf{x}$.

$\mathbf{x} = (1.0, 1.0, \ldots, 1.0)$
$acc = f_{accuracy}(\mathbf{x}), pre\_acc = 0$
while $acc > pre\_acc$
  $pre\_acc = acc$
  for $i \in \{0, \ldots, |\mathbf{x}| - 1\}$
    $acc_+ = f_{accuracy}(x_0, \ldots, x_i + 0.1, \ldots, x_{|\mathbf{x}|-1})$
    $acc_- = f_{accuracy}(x_0, \ldots, x_i - 0.1, \ldots, x_{|\mathbf{x}|-1})$
    if $acc_+ > acc$ **and** $acc_+ > acc_-$ **then** $x_i = x_i + 0.1$
    else if $acc_- > acc$ **then** $x_i = x_i - 0.1$
    $acc = f_{accuracy}(\mathbf{x})$
  end for
end while

	Model			Accuracy
	sim_{SEM}	sim_{DIST}	Parameter tuning	
Model 1_S	✓			0.902 (3,224/3,576)
Model 1_D		✓		0.857 (3,063/3,576)
Model 1	✓	✓		0.906 (3,239/3,576)
Model 2_S	✓		✓	0.928 (3,320/3,576)
Model 2_D		✓	✓	0.927 (3,314/3,576)
Model 2	✓	✓	✓	0.938 (3,353/3,576)
Baseline				0.883 (3,159/3,576)

Table 5: Experimental results of case alternation without training data.

Experiments with development data In the second setting, we aligned the input passive case frame to one of the active case frames of the same predicate by using sim_{SEM}, sim_{DIST}, and f_{PP} with α tuned on the development data. In advance, we divided the tagged data into two parts just as Murata et al. (2006) did, both of which contained 1,788 case particles, and performed 2-fold cross-validation. We used one part for development and the other for testing, and vice versa.

We tuned $w(ga \rightarrow c_{ga\_to})$, $w(c_{to\_ga} \rightarrow ga)$ in Equation (i), and α in Equation (ii) by a simple hill-climbing strategy. Since the candidate cases for c_{ga_to} are *ni*, *niyotte*, *kara*, *de*, and NIL, and the candidate cases for c_{to_ga} are *wo*, *ni*, *no*, and NIL, we defined parameter vector $\mathbf{x}$ as follows:

$$\mathbf{x} = (w(ga \rightarrow ni), w(ga \rightarrow niyotte), w(ga \rightarrow kara), w(ga \rightarrow de), w(ga \rightarrow \text{NIL}), w(wo \rightarrow ga), w(ni \rightarrow ga), w(wo \rightarrow no), w(\text{NIL} \rightarrow ga), \alpha).$$

Algorithm 2 shows the hill-climbing algorithm for tuning the parameter vector $\mathbf{x}$, where $f_{accuracy}(\mathbf{x})$ is a function that returns the case alternation accuracy on the development data with parameter $\mathbf{x}$. This algorithm varies one parameter at a time with a step-size of 0.1 until there is no accuracy improvement in the development data. After acquiring knowledge for case alternation with the tuned parameter, we applied the same method for case alternation as the first setting. We call this model Model 2.

Experiments with training data In the third setting, we also performed 2-fold cross validation, that is, we used one part of the divided tagged corpus for training and the other for testing, and vice versa. Although we basically applied Murata et al. (2006)'s method, which is based on SVMs, we added the output of Model 2 as a new feature.

Specifically, we first tuned the parameter vector $\mathbf{x}$ on the training data and acquired the knowledge for case alternation with the tuned parameter. By using the acquired knowledge, we alternated the input cases in both the training and test data and obtained the resulting case of Model 2. Note that, we did not use any annotations for the test data in this process. We then trained the SVMs on the training data and applied them to the test data using the resulting case as a new feature. We call this model Model 3.

6.2 Results and Discussion

Table 5 shows the results of the experiments without training data. Baseline is a system that outputs the most frequently alternated cases in the development data, which was also used by Murata et al. (2006). The baseline score was higher than that reported by Murata et al. because we modified 21 annotations. We also performed experiments without using case distribution similarity or semantic similarity. We call these models in the first setting Model 1_S and Model 1_D, and these models in the second setting Model 2_S and Model 2_D, respectively.

Although Models 1_S, 1_D, and 1 were fully unsupervised models, Models 1_S and 1 significantly[10] outperformed the baseline model (p-values of McNemar (1947)'s test were smaller than 0.00001). On the other hand, the difference between Models 1_S

[10] In this paper, we call a difference significant if the p-value of McNemar (1947)'s test is less than 0.01.

Model	Accuracy
(Murata et al., 2006)	0.944 (3,376/3,576)
Model 3	0.956 (3,417/3,576)

Table 6: Comparison between Murata et al. (2006)'s method and our method with training data.

and 1 is not statistically significant, and thus the effect of the case distribution similarity was not confirmed by these experiments.

Models 2_S, 2_D, and 2 were models with parameter tuning. Parameter tuning significantly improved the performance. In addition, the difference between Models 2_S and 2 and the difference between Models 2_D and 2 were both significant (p-values of McNemar's test were 0.00032 and 0.00039, respectively), and thus we confirmed the usefulness of the two similarity measures. The parameter α that controls the impact of the case distribution similarity was tuned to 0.3, which means semantic similarity between the instances of the aligned cases is more important than case distribution similarity for this task.

Table 6 compares Murata et al.'s method and our method with training data. We used Murata et al.'s method without feature selection because it achieved the highest performance on this setting. Their method's score was higher than that they reported, again due to the corpus modification. The difference between their method and our method was significant (p-value of McNemar's test was 0.00011), and we confirmed the usefulness of the acquired knowledge for case alternation.

Table 7 shows an example of case alternation between the passive and active voices. When the passive sentence was input, the argument "松樹さんが(Mr. Matsuki-*ga*)" was first assigned to *ga*-case of the case frame "殴られる-2 (be hit-2)." Since this case was aligned to *no*-case of the case frame "殴る-2 (hit-2)," the input *ga*-case was alternated with *no*-case. On the other hand, the cases of the other arguments "バットで (bat-*de*)" and "頭を (head-*wo*)" were output as they were in the passive sentence.

We now list three error causes observed in our experiments of the case alternation task:

1) The passive voice in Japanese is expressed by using the auxiliary verbs "れる (*reru*)" and "られる (*rareru*)." However, these auxiliary verbs can represent several other meanings, such as honorific and possibility. Since Kawahara and Kurohashi (2002)'s method does not distinguish between these meanings, our case frames sometimes contain improper cases such as *wo*-case in case frame "選ばれる-1 (be selected-1)" in Table 4.

Input Text:
· · · 松樹さんが　金属バットで　頭を　殴られ、· · ·
Mr. Matsuki-*ga* metal bat-*de* head-*wo* was hit
(. . . Mr. Matsuki was hit on the head with a metal bat . . .)

Identified passive case frame:
Case Frame: "殴られる-2 (be hit-2)"
{ 何者か (someone):2, 部員 (member):1, · · · }-*niyotte*
{ 女性 (woman):5, 女児 (girl)):4, · · · }-***ga***
{ 頭 (head):3944, 顔 (face):1186, · · · }-*wo*
{ 鈍器 (blunt weapon):84, バット (bat):45, · · · }-*de*
· · ·

Corresponding active case frame and case alignment:
Case alignment: {*niyotte*→*ga*, ***ga*** **→*no***, *wo*→*wo*, *de*→*de*}
Case Frame: "殴る-2 (hit-2)"
{ 男 (man):51, 拳 (fist):30, 誰か (someone):23, · · · }-*ga*
{ 自分 (myself):360, 私 (I):223, · · · }-***no***
{ 頭 (head):5424, 顔 (face):3215, · · · }-*wo*
{ 拳 (fist):316, 平手 (palm):157, 拳 (fist):43, · · · }-*de*
· · ·

Table 7: An example of case alternation. The input *ga*-case was alternated with *no*-case.

2) In some passive sentences, there are two surface *ni*-cases as in Example (11). However, our method does not assume such sentences, and thus cannot deal with them properly.

(11) 男に (man-*ni*) オフィスに (office-*ni*) 派遣された. (was sent)
(ϕ was sent to the office by a man.)

3) *Agent* of a predicate can be represented by using several types of case particles in the passive voice. For example, "会社 (company)" in Example (12) is the *agent* of "雇用した (employed)," which can be represented by either of *ni*, *niyotte*, and *kara* in the passive voice. Since Kawahara and Kurohashi (2002)'s method can not recognize the exchangeablity of case particles, some case frames contain several cases of the same semantic role. However, since our method enforces a one-to-one alignments, only one of these cases is properly aligned to the corresponding case in the active voice.

(12) 会社 が　　男を　　雇用した.
company-*ga* man-*wo* employed
(The company employed a man.)

6.3 Application to Alternation between the Causative and Active Voices

To confirm the applicability of our framework to other types of alternation than the active-passive alternation, we applied our framework to case alternation between the causative and active voices. The causative voice in Japanese is a grammatical voice and is expressed by using the auxiliary verbs "せる (*seru*)" and "させる (*saseru*)." We basically used the same algorithm as Algorithm 1 for acquiring the knowledge for case alternation, but used different constraints on case alternation patterns because possible case alternation patterns are different from those of active-passive alternation. Specifically, we replaced the third and fourth lines of Algorithm 1 with "for each $c_{to\_ga} \in \{\text{NIL}, ni\}$" and "for each $c_{ga\_to} \in \{wo, ni\}$," respectively, based on linguistic analysis of active-causative alternation in Japanese.

We used a part of the data created by Murata and Isahara (2003) to evaluate the usefulness of the acquired knowledge. Their data consists of 4,671 case particles in passive or causative sentences from the Kyoto University Text Corpus with their cases in the active voice. We first extracted 524 case particles that were extracted from causative sentences. Since the annotation quality was not very high, we manually checked all tags and modified inappropriate ones. We then performed 2-fold cross validation experiments. Table 8 shows experimental results. Baseline is a system that outputs the most frequently alternated cases in the training data. The difference between Murata et al. (2006)'s model[11] and our method was significant (p-value of McNemar's test was 0.0019), and we confirmed the applicability of our framework to active-causative alternation.

Model	Accuracy
Baseline	0.781 (409/524)
Murata et al. (2006)'s model	0.836 (438/524)
Our method with training data	0.872 (457/524)

Table 8: Experimental results of case alternation between the causative and active voices.

7 Conclusions and Future Directions

We have presented a method for automatically acquiring knowledge for case alternation between the passive and active voices in Japanese. Our method aligned an input case frame in the passive voice to a corresponding case frame in the active voice and found an alignment between their cases. We then applied the acquired knowledge to a case alternation task and proved its usefulness.

The knowledge we have to manually construct is only the knowledge of linguistic constraints on case alternation patterns. The other types of knowledge are automatically acquired from a large raw corpus. Thus, although this paper focused on the active-passive alternation in Japanese, our framework is applicable to the other types of case alternation and to other languages, especially similar languages such as Korean. We plan to apply our framework to other types of case alternation such as case alternation between intransitive and transitive verbs.

[11] In this experiment, we used the same features as those used by Murata and Isahara (2003).

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 23800025 and 25730131.

References

Timothy Baldwin and Hozumi Tanaka. 2000. Verb alternations and Japanese – how, what and where? In *Proc. of PACLIC 14*, pages 3–14.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistic*, 36(4):673–721.

Charles J. Fillmore, Christopher R. Johnson, and Miriam. R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proc. of ACL'07 Workshop: Linguistic Annotation Workshop*, pages 132–139.

IPA. 1987. Japanese verbs : A guide to the IPA lexicon of basic Japanese verbs.

Eric Joanis, Suzanne Stevenson, and David James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

Daisuke Kawahara and Sadao Kurohashi. 2002. Fertilization of case frame dictionary for robust Japanese case analysis. In *Proc. of COLING'02*, pages 425–431.

Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proc. of HLT-NAACL'06*, pages 176–183.

Daisuke Kawahara, Ryohei Sasano, and Sadao Kurohashi. 2004. Toward text understanding: Integrating relevance-tagged corpora and automatically constructed case frames. In *Proc. of LREC'04*, pages 1833–1836.

Keiko Kondo, Satoshi Sato, and Manabu Okumura. 2001. Paraphrasing by case alternation (in Japanese). *Journal of Information Processing Society of Japan*, 42(3):465–477.

Anna Korhonen, Yuval Krymolowski, and Ted Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proc.of LREC'06*, pages 3000–3006.

Mirella Lapata and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–73.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Jianguo Li and Chris Brew. 2008. Which are the best features for automatic verb classification. In *Proc. of ACL-HLT'08*, pages 434–442.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of ACL-COLING'98*, pages 768–774.

Diana McCarthy. 2000. Using semantic preferences to identify verbal participation in role switching alternations. In *Proc. of NAACL'00*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157.

Masaki Murata and Hitoshi Isahara. 2003. Conversion of Japanese passive/causative sentences into active sentences using machine learning. In *Proc. of CICLing'03*, pages 115–125.

Masaki Murata, Toshiyuki Kanamaru, Tamotsu Shirado, and Hitoshi Isahara. 2006. Machine-learning-based transformation of passive Japanese sentences into active by separating training data into each input particle. In *Proc. of COLING-ACL'06*, pages 587–594.

Katashi Nagao and Koiti Hasida. 1998. Automatic text summarization based on the global document annotation. In *Proc. of ACL'98*, pages 917–921.

NLRI. 1993. *Bunrui Goi Hyo (in Japanese)*. Shuuei Publishing.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1):71–105.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proc. of IJCNLP'11*, pages 758–766.

Sabine Schulte im Walde, Christian Hying, Christian Scheible, and Helmut Schmid. 2008. Combining EM training and the MDL principle for an automatic verb classification incorporating selectional preferences. In *Proc. of ACL-HLT'08*, pages 496–504.

Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proc. of EMNLP'09*, pages 638–647.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proc. of EMNLP'08*, pages 523–532.

Daphne Theijssen, Lou Boves, Hans van Halteren, and Nelleke Oostdijk. 2012. Evaluating automatic annotation: automatically detecting and enriching instances of the dative alternation. *Language Resources and Evaluation*, 46(4):565–600.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with markov logic. In *Proc. of IJCNLP'11*, pages 1125–1133.

Exploiting Multiple Sources for Open-domain Hypernym Discovery

Ruiji Fu, Bing Qin, Ting Liu*
Research Center for Social Computing and Information Retrieval
School of Computer Science and Technology
Harbin Institute of Technology, China
{rjfu, bqin, tliu}@ir.hit.edu.cn

Abstract

Hypernym discovery aims to extract such noun pairs that one noun is a hypernym of the other. Most previous methods are based on lexical patterns but perform badly on open-domain data. Other work extracts hypernym relations from encyclopedias but has limited coverage. This paper proposes a simple yet effective distant supervision framework for Chinese open-domain hypernym discovery. Given an entity name, we try to discover its hypernyms by leveraging knowledge from multiple sources, i.e., search engine results, encyclopedias, and morphology of the entity name. First, we extract candidate hypernyms from the above sources. Then, we apply a statistical ranking model to select correct hypernyms. A set of novel features is proposed for the ranking model. We also present a heuristic strategy to build a large-scale noisy training data for the model without human annotation. Experimental results demonstrate that our approach outperforms the state-of-the-art methods on a manually labeled test dataset.

1 Introduction

Hypernym discovery is a task to extract such noun pairs that one noun is a hypernym of the other (Snow et al., 2005). A noun H is a hypernym of another noun E if E is an instance or subclass of H. In other word, H is a semantic class of E. For instance, "actor" is a hypernym of "Mel Gibson"; "dog" is a hypernym of "Caucasian sheepdog"; "medicine" is a hypernym of "Aspirin". Hypernym discovery is an important subtask of semantic relation extraction and has many applications in ontology construction (Suchanek et al., 2008), machine reading (Etzioni et al., 2006), question answering (McNamee et al., 2008), and so on.

Some manually constructed thesauri such as WordNet can also provide some semantic relations such as hypernyms. However, these thesauri are limited in its scope and domain, and manual construction is knowledge-intensive and time-consuming. Therefore, many researchers try to automatically extract semantic relations or to construct taxonomies.

Most previous methods on automatic hypernym discovery are based on lexical patterns and suffer from the problem that such patterns can only cover a small part of complex linguistic circumstances (Hearst, 1992; Turney et al., 2003; Zhang et al., 2011). Other work tries to extract hypernym relations from large-scale encyclopedias like Wikipedia and achieves high precision (Suchanek et al., 2008; Hoffart et al., 2012). However, the coverage is limited since there exist many infrequent and new entities that are missing in encyclopedias (Lin et al., 2012). We made similar observation that more than a half of entities in our data set have no entries in the encyclopedias.

This paper proposes a simple yet effective distant supervision framework for Chinese open-domain hypernym discovery. Given an entity name, our goal is to discover its hypernyms by leveraging knowledge from multiple sources. Considering the case where a person wants to know the meaning of an unknown entity, he/she may search it in a search engine and then finds out the answer after going through the search results. Furthermore, if he/she finds an entry about the entity in an authentic web site, such as Wikipedia, the information will help him/her under-

*Email correspondence.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1224–1234,
Seattle, Washington, USA, 18-21 October 2013.

stand the entity. Also, the morphology of the entity name can provide supplementary information. In this paper, we imitate the process. The evidences from the above sources are integrated in our hypernym discovery model.

Our approach is composed of two major steps: hypernym candidate extraction and ranking. In the first step, we collect hypernym candidates from multiple sources. Given an entity name, we search it in a search engine and extract high-frequency nouns as its main candidate hypernyms from the search results. We also collect the category tags for the entity from two Chinese encyclopedias and the head word of the entity as the candidates.

In the second step, we identify correct hypernyms from the candidates. We view this task as a ranking problem and propose a set of effective features to build a statistical ranking model. For the parameter learning of the model, we also present a heuristic strategy to build a large-scale noisy training data without human annotation.

Our contributions are as follows:

- We are the first to discover hypernym for Chinese open-domain entities by exploiting multiple sources. The evidences from different sources can authenticate and complement each other to improve both precision and recall.
- We manually annotate a dataset containing 1,879 Chinese entities and their hypernyms, which will be made publicly available. To the best of our knowledge, this is the first dataset for Chinese hypernyms.
- We propose a set of novel and effective features for hypernym ranking. Experimental results show that our method achieves the best performance.

Furthermore, our approach can be easily ported from Chinese to English and other languages, except that a few language dependent features need to be changed.

The remainder of the paper is organized as follows: Section 2 discusses the related work. Section 3 introduces our method in detail. Section 4 describes the experimental setup. Section 5 shows the experimental results. Conclusion and future work are presented in Section 6.

2 Related Work

Previous methods for hypernym discovery can be summarized into two major categories, i.e., pattern-based methods and encyclopedia-based methods.

Pattern-based methods make use of manually or automatically constructed patterns to mine hypernym relations from text corpora. The pioneer work by Hearst (1992) finds that linking two noun phrases (NPs) via certain lexical constructions often implies hypernym relations. For example, NP_1 is a hypernym of NP_2 in the lexical pattern "such NP_1 as NP_2". Similarly, succeeding researchers follow her work and use handcrafted patterns to extract hypernym pairs from corpora (Caraballo, 1999; Scott and Dominic, 2003; Ciaramita and Johnson, 2003; Turney et al., 2003; Pasca, 2004; Etzioni et al., 2005; Ritter et al., 2009; Zhang et al., 2011).

Evans (2004) considers the web data as a large corpus and uses search engines to identify hypernyms based on lexical patterns. Given an arbitrary document, he takes each capitalized word sequence as an entity and aims to find its potential hypernyms through pattern-based web searching. Suppose `X` is a capitalized word sequence. Some pattern queries like "such as `X`" are threw into the search engine. Then, in the retrieved documents, the nouns that immediately precede the pattern are recognized as the hypernyms of `X`. This work is most related to ours. However, the patterns used in his work are too strict to cover many low-frequency entities, and our experiments show the weakness of the method.

Snow et al. (2005) for the first time propose to automatically extract large numbers of lexico-syntactic patterns and then detect hypernym relations from a large newswire corpus. First, they use some known hypernym-hyponym pairs from WordNet as seeds and collect many patterns from a syntactically parsed corpus in a bootstrapping way. Then, they consider all noun pairs in the same sentence as potential hypernym-hyponym pairs and use a statistical classifier to recognize the correct ones. All patterns corresponding to the noun pairs in the corpus are fed into the classifier as features. Their method relies on accurate syntactic parsers and it is difficult to guarantee the quality of the automatically extracted patterns. Our experiments show that their method is inferior to ours.

Encyclopedia-based methods extract hypernym relations from encyclopedias like Wikipedia (Suchanek et al., 2008; Hoffart et al., 2012). The user-labeled information in encyclopedias, such as category tags in Wikipedia, is often used to derive hypernym relations.

In the construction of the famous ontology YAGO, Suchanek et al. (2008) consider the title of each Wikipedia page as an entity and the corresponding category tags as its potential hypernyms. They apply a shallow semantic parser and some rules to distinguish the correct hypernyms. Heuristically, they find that if the head of the category tag is a plural word, the tag is most likely to be a correct hypernym. However, this method cannot be used in Chinese because of the lack of plurality information.

The method of Suchanek et al. (2008) cannot handle the case when the entity is absent in Wikipedia. To solve this problem, Lin et al. (2012) connect the absent entities with the entities present in Wikipedia sharing common contexts. They utilize the Freebase semantic types to label the present entities and then propagate the types to the absent entities. The Freebase contains most of entities in Wikipedia and assigns them semantic types defined in advance. But there are no such resources in Chinese.

Compared with previous work, our approach tries to identify hypernyms from multiple sources. The evidences from different sources can authenticate and complement each other to improve both precision and recall. Our experimental results show the effectiveness of our method.

3 Method

Our method is composed of two steps. First, we collect candidate hypernyms from multiple sources for a given entity. Then, a statistical model is built for hypernym ranking based on a set of effective features. Besides, we also present a heuristic strategy to build a large-scale training data.

3.1 Candidate Hypernym Collection from Multiple Sources

In this work, we collect potential hypernyms from four sources, i.e., search engine results, two encyclopedias, and morphology of the entity name.

We count the co-occurrence frequency between the target entities and other words in the returned snippets and titles, and select top N nouns (or noun phrases) as the main candidates. As the experiments show, this method can find at least one hypernym for 86.91% entities when N equals 10 (see Section 5.1). This roughly explains why people often can infer semantic meaning of unknown entities after going through several search results.

Furthermore, the user-generated encyclopedia category tags are important clues if the entity exists in a encyclopedia. Thus we add these tags into the candidates. In this work, we consider two Chinese encyclopedias, Baidubaike and Hudongbaike[1], as hypernym sources.

In addition, the head words of entities are also their hypernyms sometimes. For example, the head word of "皇帝企鹅 (Emperor Penguin)" indicates that it's a kind of "企鹅 (penguins)". Thus we put head words into the hypernym candidates. In Chinese, head words are often laid after their modifiers. Therefore, we try to segment a given entity. If it can be segmented and the last word is a noun, we take the last word as the head word. In our data set, the head words of 41.35% entities are real hypernyms (see Section 5.1).

We combine all of these hypernym candidates together as the input of the second stage. The final coverage rate reaches 93.24%.

3.2 Hypernym Ranking

After getting the candidate hypernyms, we then adopt a ranking model to determine the correct hypernym. In this section, we propose several effective features for the model. The model needs training data for learning how to rank the data in addition to parameter setting. Considering that manually annotating a large-scale hypernym dataset is costly and time-consuming, we present a heuristic strategy to collect training data. We compare three hypernym ranking models on this data set, including Support Vector Machine (SVM) with a linear kernel, SVM with a radial basis function (RBF) kernel and Logistic Regression (LR).

[1]Baidubaike (`http://baike.baidu.com`) and Hudongbaike (`http://www.baike.com`) are two largest Chinese encyclopedias containing more than 6.26 million and 7.87 million entries respectively, while Chinese Wikipedia contains about 0.72 million entries until September, 2013.

Feature	Comment	Value Range
Prior	the prior probability of a candidate being a potential hypernym	[0, 1]
Is_Tag	whether a candidate is a category tag in the encyclopedia page of the entity if it exists	0 or 1
Is_Head	whether a candidate is the head word of the entity	0 or 1
In_Titles	some binary features based on the frequency of occurrence of a candidate in the document titles in the search results	0 or 1
Synonyms	the ratio of the synonyms of the candidate in the candidate list of the entity	[0, 1]
Radicals	the ratio of the radicals of characters in a candidate matched with the last character of the entity	[0, 1]
Source_Num	the number of sources where the candidate is extracted	1, 2, 3, or 4
Lexicon	the hypernym candidate itself and its head word	0 or 1

Table 1: The features for ranking

3.2.1 Features for Ranking

The features for hypernym ranking are shown in Table 1. We illustrate them in detail in the following.

Hypernym Prior: Intuitively, different words have different probabilities as hypernyms of some other words. Some are more probable as hypernyms, such as *animal*, *plant* and *fruit*. Some other words such as *sun*, *nature* and *alias*, are not usually used as hypernyms. Thus we use a prior probability to express this phenomenon. The assumption is that if the more frequent that a noun appears as category tags, the more likely it is a hypernym. We extract category tags from 2.4 million pages in Baidubaike, and compute the prior probabilities $prior(w)$ for a word w being a potential hypernym using Equation 1. $count_{CT}(w)$ denotes the times a word appeared as a category tag in the encyclopedia pages.

$$prior(w) = \frac{count_{CT}(w)}{\sum_{w'} count_{CT}(w')} \quad (1)$$

In Titles: When we enter a query into a search engine, the engine returns a search result list, which contains document titles and their snippet text. The distributions of hypernyms and non-hypernyms in titles are compared with that in snippets respectively in our training data. We discover that the average frequency of occurrence of hypernyms in titles is 15.60 while this number of non-hypernyms is only 5.18, while the difference in snippets is very small (Table 2). Thus the frequency of candidates in titles can be used as features. In this work the frequency is divided into three cases: greater than 15.60, less than 5.18, and between 5.18 and 15.60. Three binary features are used to represent these cases.

	Avg. Frequency in titles	Avg. Frequency in snippets
Hypernym	15.60	33.69
Non-Hypernym	5.18	30.61

Table 2: Distributions of candidate hypernyms in titles and snippets

Synonyms: If there exist synonyms of a candidate hypernym in the candidate list, the candidate is probably correct answer. For example, when "药品 (medicine)" and "药物 (medicine)" both appear in the candidate list of an entity, the entity is probably a kind of medicine. We get synonyms of a candidate from a Chinese semantic thesaurus – Tongyi Cilin (Extended) (CilinE for short)[2] and compute the score as a feature using Equation 2.

$$ratio_{syn}(h, l_e) = \frac{count_{syn}(h, l_e)}{len(l_e)} \quad (2)$$

Given a hypernym candidate h of an entity e and the list of all candidates l_e, we compute the ratio of the synonyms of h in l_e. $count_{syn}(h, l_e)$ denotes the count of the synonyms of h in l_e. $len(l_e)$ is the total count of candidates.

[2]CilinE contains synonym and hypernym relations among 77 thousand words, which is manually organized as a hierarchy of five levels.

Radicals: Chinese characters are a form of ideogram. By far, the bulk of Chinese characters were created by linking together a character with a related meaning and another character to indicate its pronunciation. The character with a related meaning is called radical. Sometimes, it is a important clue to indicate the semantic class of the whole character. For example, the radical "虫" means insects, so it hints "蜻蜓 (dragonfly)" is a kind of insects. Similarly "疒" hints "淋巴癌 (lymphoma)" is a kind of diseases. Thus we use radicals as a feature the value of which is computed by using Equation 3.

$$radical(e,h) = \frac{count_{RM}(e,h)}{len(h)} \quad (3)$$

Here $radical(e,h)$ denotes the ratio of characters radical-matched with the last character of the entity e in the hypernym h. $count_{RM}(e,h)$ denotes the count of the radical-matched characters in h. $len(h)$ denotes the total count of the characters in h.

3.2.2 Training Data Collection

Now training data is imperative to learn the weights of the features in Section 3.2.1. Hence, we propose a heuristic strategy to collect training data from encyclopedias.

Firstly, we extract a number of open-domain entities from encyclopedias randomly. Then their hypernym candidates are collected by using the method proposed in Section 3.1. We select positive training instances following two principles:

- Principle 1: Among the four sources used for candidate collection, the more sources from which the hypernym candidate is extracted, the more likely it is a correct one.
- Principle 2: The higher the prior of the candidate being a hypernym is, the more likely it is a correct one.

We select the best candidates following Principle 1 and then select the best one in them as a positive instance following Principle 2. And we select a candidate as a negative training instance when it is from only one source and its prior is the lowest. If there are synonyms of training instances in the candidates list, the synonyms are also extended into the training set.

Domain	# of entities	
	Dev.	Test
Biology	72	351
Health Care	61	291
Food	75	303
Movie	51	204
Industry	56	224
Others	35	136
Total	350	1529

Table 3: The evaluation data

In this way, we collect training data automatically, which are used to learn the feature weights of the ranking models.

4 Experimental Setup

In this work, we use Baidu[3] search engine, the most popular search engine for Chinese, and get the top 100 search results for each entity. The Chinese segmentation, POS tagging and dependency parsing is provided by an open-source Chinese language processing platform LTP[4] (Che et al., 2010).

4.1 Experimental Data

In our experiments, we prepare open-domain entities from dictionaries in wide domains, which are published by a Chinese input method editor software Sogou Pinyin[5]. The domains include biology, health care, food, movie, industry, and so on. We sample 1,879 entities from these domain dictionaries and randomly split them into 1/5 for development and 4/5 for test (Table 3). We find that only 865 (46.04%) entities exist in Baidubaike or Hudongbaike. Then we extract candidate hypernyms for the entities and ask two annotators to judge each hypernym relation pair true or false manually. A pair `(E, H)` is annotated as true if the annotators judge "`E` is a (or a kind of) `H`" is true. Finally, we get 12.53 candidate hypernyms for each entity on average in which about 2.09 hypernyms are correct. 4,330 hypernym relation pairs are judged by both the annotators. We measure the agreement of the judges using the Kappa coefficient (Siegel and Castellan Jr, 1988). The

[3] http://www.baidu.com
[4] http://ir.hit.edu.cn/demo/ltp/
[5] http://pinyin.sogou.com/dict/

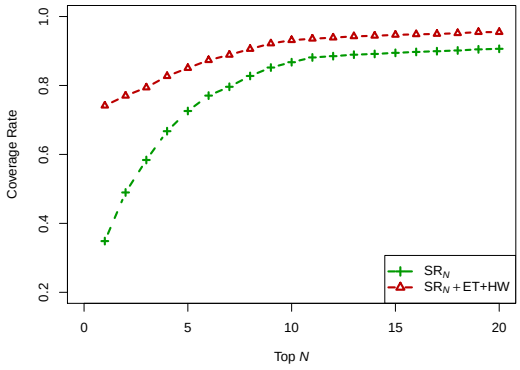


Figure 1: Effect of candidate hypernym coverage rate while varying N

Kappa value is 0.79.

Our training data, containing 11,481 positive instances and 18,378 negative ones, is extracted from Baidubaike and Hudongbaike using the heuristic strategy proposed in Section 3.2.2.

4.2 Experimental Metrics

The evaluation metrics for our task include:

Coverage Rate: We evaluate coverage rate of the candidate hypernyms. Coverage rate is the number of entities for which at least one correct hypernym is found divided by the total number of all entities.

Precision@1: Our method returns a ranked list of hypernyms for each entity. We evaluate precision of top-1 hypernyms (the most probable ones) in the ranked lists, which is the number of correct top-1 hypernyms divided by the number of all entities.

R-precision: It is equivalent to Precision@R where R is the total number of candidates labeled as true hypernyms of an entity.

Precision, Recall, and F-score: Besides, we can convert our ranking models to classification models by setting thresholds. Varying the thresholds, we can get different precisions, recalls, and F-scores.

5 Results and Analysis

5.1 The Coverage of Candidate Hypernyms

In this section, we evaluate the coverage rate of the candidate hypernyms. We check the candidate hypernyms of the whole 1,879 entities in the development and test sets and see how many entities we can collect at least one correct hypernym for.

Source	Coverage Rate	Avg. #
SR_{10}	0.8691	9.44*
ET	0.3938	3.07
HW	0.4135	0.87†
SR_{10} + ET	0.8909	12.02
SR_{10} + HW	0.9117	9.75
ET + HW	0.7073	3.92
SR_{10} + ET + HW	0.9324	12.53

Table 4: Coverage evaluation of the candidate hypernym extraction

There are four different sources to collect candidates as described in Section 3.1, which can be divided into three kinds: search results (SR for short), encyclopedia tags (ET) and head words (HW). For SR, we select top N frequent nouns (SR_N) in the search results of an entity as its hypernym candidates. The effect of coverage rate while varying N is shown in Figure 1. As we can see from the figure, the coverage rate is improved significantly by increasing N until N reaches 10. After that, the improvement becomes slight. When the candidates from all sources are merged, the coverage rate is further improved.

Thus we set N as 10 in the remaining experiments. The detail evaluation is shown in Table 4. We can see that top 10 frequent nouns in the search results contain at least one correct hypernym for 86.91% entities in our data set. This coincides with the intuition that people usually can infer the semantic classes of unknown entities by searching them in web search engines.

The coverage rate of ET merely reaches 39.38%. We find the reason is that more than half of the entities have no encyclopedia pages. The average number of candidate hypernyms from ET is 3.07. Note that the number is calculated among all the entities. We also calculate the average number only for the present entities in encyclopedias. The number reaches 6.68. The reason is that for many present entities, the category tags include not only hypernyms

*For some of entities are rare, there may be less than 10 nouns in the search results. So the average count of candidates is less than 10.

†Not all of the entities can be segmented. We cannot get the head words of the ones that cannot be segmented.

Method	Present Entities		Absent Entities		All Entities	
	P@1	R-Prec	P@1	R-Prec	P@1	R-Prec
$M_{Pattern}$	0.5542	0.4937	0.4306	0.3638	0.5229	0.4608
M_{Snow}	0.3199	0.2592	0.2827	0.2610	0.3092	0.2597
M_{Prior}	0.7339	0.5483	0.3940	0.3531	0.5494	0.4423
$M_{SVM-linear}$	0.8569	0.6899	0.6157	0.5837	0.7260	0.6322
$M_{SVM-rbf}$	0.8484	0.6940	0.6241	0.5901	0.7266	0.6376
M_{LR}	**0.8612**	**0.7052**	**0.6807**	**0.6258**	**0.7632**	**0.6621**

Table 5: Precision@1 and R-Precision results on the test set. Here the present entities mean the entities existing in the encyclopedias. The absent entities mean the ones not existing in the encyclopedias.

but also related words. For example, "布拉德利中心 (Bradley Center)" in Baidubaike have 5 tags, i.e., "NBA", "体育 (sports)", "体育运动 (sports)", "篮球 (basketball)", and "场馆 (arena)". Among them, only "场馆 (arena)" is a proper hypernym whereas the others are some related words indicating merely thematic vicinity. Comparing the results of SR_{10} and SR_{10} + ET, we can see that collecting candidates from ET can improve coverage, although many incorrect candidates are added in at the same time.

The HW source provides 0.87 candidates on average with 41.35% coverage rate. That is to say, for these entities, people can infer the semantic classes when they see the surface lexicon.

At last, we combine the candidates from all of the three sources as the input of the ranking methods. The coverage rate reaches 93.24%.

We also compare with the manually constructed semantic thesaurus CilinE mentioned in Section 3.2.1. Only 29 entities exist in CilinE (coverage rate is only 1.54%). That is why we try to automatically extract hypernym relations.

5.2 Evaluation of the Ranking

5.2.1 Overall Performance Comparison

In this section, we compare our proposed methods with other methods. Table 5 lists the performance measured by precision at rank 1 and R-precision of some key methods. The precision-recall curves of all the methods are shown in Figure 2. Table 7 lists the maximum F-scores.

$M_{Pattern}$ refers to the pattern-based method of Hearst (1992). We craft Chinese Hearst-style patterns (Table 6), in which E represents an entity and H represents one of its hypernyms. Following Evans (2004), we combine each pattern and each entity and submit them into the Baidu search engine. For example, for an entity E, we search "E 是一个 (E is a)", "E 等 (E and other)", and so on. We select top 100 search results of each query and get 1,285,209 results in all for the entities in the test set. Then we use the patterns to extract hypernyms from the search results. The result shows that 508 correct hypernyms are extracted for 568 entities (1,529 entities in total). Only a small part of the entities can be extracted hypernyms for. This is mainly because only a few hypernym relations are expressed in these fixed patterns in the web, and many ones are expressed in more flexible manners. The hypernyms are ranked based on the count of evidences where the hypernyms are extracted.

Pattern	Translation
E 是(一个/一种) H	E is a (a kind of) H
E (、)等 H	E(,) and other H
H (，)叫(做) E	H(,) called E
H (，)(像)如 E	H(,) such as E
H (，)特别是 E	H(,) especially E

Table 6: Chinese Hearst-style lexical patterns

M_{Snow} is the method originally proposed by Snow et al. (2005) for English but we adapt it for Chinese. We consider the top 100 search results for each known hypernym-hyponym pairs as a corpus to extract lexico-syntactic patterns. Then, an LR classifier is built based on this patterns to recognize hypernym relations. This method considers all nouns co-occurred with the focused entity in the same sentences as candidate hypernyms. So the number of candidates is huge, which causes inefficiency. In

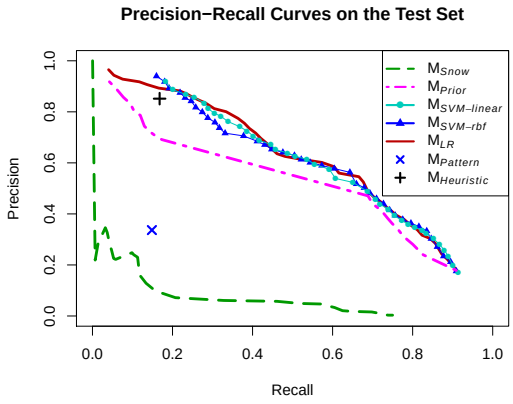


Figure 2: Precision-Recall curves on the test set

our corpus, there are 652,181 candidates for 1,529 entities (426.54 for each entity on average), most of which are not hypernyms. One possible reason is that this method relies on an accurate syntactic parser and it is difficult to guarantee the quality of the automatically extracted patterns. Even worse, the low quality of the language in the search results may make this problem more serious.

M_{Prior} refers to the ranking method based on only the prior of a candidate being a hypernym. As Table 5 shows, it outperforms M_{Snow} and achieves comparable results with $M_{Pattern}$ on Precision@1 and R-Precision.

Based on the features proposed in Section 3.2.1, we train several statistical models based on SVM and LR on the training data. $M_{SVM-linear}$ and $M_{SVM-rbf}$ refer to the SVM models based on linear kernels and RBF kernels respectively. M_{LR} refers to the LR model. The probabilities[6] output by the models are used to rank the candidate hypernyms. All of the parameters which need to be set in the models are selected on the development set. Table 5 shows the best models based on each algorithm. These supervised models outperform the previous methods. M_{LR} achieves the best performance.

The precision-recall plot of the methods on the test set is presented in Figure 2. $M_{Heuristic}$ refers to the heuristic approach, proposed in Section 3.2.2, to collect training data. Because this method cannot provide ranking information, it is not listed in Table 5. For fair comparison of R-precision and recall, we add the extra correct hypernyms from $M_{Pattern}$ and M_{Snow} to the test data set. The models based on SVM and LR still perform better than the other methods. $M_{Pattern}$ and M_{Snow} suffer from low recall and precision. $M_{Heuristic}$ get a high precision but a low recall, because it can only deal with a part of entities appearing in encyclopedias. The precision of $M_{Heuristic}$ reflects the quality of our training data. We summarize the maximum F-score of different methods in Table 7.

[6]The output of an SVM is the distance from the decision hyper-plane. Sigmoid functions can be used to convert this uncalibrated distance into a calibrated posterior probability (Platt, 1999).

Method	Max. F-score
$M_{Pattern}$	0.2061
M_{Snow}	0.1514
$M_{Heuristic}$	0.2803
M_{Prior}	0.5591
$M_{SVM-linear}$	0.5868
$M_{SVM-rbf}$	**0.6014**
M_{LR}	0.5998

Table 7: Summary of maximum F-score on the test set

Feature	P@1	R-Prec	Max. F-score
All	**0.7632**	**0.6621**	**0.5998**
– Prior	0.7534	0.6546	0.5837
– Is_Tag	0.6965	0.6039	0.5605
– Is_Head	0.7018	0.6036	0.5694
– In_Titles	0.7436	0.6513	0.5868
– Synonyms	0.7495	0.6493	0.5831
– Radicals	0.7593	0.6584	0.5890
– Source_Num	0.7364	0.6556	0.5984
– Lexicon	0.7377	0.6422	0.5851
– Source_Info	0.6128	0.5221	0.5459

Table 8: Performance of LR models with different features on the test set

5.2.2 Feature Effect

Table 8 shows the impact of each feature on the performance of LR models. When we remove any one of the features, the performance is degraded more or less. The most effective features are Is_Tag and Is_Head. The last line in Table 8 shows the performance when we remove all features about the source information, i.e., Is_Tag, Is_Head, and

Entity	Top-1 Hypernym	Entity	Top-1 Hypernym
头孢哌酮钠(cefoperazone sodium)	药品(drug)	圆舵鲣(bullet tuna)	鱼类(fish)
佛手卷(finger citron rolls)	小吃(snack)	锆英石(zirconite)	矿石(ore)
复仇者联盟(The Avengers)	电影(movie)	费利克斯托(Felixstowe)	港口(port)
套叠鞭(mastigium)	基准(datum)	基节臼(coxal cavity)	植物(plant)
乙醇胺磷酸转移酶 (Ethanolamine phosphotransferase)	生物 (organism)	彗发 (coma)	知识 (knowledge)

Table 10: Examples of entity-hypernym pairs extracted by M_{LR}

Domain	P@1	R-Prec	Max. F-score
Biology	0.8165	0.7203	0.6424
Health Care	0.7354	0.5962	0.6061
Food	0.7450	0.6634	0.6938
Movie	0.9310	0.8069	0.7031
Industry	0.6286	0.5841	0.4624
Others	0.6324	0.4936	0.4318

Table 9: Performance of M_{LR} in various domains

Source_Num. The performance is degraded sharply. This indicates the importance of the source information for hypernym ranking.

5.2.3 The Performance in Each Domain

In this section, we evaluate the performance of M_{LR} method in various domains. We can see from Table 9 that the performance in movie domain is best while the performance in industry domain is worst. That is because the information about movies is abundant on the web. Furthermore, most of movies have encyclopedia pages. It is easy to get the hypernyms. In contrast, the entities in industry domain are more uncommon. On the whole, our method is robust for different domains. In Table 10, some instances in various domains are presented.

5.3 Error Analysis

The uncovered entities[7] and the false positives[8] are analyzed after the experiments. Some error examples are shown in Table 10 (in red font).

[7]Uncovered entities are entities which we do not collect any correct hypernyms for in the first step.

[8]False positives are hypernyms ranked at the first places, but actually are not correct hypernyms.

Uncovered entities: About 34% of the errors are caused by uncovered entities. It is found that many of the uncovered entities are rare entities. Nearly 36% of them are very rare and have only less than 100 search results in all. When we can't get enough information of an unknown entity from the search engine, it's difficult to know its semantic meaning, such as "套叠鞭 (mastigium)", "基节臼 (coxal cavity)", "彗发 (coma)". The identification of their hypernyms requires more human-crafted knowledge. The ranking models we used are unable to select them, as the true synonyms are often below rank 10.

False positives: The remained 66% errors are false positives. They are mainly owing to the fact that some other related words in the candidate lists are more likely hypernyms. For example, "生物 (organism)" is wrongly recognized as the most probable hypernym of "乙醇胺磷酸转移酶 (Ethanolamine phosphotransferase)", because the entity often co-occurs with word "生物 (organism)" and the latter is often used as a hypernym of some other entities. The correct hypernyms actually are "酶 (enzyme)", "化学物质 (chemical substance)", and so on.

6 Conclusion

This paper proposes a novel method for finding hypernyms of Chinese open-domain entities from multiple sources. We collect candidate hypernyms with wide coverage from search results, encyclopedia category tags and the head word of the entity. Then, we propose a set of features to build statistical models to rank the candidate hypernyms on the training data collected automatically. In our experiments, we show that our method outperforms the state-of-the-art methods and achieves the best preci-

sion of 76.32% on a manually labeled test dataset. All of the features which we propose are effective, especially the features of source information. Moreover, our method works well in various domains, especially in the movie and biology domains. We also conduct detailed analysis to give more insights on the error distribution. Except some language dependent features, our approach can be easily transfered from Chinese to other languages. For future work, we would like to explore knowledge from more sources to enhance our model, such as semantic thesauri and infoboxes in encyclopedias.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) via grant 61133012, 61073126 and the National 863 Leading Technology Research Project via grant 2012AA011102. Special thanks to Zhenghua Li, Wanxiang Che, Wei Song, Yanyan Zhao, Yuhang Guo and the anonymous reviewers for insightful comments and suggestions. Thanks are also due to our annotators Ni Han and Zhenghua Li.

References

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126, College Park, Maryland, USA, June.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Coling 2010: Demonstrations*, pages 13–16, Beijing, China, August.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 168–175.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

Oren Etzioni, Michele Banko, and Michael J Cafarella. 2006. Machine reading. In *AAAI*, volume 6, pages 1517–1519.

Richard Evans. 2004. A framework for named entity recognition in the open domain. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, 260:267–274.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2012. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, pages 1–63.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 893–903, Jeju Island, Korea, July.

Paul McNamee, Rion Snow, Patrick Schone, and James Mayfield. 2008. Learning named entity hyponyms for question answering. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 799–804.

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145.

John Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the 2009 AAAI Spring Symposium on Learning by Reading and Learning to Read*, pages 88–93.

Cederberg Scott and Widdows Dominic. 2003. Using lsa and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 111–118.

Sidney Siegel and N John Castellan Jr. 1988. Nonparametric statistics for the behavioral sciences. McGraw-Hill, New York.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Peter Turney, Michael L Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference RANLP-2003*, pages 482–489.

Fan Zhang, Shuming Shi, Jing Liu, Shuqi Sun, and Chin-Yew Lin. 2011. Nonlinear evidence fusion and propagation for hyponymy relation mining. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1159–1168, Portland, Oregon, USA, June.

A Semantically Enhanced Approach to Determine Textual Similarity

Eduardo Blanco and **Dan Moldovan**
Lymba Corporation
Richardson, TX 75080 USA
{eduardo,moldovan}@lymba.com

Abstract

This paper presents a novel approach to determine textual similarity. A layered methodology to transform text into logic forms is proposed, and semantic features are derived from a logic prover. Experimental results show that incorporating the semantic structure of sentences is beneficial. When training data is unavailable, scores obtained from the logic prover in an unsupervised manner outperform supervised methods.

1 Introduction

The task of Semantic Textual Similarity (Agirre et al., 2012) measures the degree of semantic equivalence between two sentences. Unlike textual entailment (Giampiccolo et al., 2007), textual similarity is symmetric, and unlike both textual entailment and paraphrasing (Dolan and Brockett, 2005), textual similarity is modeled using a graded score rather than a binary decision. For example, sentence pair (1) below is very similar [5 out of 5], (2) is somewhat similar [3 out of 5] and (3) is not similar at all [0 out of 5]:

1. Someone is removing the scales from the fish.
   A person is descaling a fish.
2. A woman is chopping an herb.
   A man is finely chopping a green substance.
3. A cat is playing with a watermelon on a floor.
   A man is pouring oil into a pan.

State-of-the-art systems to determine textual similarity (Bär et al., 2012; Šarić et al., 2012; Banea et al., 2012) do not account for the semantic structure of sentences, and mostly rely on word pairings and knowledge derived from large corpora, e.g., Wikipedia. Regardless of details, each word in sent_1 is paired with the word in sent_2 that is most similar according to some similarity measure. Then, all similarities are added and normalized by the length of sent_1 to obtain the similarity score from sent_1 to sent_2. The process is repeated to obtain the similarity score from sent_2 to sent_1, and both scores are then averaged to determine the overall textual similarity. Several word-to-word similarity measures are often combined with other shallow features, e.g., n-gram overlap, syntactic dependencies, to obtain the final similarity score.

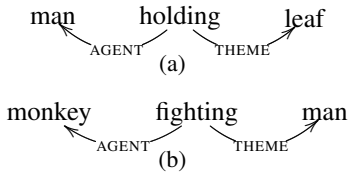


Figure 1: Semantic representation of 1(a) *A man is holding a leaf* and 1(b) *A monkey is fighting a man*.

Consider sentences 1(a) *A man is holding a leaf* and 1(b) *A monkey is fighting a man*. These two sentences are very dissimilar, the only commonality is the concept '*man*'. Any approach that blindly searches for the word in 1(b) that is the most similar to word '*man*' in 1(a) will find '*man*' from 1(b) to be a perfect match. One of three content words is a match and thus the estimated similarity will be much higher than it actually is.

Consider now the semantic representations for sentences 1(a) and 1(b) in Figure 1. '*man*' plays the role of AGENT in 1(a), and THEME in 1(b). While in both sentences the word '*man*' encodes the same concept, their semantic functions with respect to other concepts are different. Intuitively, it seems reasonable to penalize the similarity score based on the role discrepancy.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1235–1245,
Seattle, Washington, USA, 18-21 October 2013.

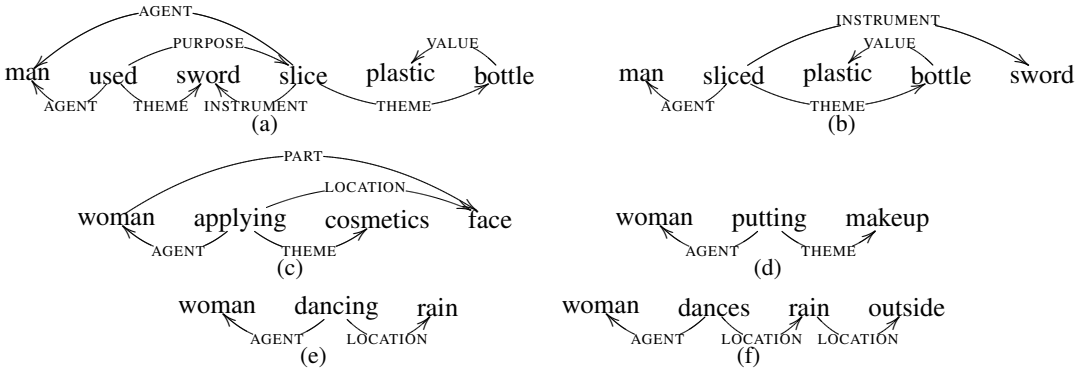


Figure 2: Semantic representations of 2(a) *The man used a sword to slice a plastic bottle*, 2(b) *A man sliced a plastic bottle with a sword*, 2(c) *A woman is applying cosmetics to her face*, 2(d) *A woman is putting on makeup*, 2(e) *A woman is dancing in the rain*, and 2(f) *A woman dances in the rain outside*. Pairs (a, b), (c, d) and (e, f) are highly similar even though concepts and relations only match partially.

This paper proposes a novel approach to determine textual similarity. Semantic representations of sentences are exploited, syntactic features omitted and the only external resource used in WordNet (Miller, 1995). The main novelties of our approach are: it (1) derives semantic features from a logic prover to be used in a machine learning framework; (2) uses three logic form transformations capturing different levels of knowledge; and (3) incorporates semantic representations extracted automatically.

1.1 Matching Semantic Representations and Determining Textual Similarity

Throughout this paper, the semantic representation of a sentence comprises the concepts in it, semantic relations linking those concepts and named entities qualifying them. First, we note that existing tools to extract semantic relations and named entities are not perfect, thus any system relying on them will suffer from incomplete and incorrect representations. Second, even if flawless representations were readily available, the problem of determining textual similarity cannot be reduced to matching semantic representations: partial matches may correspond to completely similar sentences. The rest of this section illustrates this point with the examples in Figure 2. Our approach (Section 3) copes with the inherent errors made by tools used to obtain semantic representations and learns which parts of a representation are important to determine textual similarity.

Consider sentences 2(a) *The man used a sword to slice a plastic bottle* and 2(b) *A man sliced a plastic bottle with a sword*. Both sentences have high similarity [5 out of 5], and yet their semantic representations only match partially. In this example, the verb '*used*' in 2(a) and its semantic links are somewhat semantically superfluous. Note that in other cases, missing a semantic relation signals lower similarity, e.g., *I had fun [at the party]*$_{\text{LOCATION}}$ and *I had fun*, while similar, do not convey the same meaning.

Sentence 2(c) *A woman is applying cosmetics to her face* and 2(d) *A woman is putting on makeup* are highly similar even though the latter specifies neither the LOCATION where the '*makeup*' is applied nor the fact that a PART of the '*woman*' is her '*face*'. Similarly, sentences 2(e) *A woman is dancing in the rain* and 2(f) *A woman dances in the rain outside* are semantically equivalent since '*rain*' always has LOCATION '*outside*': missing this information does not carry loss of meaning.

2 Related Work

Determining similarity between text snippets is relevant to information retrieval (Hatzivassiloglou et al., 1999), paraphrase recognition (Madnani and Dorr, 2010), grading answers to questions (Mohler et al., 2011) and many others. We focus on recent work and emphasize the differences from our approach.

The SemEval 2012 Task 6: A Pilot on Semantic Textual Similarity (Agirre et al., 2012) brought together 35 teams that competed against each other. The top 3 performers (Bär et al., 2012; Šarić et al., 2012; Banea et al., 2012), followed a ma-

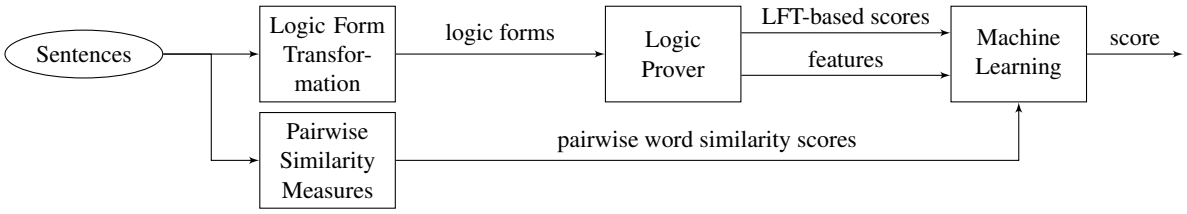


Figure 3: Main components of our system to determine textual similarity.

chine learning approach with features that do not take into account the semantic structure of sentences, e.g., n-grams, word overlap, evaluation measures for machine translation, pairwise word similarities, syntactic dependencies. All three used WordNet, Wikipedia and other large corpora. In particular, Banea et al. (2012) obtained models from 6 million Wikipedia articles and more than 9.5 million hyperlinks; Bär et al. (2012) used Wiktionary[1], which contains over 3 million entries; and Šarić et al. (2012) used The New York Times Annotated Corpus (Sandhaus, 2008), which contains over 1.8 million news articles, and Google n-grams (Lin et al., 2012), which consists of approximately 24GB of compressed text files. Our approach only uses WordNet, by far the smallest external resource with less than 120,000 synsets.

Participants that incorporated information about the semantic structure of sentences (Glinos, 2012; Rios et al., 2012)[2] did not perform at the top. Out of 88 runs, they were ranked 16, 36 and 64. We believe this is because they use semantic relations to calculate some ad-hoc similarity score. In contrast, our approach derives features from semantic representations encoded using logic, and combine these features using machine learning. Moreover, we use three logic form transformations capturing different levels of knowledge, from only content words to semantic structure. In turn, this allows us to boost performance by relying on semantics when simpler shallow methods fail.

A few logic-based approaches to recognize textual entailment are similar to the work presented here. Bos and Markert (2006) extract semantic representations with Boxer (Bos et al., 2004) and incorporate background knowledge from external resources. They use a standard theorem prover and extract 8 features that are later combined using machine learning. Raina et al. (2005) use a logic form transformation derived from dependency parses and named entities. They use abductive reasoning and define an assumption cost model to account for partial entailments. Unlike them, we define three logic from transformations, use a modified resolution step and extract hundreds of features from the proofs. Tatu and Moldovan (2005) use a modified logic prover that drops predicates when a proof cannot be found. Unlike us, they do not drop unbound predicates and use a single logic form transformation. Another key difference is that they assign fixed weights to predicates a priori instead of using machine learning to determine them.

3 Approach

Our approach to determine textual similarity (Figure 3) is grounded on using semantic features derived from a logic prover that are later combined in a standard supervised machine learning framework. First, sentences are transformed into logic forms (lft_1, lft_2). Then, a modified logic prover is used to find a proof in both directions (lft_1 to lft_2 and lft_2 to lft_1). The prover yields similarity scores based on the number of predicates dropped and features characterizing the proofs. Additional similarity scores are obtained using standard pairwise word similarity measures. Finally, all scores and features are combined using machine learning to yield the final textual similarity score.

If training data is unavailable, only the LFT-based and individual pairwise word similarity scores apply, the machine learning component is the only one supervised. The rest of this section details each component and exemplifies it with 2(e) *A woman is dancing in the rain* and 2(f) *A woman dances in the rain outside*.

[1] http://www.wiktionary.org/

[2] A third team, *spirin2*, submitted results but a description paper could not be found in the ACL anthology.

	sent$_1$: *A woman is dancing in the rain.*
	semantic relations extracted: AGENT(*dancing*, *woman*), LOCATION(*dancing*, *rain*)
Basic	woman_N(x_1) & dance_V(x_2) & rain_N(x_3)
SemRels	woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2,x_1) & rain_N(x_3) & LOCATION_SR(x_2,x_3)
Full	woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2,x_1) & rain_N(x_3) & LOCATION_SR(x_2,x_3)
	sent$_2$: *A woman dances in the rain outside.*
	semantic relations extracted: AGENT(*dances*, *woman*), LOCATION(*dances*, *rain*)
Basic	woman_N(x_1) & dance_V(x_2) & rain_N(x_3) & outside_M(x_4)
SemRels	woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2,x_1) & rain_N(x_3) & LOCATION_SR(x_2,x_3)
Full	woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2,x_1) & rain_N(x_3) & LOCATION_SR(x_2,x_3) & outside_M(x_4)

Table 1: Examples of logic from transformation using modes *Basic*, *SemRels* and *Full*.

3.1 Logic Form Transformation

The logic form transformation (LFT) of a sentence is derived from the concepts in it, the semantic relations linking them and named entities. Unlike other LFT proposals (Zettlemoyer and Collins, 2005; Poon and Domingos, 2009), transforming sentences into logic forms is a straightforward step, the quality of the logic forms is determined by the output of standard NLP tools.

We distinguish six types of predicates:

- N for nouns, e.g., *woman*: woman_N(x_1).
- V for verbs, e.g., *dances*: dance_V(x_2).
- M for adjectives and adverbs, e.g., *outside*: outside_M(x_3).
- O for concepts encoded by other POS tags.
- NE for named entities, e.g., *guitar*: guitar_N(x_4) & instrument_NE(x_4).
- SR for semantic relations, e.g., *A woman dances*: woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2,x_1).

In order to overcome semantic relation extraction errors, we have experimented with three logic form transformation modes. Each mode captures different levels of knowledge:

Basic generates predicates for all nouns, verbs, modifiers and named entities. This logic form is parallel to accounting for content words, their POS tags and named entity types.

SemRels generates predicates for all semantic relations, concepts that are arguments of relations and named entities qualifying those concepts. This mode ignores concepts not linked to other concepts through a relation and might miss key concepts if some relations are missing. If no semantic relations are found, this mode backs off to *Basic* to avoid empty logic forms.

Full generates predicates for all concepts, all semantic relations and all named entities. It is equivalent to *SemRels* after adding predicates for concepts that are not arguments of a semantic relation.

Table 1 exemplifies the three logic form modes. If perfect semantic relations were always available, *SemRels* would be the preferred mode. However, this is often not the case and combining the three logic forms yields better performance (Section 4). Note that since relation LOCATION(*rain*, *outside*) is not extracted from sent$_2$, predicate outside_M(x_4) is not present in mode *SemRels*.

3.2 Modified Logic Prover

Textual similarity is symmetric and therefore we find proofs in both directions (from lft_1 to lft_2 and from lft_2 to lft_1). The logic prover uses a modified resolution procedure to calculate a similarity score and features derived from the proof. The rest of this section exemplifies one direction, lft_1 to lft_2. The logic prover is a modification of OTTER[3] (McCune and Wos, 1997), an automated theorem prover for first-order logic. For the textual similarity task, we load lft_1 and $\neg lft_2$ to the *set of support* and lexical chain axioms to the *usable list*. Then, the logic prover begins its search for a proof. Two scenarios are possible: (1) a contradiction is found, i.e., a proof is found; or (2) a contradiction cannot be found. The modifications to the standard resolution

[3] http://www.cs.unm.edu/~mccune/otter/

sent$_1$: *A woman plays an electric guitar*			sent$_2$: *A man is cutting a potato*	
lft_1: woman_N(x_1) & play_V(x_2) & AGENT_SR(x_2, x_1) & electric_M(x_3) & guitar_N(x_4) & instrument_NE(x_4) & VALUE_SR(x_4, x_3) & THEME_SR(x_2, x_4)				
$\neg lft_2$: ¬man_N(x_1) ∨ ¬cut_V(x_2) ∨ ¬AGENT_SR(x_2, x_1) ∨ ¬potato_N(x_3) ∨ ¬THEME_SR(x_2, x_3)				
Step	Predicate dropped (regular)	Score	Predicate dropped (unbound)	Score
1	woman_N(x_1)	0.875	n/a	0.875
2	play_V(x_2)	0.750	AGENT_SR(x_2, x_1)	0.625
3	electric_M(x_3)	0.500	n/a	0.500
4	guitar_N(x_4)	0.375	instrument_NE(x_4), VALUE_SR(x_4, x_3), THEME_SR(x_2, x_4)	0.000

Table 2: Example of predicate dropping step by step. Predicates AGENT_SR(x_2, x_1) and THEME_SR(x_2, x_4) would not be dropped if unbound predicates were not dropped, yielding a score of 0.250 instead of 0.000.

procedure are used in scenario (2), when a proof cannot be found. In this case, predicates from *lft*$_1$ are dropped until a proof is found. The worst case occurs when all predicates in *lft*$_1$ are dropped. The goal of the dropping mechanism is to force the prover to always find a proof, and penalize partial proofs accordingly.

Lexical chain axioms are extracted from WordNet. Assuming each word w in sent$_1$ has the first sense, axioms $w \rightarrow c$, where c is at most distance 2 in the WordNet hierarchy are generated. For example, axioms derived from *woman* include *woman* → *female*, *woman* → *mistress*, *woman* → *widow* and *woman* → *madam*. Although simple, this WordNet expansion proved useful in our experiments.

3.2.1 Predicate Dropping Criteria

When a proof cannot be found, individual predicates from *lft*$_1$ not present in *lft*$_2$ are dropped. A greedy algorithm was implemented for this step: out of all predicates from *lft*$_1$ not present in *lft*$_2$, drop whichever occurs first.

Dropping a predicate is not done in isolation. After dropping a predicate, all predicates that become unbound are dropped as well. With our current logic form transformation, dropping a noun, verb or modifier may make a semantic relation (_SR) or named entity (_NE) predicate unbound. To avoid determining high similarity between sentences with a *common* semantic structure but *unrelated* concepts instantiating this structure, predicates encoding semantic relations and named entities are automatically dropped when they become unbound.

3.2.2 Proof Scoring Criterion

The score assigned to the proof from *lft*$_1$ to *lft*$_2$ is calculated as the ratio of number of predicates in *lft*$_1$ not dropped to find the proof over the original number of predicates in *lft*$_1$.

Note that the dropping mechanism, and in particular whether predicates that become unbound are automatically dropped, greatly impact the proof obtained and its score (Table 2). If predicates that become unbound were not automatically dropped in each step, instrument_NE(x_4) and VALUE_SR(x_4, x_3) would be dropped in steps 5 and 6, AGENT_SR(x_2, x_1) and THEME_SR(x_2, x_4) would not be dropped, and the final score would be 0.250 instead of 0.000. In plain English, dropping unbound predicates avoids matching semantic structures instantiated by unrelated concepts.

3.2.3 Feature Selection

While the proof score can be used directly as an estimator of the similarity between *lft*$_1$ and *lft*$_2$, additional features are extracted from the proof itself. Namely, for each predicate type (N, V, M, O, SR, NE), we count the number of predicates present in *lft*$_1$, the number of predicates dropped to find a proof for *lft*$_2$ and the ratio of the two counts. These three counts are also calculated for each specific semantic relation predicate (AGENT_SR, LOCATION_SR, etc.). An example of score and feature calculation in both directions is shown in Table 3.

The LFT-based scores and features are fed to a machine learning algorithm. Specifically, there are 477 features derived from the logic prover:

- 9 LFT-based scores (3×3; three scores (2 directions and average), three LFT modes)

lft_1: woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2, x_1) & rain_N(x_3) & LOCATION_SR(x_2, x_3)																
lft_2: woman_N(x_1) & dance_V(x_2) & AGENT_SR(x_2, x_1) & rain_N(x_3) & LOCATION_SR(x_2, x_3) & outside_M(x_4)																
lft_1 to lft_2	pred. dropped	none														
	score	1														
	features	n_t	n_d	n_r	v_t	v_d	v_r	m_t	m_d	m_r	ne_t	ne_d	ne_r	sr_t	sr_d	sr_r
		2	0	0	1	0	0	0	0	0	0	0	0	2	0	0
lft_2 to lft_1	pred. dropped	outside_M(x_4)														
	score	$5/6 = 0.833$														
	features	n_t	n_d	n_r	v_t	v_d	v_r	m_t	m_d	m_r	ne_t	ne_d	ne_r	sr_t	sr_d	sr_r
		2	0	0	1	0	0	1	1	1	0	0	0	2	0	0

Table 3: Two logic forms and output of logic prover in both directions. For each predicate type (n, v, m, o, ne, sr) and semantic relation type (AGENT, LOCATION, etc.) features indicate the total number of predicates, the number of predicates dropped until a proof is found and ratio of the two counts ($_t$, $_d$ and $_r$ respectively). We omit the features for predicate O and individual semantic relations because of space constraints.

- 108 features for predicates ($3 \times 6 \times 3 \times 2 = 108$; three features for each of the six predicate types, three LFT modes, two directions)

- 360 features specific to a semantic relation ($3 \times 20 \times 3 \times 2 = 360$; three features for each of the 20 semantic relations types, three LFT modes, two directions)

3.3 Pairwise Word Similarities

Pairwise word similarity measures between concepts have been long studied, and they have been used for the task of textual similarity before (Mihalcea et al., 2006). We incorporate scores derived using these measures for comparison purposes and to improve robustness in our approach.

Basically, each open-class word in $sent_1$ is paired with the open-class word in $sent_2$ that is most similar according to some similarity measure. All these individual similarities are summed and normalized by the length of $sent_1$ to find the similarity between $sent_1$ and $sent_2$. The process is repeated from $sent_2$ to $sent_1$ to obtain the similarity between $sent_2$ and $sent_1$, and both overall similarities are averaged to determine the final similarity score.

We have experimented with measures Path (distance in a taxonomy), LCH (Leacock and Chodorow, 1998), Lesk (Lesk, 1986), WUP (Wu and Palmer, 1994), Resnik (Resnik, 1995), Lin (Lin, 1998) and JCN (Jiang and Conrath, 1997), and use the WordNet::Similarity package[4].

[4] http://wn-similarity.sourceforge.net/

3.4 Machine Learning Algorithm

We follow a standard supervised machine learning framework. Instances from the training split are used to create a model that is later tested with test instances not seen during training. The model was tuned using 10-fold cross-validation over the training instances. As a learning algorithm, we use bagging with M5P decision trees (Quinlan, 1992; Wang and Witten, 1997) as implemented in the Weka software package (Hall et al., 2009).

4 Experiments and Results

Logic forms are derived from the output of state-of-the-art NLP tools developed previously and not tuned in any way to the current task or corpora. Our approach is not tied to any tool, set of named entities or relations. Any other semantic representation could be used; the only required modification would be the LFT component (Figure 3) so that it accounts for the subtleties of the representation of choice.

The named entity recognizer extracts 35 fine-grained types organized in a taxonomy (date, language, city, instrument, etc.) and was first developed for a question answering system (Moldovan et al., 2002). The implementation uses publicly available gazetteers as well as machine learning.

Semantic relations are extracted with Polaris (Moldovan and Blanco, 2012), a semantic parser that given text extracts semantic relations. Polaris is trained using FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), NomBank (Meyers et al., 2004), several SemEval corpora (Girju et al., 2007;

	Score	Sentence Pair	Notes
MSRpar (36/35) [750/750]	2.600	The unions also staged a five-day strike in March that forced all but one of Yale's dining halls to close. The unions also staged a five-day strike in March; strikes have preceded eight of the last 10 contracts.	Long sentences, difficult to parse; often several details are missing in one sentence but the pair is similar
MSRvid (13/13), [750/750]	0.000	A woman is swimming underwater. A man is slicing some carrots.	Short sentences, easy to parse
SMTeuroparl (56/21), [734/459]	4.250	Then perhaps we could have avoided a catastrophe. We would perhaps then able prevent a disaster.	One sentence often ungrammatical (SMT)
surprise.OnWN (–/16), [–/750]	1.500	the alleviation of distress a change for the better.	WN glosses, difficult to parse with standard tools
surprise.SMTnews (–/24), [–/399]	3.000	He did, but the initiative did not get very far What he has done without the initiative goes too far.	One sentence often ungrammatical (SMT)

Table 4: Examples of sentence pairs belonging to the five sources. The numbers between round (square) parenthesis indicate the average number of tokens per sentence pair (number of instances) in the train and test splits.

Pustejovsky and Verhagen, 2009; Hendrickx et al., 2010) and in-house annotations.

4.1 Corpora

We use the corpora released by SemEval 2012 Task 06: A Pilot on Semantic Textual Similarity[5] (Agirre et al., 2012). These corpora consist of pairs of sentences labeled with their semantic similarity score, ranging from 0.0 to 5.0. Sentence pairs come from five sources: (1) MSRpar, a corpus of paraphrases; (2) MSRvid, short video descriptions; (3) SMTeuroparl, output of machine translation systems and reference translations; (4) surprise.OnWN, OntoNotes (Hovy et al., 2006) and WordNet (Miller, 1995) glosses; and (5) surprise.SMTnews, output of machine translation systems in the news domain and gold translations. Examples can be found in Table 4, for more details refer to the aforementioned citation.

4.2 Results and Error Analysis

Results are reported using the same train and test splits provided by the organization of SemEval 2012 Task 6. For surprise.OnWn and surprise.SMTnews, only test data is available and supervised machine learning is not an option.

Table 5 shows results obtained with the test split not dropping and dropping unbound predicates. For comparison purposes, results of the top-3 performers and participants using the semantic structure of sentences are also shown. *LFT-score* systems output the score (average of both directions) obtained with the corresponding logic form transformation (Basic, SemRels or Full) and are unsupervised: training data with textual similarity scores is not used. The other three systems presented are supervised. *LFT-scores + features* combines the 9 LFT-scores and 468 features derived from the logic proof. *WN-scores* uses as features the 7 scores derived using pairwise word similarity measures. Finally, *All* combines the full set of 484 features. We indicate that the performance of one of our systems with respect to *LFT score Basic not dropping unbound predicates* is significant with * (confidence 99%) and † (confidence 95%).

Overall, systems that drop unbound predicates perform better than systems that do not drop them. The only noticeable exception is *LFT-score* with sentences from SMTeuroparl. However, best results for SMTeuroparl are obtained dropping unbound predicates and using *All* features. Henceforth, we comment on results dropping unbound predicates as they are higher.

Regarding logic form transformations, one can see a trend depending on the source of sentences. Polaris, the semantic parser, and the syntactic parser Polaris relies on are mostly trained in the news domain, and thus semantic representations have higher quality in that domain. For SMTeuroparl and SMTnews, the two corpora closest to the news domain, *Full* obtains better results than *Basic* and *SemRels*. The difference is most noticeable in SMTnews, where *Basic* yields 0.4616, *SemRels*

[5] http://www.cs.york.ac.uk/semeval-2012/task6/

	System			MSRpar	MSRvid	SMTeuroparl	OnWN	SMTnews
not dropping unbound predicates	noML	LFT score	Basic	0.4963	0.8198	0.5101	0.6103	0.4588
			SemRels	*0.3952	*0.6753	0.4920	*0.5055	0.4477
			Full	0.4525	*0.7024	0.5183	0.5895	0.4956
	ML	LFT scores + features		*0.5750	0.8466	0.4725	n/a	n/a
		WN scores		0.4978	0.8495	0.5217	n/a	n/a
		All		*0.5992	†0.8660	0.5194	n/a	n/a
dropping unbound predicates	noML	LFT score	Basic	†0.5552	0.8234	0.4994	0.6120	0.4616
			SemRels	0.4556	*0.7388	0.4871	*0.5113	0.4796
			Full	0.5250	*0.7672	0.5130	0.5895	†0.5291
	ML	LFT scores + features		*0.5770	0.8440	0.5277	n/a	n/a
		WN scores		0.4977	0.8495	0.5217	n/a	n/a
		All		***0.6157**	***0.8709**	†**0.5745**	n/a	n/a
Top performer	(Bär et al., 2012)			0.6830	0.8739	0.5280	0.6641	0.4937
	(Šarić et al., 2012)			0.6985	0.8620	0.3612	0.7049	0.4683
	(Banea et al., 2012)			0.5353	0.8750	0.4203	0.6715	0.4033
Team w/ semantic structure	spirin2			0.5769	0.8203	0.4667	0.5835	0.4945
	(Rios et al., 2012)			0.3628	0.6426	0.3074	0.2806	0.2082
	(Glinos, 2012)			0.2312	0.6595	0.1504	0.2735	0.1426

Table 5: Correlations obtained with the test split using our approach (not dropping and dropping unbound predicates), and results obtained by the top-3 performers and teams that included in their models features derived from the semantic structure of sentences. Statistically significant differences in performance between our systems and *LFT score Basic not dropping unbound predicates* are indicated with * (confidence 99%) and † (confidence 95%).

0.4796 (+0.0180) and *Full* 0.5291 (+0.0675 and +0.0495 respectively).

Outside the news domain (MSRpar, MSRvid, OnWN), *Basic* performs better than *SemRels* and *Full*, and *Full* performs better than *SemRels*. This leads to the conclusion that several semantic relations are often missing, and thus considering concepts even if they are not linked to other concepts via a semantic relation (*Full*) is more sound than ignoring them (*SemRels*).

When training data is available (MSRpar, MSRvid, SMTeuroparl), *LFT-scores + features* always outperforms the scores obtained with a single logic form transformation in an unsupervised manner. In other words, combining the scores obtained with the three logic form transformations and incorporating the additional features derived from the proofs improves performance. These results demonstrate that while a shallow logic form transformation (*Basic*) offers a strong baseline, it can be successfully complemented with logic form transformations that consider the semantic structure of sentences (*SemRels*, *Full*) and additional features characterizing the proofs. The improvements *LFT-scores + features* brings over the LFT-score obtained with *Basic* are substantial: 0.0218 (3.9%) for MSRpar, 0.0206 (2.5%) for MSRvid and 0.0283 (5.7%) for SMTeuroparl.

WN scores, which only uses as features the scores derived from pairwise word similarity measures, performs astonishingly well for some corpora. Namely, the differences in performance between *LFT scores + features* and *WN scores* in MSRvid and SMTeuroparl are minimal (−0.0055 and +0.0060). We believe this is due to the characteristics of these two corpora. Sentence pairs from MSRvid are very short with 13 tokens on average (Table 4), i.e., 6.5 tokens per sentence, and SMTeuroparl pairs are hard to parse: at least one comes from a machine translation system and is often ungrammatical.

Finally, dropping unbound predicates and using *All* features outperforms any other system. While both *LFT scores + features* and *WN scores* yield

good performance, the combination of the two outperforms them. Features extracted successfully complement each other for all corpora.

4.2.1 A Look at the ML Model

A benefit of decision trees is that one can inspect them. This section briefly gives insight about the most predictive features for *All* system.

The best features, i.e., features used in decisions closer to the root, are the LFT-scores calculated using *Basic* and *Full*. The LFT-score obtained using *SemRels* is used only when the other two cannot discriminate. Sorted by impact, the features extracted for verbs, nouns, semantic relations, named entities and modifiers follow. Towards the bottom of the tree, features for specific semantic relations (AGENT_SR, LOCATION_SR, etc.) are used. All three sources (MSRpar, MSRvid and SMTeuroparl) use features for THEME, LOCATION, AGENT and QUANTIFICATION. MSRpar also benefits from features for TIME and only SMTeuroparl benefits from TOPIC and MANNER.

4.2.2 Comparison with Previous Work

The semantic logic-based approach presented in this paper either outperforms other systems or performs in the top-3 (Table 5). Moreover, it clearly outperforms any other proposal that takes into account the semantic structure of sentences. These results lead to the conclusion that the semantic structure of sentences is worth considering and more effort should be devoted to deeper approaches.

When using sentences in the news domain (SMTeuroparl and SMTnews), i.e., when text is closer to the domain in which the NLP tools are trained, our semantic approach yields the best results known to date. For MSRvid, the system presented here performs as well as systems that use external knowledge (Section 2), the differences are minimal ($+0.0030$, -0.0089, $+0.0041$) and not statistically significant (confidence 99%). For MSRpar, the system performs amongst the top-3 even though two of these systems clearly obtained better results ($+0.0673$, $+0.0828$); both differences are statistically significant (confidence 99%).

Performance using surprise.OnWN deserves special comment. This corpus contains definitions, not sentences (Table 4). Lin's similarity measure alone yields a correlation of 0.6787, beating all systems in Table 5 except one of the top-3 performers (Šarić et al., 2012). Our semantic approach is not successful because we cannot extract valid representations, glosses are rarely a full sentence and are hard to parse with generic NLP tools like the ones we use.

5 Conclusions

This paper presents a novel approach to determine textual similarity that employs a logic prover to extract semantic features. A layered methodology to transform text into logic forms using three logic form transformations modes is presented. Each mode captures different levels of knowledge, from only content words to semantic representations automatically extracted. Best results are obtained when features derived from the logic prover are complemented with simpler pairwise word similarity measures. Features that account for the semantic structure of sentences are incorporated when needed, as the results obtained with systems *All*, *LFT scores* and *WN scores* show.

Our approach is heavily dependent on the quality of semantic representations, and unlike current top performers, does not require knowledge derived from Wikipedia or other large corpora. State-of-the-art NLP tools to extract semantic representations from text, which are far from perfect, yield promising results. Indeed, the approach outperforms previous work when the source text is relatively familiar to the tools, i.e., within the news domain, and performs in the top-3 otherwise.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational Linguistics*, Montreal, Canada.

Carmen Banea, Samer Hassan, Michael Mohler, and Rada Mihalcea. 2012. Unt: A supervised synergistic approach to semantic text similarity. In *Proceedings of the Sixth International Workshop on Se-*

mantic Evaluation (SemEval 2012), pages 635–642, Montréal, Canada, 7-8 June.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 435–440, Montréal, Canada, 7-8 June.

Johan Bos and Katja Markert. 2006. Recognising textual entailment with robust logical inference. In *Proceedings of the First international conference on Machine Learning Challenges: evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 404–426, Berlin, Heidelberg. Springer-Verlag.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of Coling 2004*, pages 1240–1246, Geneva, Switzerland, Aug 23–Aug 27. COLING.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. Association for Computational Linguistics.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, June. Association for Computational Linguistics.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18, Prague, Czech Republic, June. Association for Computational Linguistics.

Demetrios Glinos. 2012. Ata-sem: Chunk-based determination of semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 547–551, Montréal, Canada, 7-8 June.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Vasileios Hatzivassiloglou, Judith L. Klavans, and Eleazar Eskin. 1999. Detecting text similarity over short passages: exploring linguistic feature combinations via machine learning. In *In Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 203–212.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden, July. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA. Association for Computational Linguistics.

J.J. Jiang and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*.

C. Leacock and M. Chodorow, 1998. *Combining local context and WordNet similarity for word sense identification*, pages 305–332. In C. Fellbaum (Ed.), MIT Press.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.

Yuri Lin, Jean-Baptiste Michel, Erez Aiden Lieberman, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 System Demonstrations*, pages 169–174, Jeju Island, Korea, July. Association for Computational Linguistics.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Comput. Linguist.*, 36(3):341–387, September.

William McCune and Larry Wos. 1997. Otter: The cade-13 competition incarnations. *Journal of Automated Reasoning*, 18:211–220.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating noun argument structure for nombank. In *LREC*. European Language Resources Association.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence*, AAAI'06, pages 775–780. AAAI Press.

George A. Miller. 1995. WordNet: A Lexical Database for English. In *Communications of the ACM*, volume 38, pages 39–41.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 752–762, Portland, Oregon, USA, June. Association for Computational Linguistics.

Dan Moldovan and Eduardo Blanco. 2012. Polaris: Lymba's semantic parser. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, and Jan Odijk a nd Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 66–72, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1040.

D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. 2002. Lcc tools for question answering. In Voorhees and Buckland, editors, *Proceedings of the 11th Text REtrieval Conference (TREC-2002)*, NIST, Gaithersburg.

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

Hoifung Poon and Pedro Domingos. 2009. Unsupervised Semantic Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore, August. Association for Computational Linguistics.

James Pustejovsky and Marc Verhagen. 2009. SemEval-2010 Task 13: Evaluating Events, Time Expressions, and Temporal Relations (TempEval-2). In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 112–116, Boulder, Colorado, June. Association for Computational Linguistics.

Ross J. Quinlan. 1992. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore. World Scientific.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, AAAI'05, pages 1099–1105. AAAI Press.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Miguel Rios, Wilker Aziz, and Lucia Specia. 2012. Uow: Semantically informed text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 673–678, Montréal, Canada, 7-8 June.

Evan Sandhaus. 2008. The new york times annotated corpus. In *Linguistic Data Consortium*, Philadelphia, PA.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 371–378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448, Montréal, Canada, 7-8 June.

Y. Wang and I. H. Witten. 1997. Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning*. Springer.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA. Association for Computational Linguistics.

Luke Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–666, Arlington, Virginia. AUAI Press.

Understanding and Quantifying Creativity in Lexical Composition

Polina Kuznetsova **Jianfu Chen** **Yejin Choi**
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
{pkuznetsova,jianchen,ychoi}@cs.stonybrook.edu

Abstract

Why do certain combinations of words such as *"disadvantageous peace"* or *"metal to the petal"* appeal to our minds as interesting expressions with a sense of creativity, while other phrases such as *"quiet teenager"*, or *"geometrical base"* not as much? We present statistical explorations to understand the characteristics of lexical compositions that give rise to the perception of being original, interesting, and at times even artistic. We first examine various correlates of perceived creativity based on information theoretic measures and the connotation of words, then present experiments based on supervised learning that give us further insights on how different aspects of lexical composition collectively contribute to the perceived creativity.

1 Introduction

An essential property of natural language is the generative capacity that makes it possible for people to express indefinitely many thoughts through indefinitely many different ways of composing phrases and sentences (Chomsky, 1965). The possibility of novel, creative expressions never seems to exhaust. Various types of writers, such as novelists, journalists, movie script writers, and creatives in advertising, continue creating novel phrases and expressions that are original while befitting in expressing the desired meaning in the given situation. Consider unique phrases such as *"geological split personality"*, or *"intoxicating Shangri-La of shoes"*,[1] that continue flowing into the online text drawing attention from readers.

Writers put significant effort in choosing the perfect words in completing their compositions, as a well-chosen combination of words is impactful in readers' minds for rendering the precise intended meaning, as well as stimulating an increased level of cognitive responses and attention. Metaphors in particular, one of the quintessential forms of linguistic creativity, have been discussed extensively by studies across multiple disciplines, e.g., Cognitive Science, Psychology, Linguistics, and Literature (e.g., Lakoff and Johnson (1980), McCurry and Hayes (1992), Goatly (1997)). Moreover, recent studies based on fMRI begin to discover biological evidences that support the impact of creative phrases on people's minds. These studies report that unconventional metaphoric expressions elicit significantly increased involvement of brain processing when compared against the effect of conventional metaphors or literal expressions (e.g., Mashal et al. (2007), Mashal et al. (2009)).

Several linguistic elements, e.g., syntax, semantics, and pragmatics, are likely to be working together in order to lead to the perception of creativity. However, their underlying mechanisms by and large are yet to be investigated. In this paper, as a small step toward quantitative understanding of linguistic creativity, we present a focused study on lexical composition two content words.

Being creative, by definition, implies qualities such as being unique, novel, unfamiliar or unconventional. But not every unfamiliar combination of words would appeal as creative. For example, unfa-

[1] Examples from New York Times articles in 2013.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1246–1258,
Seattle, Washington, USA, 18-21 October 2013.

miliar biomedical terms, e.g., *"cardiac glycosides"*, are only informative without appreciable creativity. Similarly, less frequent combinations of words, e.g., *"rotten detergent"* or *"quiet teenager"*, though describing situations that are certainly uncommon, do not bring about the sense of creativity. Finally, some unique combinations of words can be just nonsensical , e.g., *"elegant glycosides"*.

Different studies assumed different definitions of linguistic creativity depending on their context and end goals (e.g., Chomsky (1976), Zhu et al. (2009), Gervás (2010), Maybin and Swann (2007), Carter and McCarthy (2004)). In this paper, as an operational definition, we consider a phrase creative if it is (a) unconventional or uncommon, and (b) expressive in an interesting, imaginative, or inspirational way.

A system that can recognize creative expressions could be of practical use for many aspiring writers who are often in need of inspirational help in searching for the optimal choice of words. Such a system can also be integrated into automatic assessment of writing styles and quality, and utilized to automatically construct a collection of interesting expressions from the web, which may be potentially useful for enriching natural language generation systems.

With these practical goals in mind, we aim to understand phrases with linguistic creativity in a broad scope. Similarly as the work of Zhu et al. (2009), our study encompasses phrases that evoke the sense of interestingness and creativity in readers' minds, rather than focusing exclusively on clearly but narrowly defined figure of speeches such as *metaphors* (e.g., Shutova (2010)), *similes* (e.g., Veale et al. (2008), Hao and Veale (2010)), and *humors* (e.g., Mihalcea and Strapparava (2005), Purandare and Litman (2006)). Unlike the study of Zhu et al. (2009), however, we concentrate specifically on how combinations of different words give rise to the sense of creativity, as this is an angle that has not been directly studied before. We leave the roles of syntactic elements as future research.

We first examine various correlates of perceived creativity based on information theoretic measures and the connotation of words, then present experiments based on supervised learning that give us further insights on how different aspects of lexical composition collectively contribute to the perceived creativity.

2 Theories of Creativity and Hypotheses

Many researchers, from the ancient philosophers to the modern time scientists, have proposed theories that attempt to explain the mechanism of creative process. In this section, we draw connections from some of these theories developed for general human creativity to the problem of quantitatively interpreting linguistic creativity in lexical composition.

2.1 Divergent Thinking and Composition

Divergent thinking (e.g., McCrae (1987)), which seeks to generate multiple unstereotypical solutions to an open ended problem has been considered as the key element in creative process, which contrasts with *convergent* thinking that find a single, correct solution (e.g., Cropley (2006)). Applying the same high-level idea to lexical composition, divergent composition that explores an unusual, unconventional set of words is more likely to be creative.

Note that the key novelty then lies in the *compositional* operation itself, i.e., the act of putting together a set of words in an unexpected way, rather than the rareness of individual words being used. In recent years there has been a swell of work on compositional distributional semantics that captures the compositional aspects of language understanding, such as sentiment analysis (e.g., Yessenalina and Cardie (2011), Socher et al. (2011)) and language modeling (e.g., Mitchell and Lapata (2009), Baroni and Zamparelli (2010), Guevara (2011), Clarke (2012), Rudolph and Giesbrecht (2010)). However, none has examined the compositional nature in quantifying creativity in lexical composition.

We consider two computational approaches to capture the notion of creative composition. The first is via various information theoretic measures, e.g., relative entropy reduction, to measure the surprisal of seeing the next word given the previous word. The second is via supervised learning, where we explore different modeling techniques to capture the statistical regularities in creative compositional operations. In particular, we will explore (1) compositional operations of vector space models, (2) kernels capturing the non-linear composition of different dimensions in the meaning space, (3) the use of

neural networks as an alternative to incorporate non-linearity in vector composition. (See §5).

2.2 Latent Memory and Creative Semantic Subspace

Although we expect that unconventional composition has a connection to creativeness of resulting phrases, that alone does not explain many counter examples where the composition itself is uncommon but the resulting expression is not creative due to lack of interestingness or imagination, e.g., *"room and water"*.[2] Therefore, we must consider additional conditions that give rise to creative phrases.

Let $\mathcal{S}$ represent the semantic space, i.e., the set of all possible semantic representation that can be expressed by a phrase that is composed of two content words.[3] Then we hypothesize that some subsets of semantic space $\{\mathcal{S}_i | \mathcal{S}_i \subset \mathcal{S}\}$ are semantically futile regions for appreciable linguistic creativity, regardless of how novel the composition in itself might be. Such regions may include technical domains such as law or pharmacology. Similarly, we expect semantically fruitful subsets of semantic space where creative expressions are more frequently found. For instance, phrases such as *"guns and roses"* and *"metal to the petal"* are semantically close to each other and yet both can be considered as interesting and creative (as opposed to one of them losing the sense of creativity due to its semantic proximity to the other).

This notion of creative semantic subspace connects to theories that suggest that latent memories serve as motives for creative ideas and that one's creativity is largely depending on prior experience and knowledge one has been exposed to (e.g., Freud (1908), Necka (1999), Glaskin (2011), Cohen and Levinthal (1990), Amabile (1997)), a point also made by Einstein: *"The secret to creativity is knowing how to hide your sources."*

Figure 5 presents visualized supports for creative semantic subspace,[4] where we observe that phrases in the neighborhood of legal terms are generally not creative, while the semantic neighborhood of *"kingdom"* and *"power"* is relatively more fruitful for composing *creative* (i.e., unique and uncommon while being imaginative and interesting, per our operational definition of creativity given in §1) word pairs, e.g., *invisible empire"*. In our empirical investigation, this notion of semantically fruitful and futile semantic subspaces are captured using distributional semantic space models under supervised learning framework (§5).

Source	# of uniq words	# of sent	Avg sent len	Entropy
QUOTES^{raw}	29498	49402	28	173.05
GLOSSES^{raw}	20869	7745	53	96.79

Table 1: Entropy of word distribution in datasets

Dataset	# of word pairs			percentage
	total	#(-)	#(+)	#(+)/total %
GLOSSES	1912	149	18	0.94
QUOTES	3298	204	35	1.06

Table 2: Distribution of creative(+)/common(-) word pairs over **GLOSSES** and **QUOTES** dataset.

2.3 Affective Language

Another angle we probe is the connection between creative expressions and the use of affective language. This idea is supported in part by previous research that explored the connection between figurative languages such as metaphors and sentiment (e.g., Fussell and Moss (1998), Rumbell et al. (2008), Rentoumi et al. (2012)). The focus of previous work was either on interpretation of the sentiment in metaphors, or the use of metaphors in the description of affect. In contrast, we aim to quantify the correlation between creative expressions (beyond metaphors) and the use of sentiment-laden words in a more systematic way. This exploration has a connection to the creative semantic subspace discussed earlier (§2.2), but pays a more direct attention to the aspect of sentiment and connotation.

3 Creative Language Dataset

We start our investigation by considering two types of naturally existing collection of sentences: (1) quotes and (2) dictionary glosses. We expect that quotes are likely to be rich in creative expressions, while dictionary glosses stand in the opposite spec-

[2] With additional context this example may turn into a creative one, but for simplicity we focus on phrases with two content words considered out of context.

[3] Investigation on recursive composition of more than two content words and the influence of syntactic packaging is left as future research.

[4] See §6 for more detailed discussion.

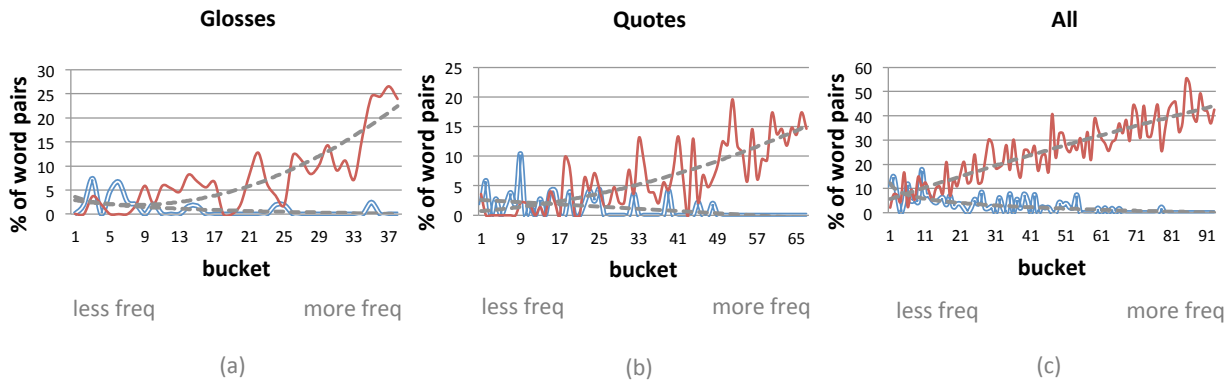


Figure 1: Distribution of *creative* (double lines in blue) versus *common* (single lines in red) word pairs with varying ranges of frequencies (x-axis) for **GLOSSES**, **QUOTES** and both datasets combined.

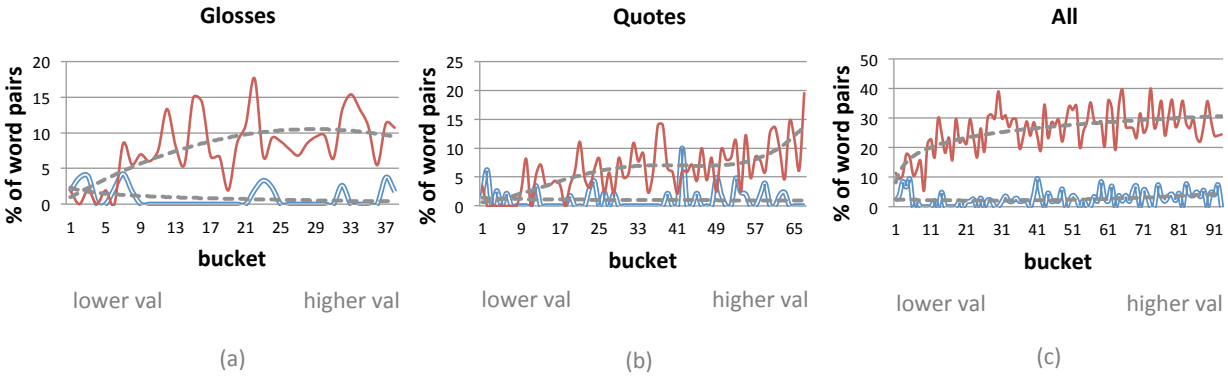


Figure 2: Distribution of *creative* (double lines in blue) versus *common* (single lines in red) word pairs with varying ranges of PMI values (x-axis) for **GLOSSES**, **QUOTES** and both datasets combined.

trum of being creative.

QUOTES^{raw}: We crawled inspirational quotes from "Brainy Quote".[5]

GLOSSES^{raw}: We collected glosses from Oxford Dictionary and Merriam-Webster Dictionary.[6] Overall we crawled about 8K definitions. Table 1 shows statistics of the dataset.[7]

Entropy of word distribution We conjecture that QUOTES and GLOSSES are different in terms of word variety, which can be quantified by the entropy of word distributions. To compute the entropy for each dataset, we use ngram statistics from the corresponding dataset to measure the probability of each word. As expected, QUOTES dataset has higher entropy than GLOSSES in Table 1.

3.1 Creative Word Pairs

We extract word pairs corresponding to the following syntactic patterns: [NN NN], [JJ NN], [NN JJ] and [JJ JJ]. Not all pairs from QUOTES^{raw} are creative, and likewise, not all pairs from GLOSSES^{raw} are uncreative. Therefore, we perform manual annotations to a subset of the collected pairs as follows. We obtain a small subset of pairs by applying stratified sampling based on bigram frequency buckets: first we sort word pairs by their bigram frequencies obtained from Web 1T corpus (Brants and Franz (2006)), group them into consecutive fre-

[5] http://www.brainyquote.com/

[6] http://oxforddictionaries.com/ and http://www.merriam-webster.com/. We only consider words appearing in both dictionaries to avoid unusual words such as compound words, e.g., "zero-base".

[7] QUOTES^{raw} contain 30K unique words and GLOSSES^{raw} has 20K unique words. QUOTES^{raw} have much arger number of sentences, while its average sentence is shorter.

quency buckets each of which containing 400 word pairs, then sample 40 word pairs from each bucket.

We label word pairs using Amazon Mechnical Turk (AMT) (e.g., Snow et al. (2008)). We ask three turkers to score each pair in 1-5 scale, where 1 is the least creative and 5 is the most creative. We then obtain the final creativity scale score by averaging the scores over 3 users. In addition, we ask turkers a series of yes/no questions to help turkers to determine whether the given pair is creative or not.[8] We determine the final label of a word pair based on two scores, creativity scale score and yes/no question-based score. If creativity scale score is 4 or 5 and question-based score is positive, we label the pair as creative. Similarly, if creativity scale score is 1 or 2 and question-based score is negative, we label the pair as common. We discard the rest from the final dataset. This filtering process is akin to the removal of neural sentiment in the early work of sentiment analysis (e.g., Pang et al. (2002)).[9] Table 2 shows the statistics of the resulting dataset.

Creative Pairs and their Frequencies: To gain insights on the stratified sample of word pairs, we plot the label ($\in \{creative, common\}$) distribution of word pairs as a function of simple statistics, such as a range (bucket) of bigram frequencies or PMI values of the given pair of words. Both bigram frequencies and PMI scores are computed based on Google Web 1T corpus Brants and Franz (2006). Figure 1 shows the results for word frequencies. As expected, word pairs with high frequencies are much more likely to be common, while word pairs with low frequencies can be either of the two. Also as expected, pairs extracted from QUOTES are relatively more likely to be creative than those from GLOSSES. In any case, it is clear that not all rare pairs are creative.

Creative Pairs and their PMI Scores: Similarly as above, Figure 2 plots the relation between the distribution of labels of word pairs and their corresponding PMI. As expected, pairs with high PMI are more likely to be common, though the trend is not as skewed as before.

Common	Creative
quiet teenager	inglorious success
constant longitude	thorny existence
watery juice	relaxed symmetry
noble political	sardonic destiny
diet cooking	dispassionate history
verbal interpretation	poetical enthusiasm
unwelcome situation	verbal beauty
migratory tuna	earth breathe
lousy businessman	disadvantageous peace
terrific marriage	alchemical marriage
solved issue	deep nonsense

Table 3: Sample Creative / Common Word Pairs

Final Dataset: From our initial annotation study, it became apparent to us that creative pairs are very rare, perhaps not surprisingly, even among infrequent pairs. In order to build the word pair corpus with as many creative pairs as possible, we focus on infrequent word pairs for further annotation, from which we construct a larger and balanced set of creative and common word pairs, with 394 word pairs for each class. The specific construction procedure is as follows: first combine all of the word pairs extracted from both QUOTES^{raw} and GLOSSES^{raw} as a single dataset, sort them by bigram frequency, group them into consecutive frequency buckets each of which has 40 word pairs; finally balance each frequency bucket, by discarding word pairs with higher frequency value from the larger class in that bucket. Examples of labeled word pairs are shown in Table 3. Hereafter we use this balanced dataset of word pairs for all experiments.[10]

4 Creativity Measures

4.1 Information Measures

In this section we explore information theoretic measures to quantify the *surprisal* aspect of creative word pairs, relating to the divergent, compositional nature of creativity discussed in §2.1.

Entropy of Context Seeing a word w changes our expectation on what might follow next. Some words have stronger selective preference (higher entropy) than others.

[8] E.g., *"is this word combination boring and not original?"* or *"does it provoke unusual imagination?"*.

[9] Cohen's Kappa and Pearson Correlation on the filtered data are 0.69 and 0.72 respectively. Corresponding scores for the unfiltered data drop to 0.26 and 0.29 respectively. All the experiments are performed on the filtered data.

[10] The resulting dataset is available at `http://www.cs.stonybrook.edu/~pkuznetsova/creativity/`

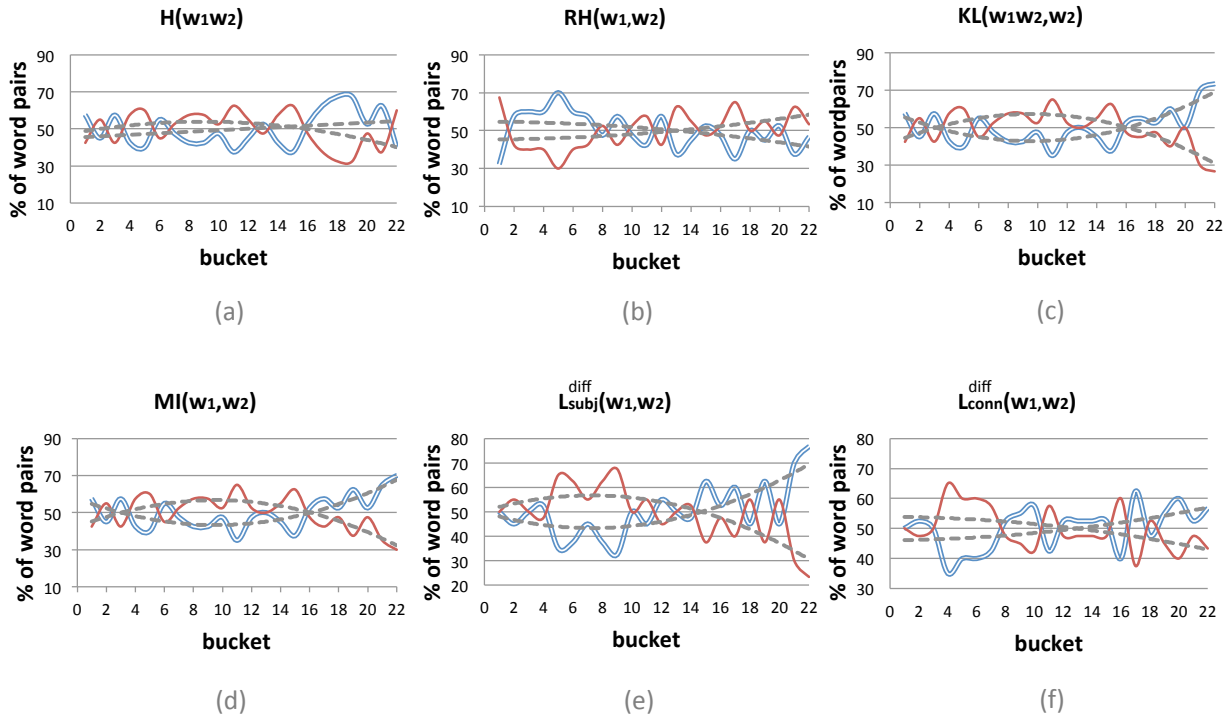


Figure 3: Distribution of *creative* (double lines in blue) versus *common* (single lines in red) word pairs with varying ranges of information or polarity measures (x-axis).

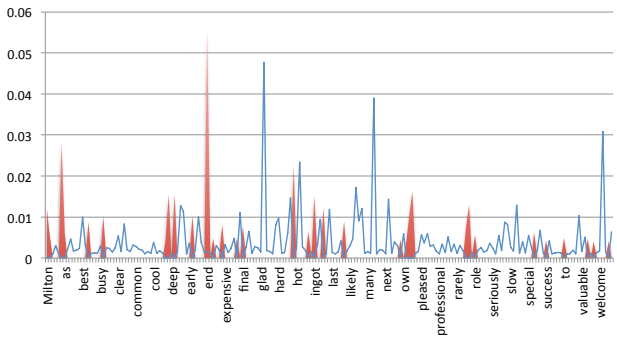


Figure 4: Conditional probability of neighboring words for "inglorious" (filled / red) and "very" (unfilled / blue).

For instance, the entropy after seeing *"very"* would be higher than that after seeing *"inglorious"*, as the former can be used in a wider variety of context than the later. Figure 4 visualizes relatively more skewed distribution of *"inglorious"*. We compute the entropy of future context conditioning on w_1, w_2 and w_1w_2, which we denote as $H(w_1)$, $H(w_2)$, $H(w_1w_2)$ respectively, latter is shown in Figure 3 – a.[11]

[11]As before, language models are drawn from Google Web 1T corpus Brants and Franz (2006).

Relative Entropy Transformation In order to focus more directly on the relative change of entropy as a result of composition, we compute Relative Entropy Transformation:

$$RH(w_1, w_2) = \frac{|H(w_1) - H(w_1w_2)|}{H(w_1) + H(w_1w_2)} \quad (1)$$

As expected (Figure 3 – b and Table 4), this relative quantity captures creativity better than the absolute measure $H(w_1w_2)$ computed above. The idea behind this measure has a connection to uncertainty reduction in psycholinguistic literature (e.g., Frank (2010), Hale (2003), Hale (2006)).

KL divergence To capture unusual combinations of words, we compare the difference between the distributional contexts of w_1 and w_1w_2 so that

$$KL(w_1w_2, w_1) = \sum_{w_i \in V} P(w_i|w_1, w_2) \log \frac{P(w_i|w_1, w_2)}{P(w_i|w_1)} \quad (2)$$

Figure (3 – c) shows that $KL(w_1w_2, w_1)$[12] is among

[12]We also compute $KL(w_1, w_2)$ in a similar manner as $KL(w_1w_2, w_1)$

the effective measures in capturing creative pairs.

Mutual Information Finally, we consider mutual information (Figure 3 – d):

$$MI(w_1, w_2) = \sum_{w_i \in V} P(w_i|w_1, w_2) \times \log \frac{P(w_i|w_1, w_2)}{P(w_i|w_1) \cdot P(w_i|w_2)} \quad (3)$$

Correlation coefficients Pearson coefficients for all measures are shown in Table 4. Interestingly, information theoretic measures that compare the distribution of word's context, such as $RH(w_1, w_2)$, $KL(w_1w_2, w_1)$ and $MI(w_1, w_2)$, capture the surprisal aspect of creativity better than simple frequencies or PMI scores that do not consider contextual changes. But even for those cases when the correlation is statistically significant, the values are not too high. We conjecture that there are two reasons for this. First, Pearson assumes linear correlations, hence not sensitive enough to capture non-linear correlations that are evident in graphs shown in Figure 3. Second, these measures only capture the surprisal aspect of creativity, missing the other important qualities: interestingness or imaginativeness.

4.2 Sentiment and Connotation

Next we investigate the connection between creativity and sentiment, as illustrated in §2.3. We consider both sentiment (more explicit) and connotation (more implicit) words,[13] and consider them with or without distinguishing the polarity (i.e., positive, negative). To determine sentiment and connotation, we use lexicons provided by OpinionFinder (Wilson et al. (2005)) and Feng et al. (2013) respectively. We denote polarity of a word w_i as $L(w_i)$.[14] When w_i has a negative polarity $L(w_i)$ is assigned a value of -1, and when w_i is positive $L(w_i)$ is equal to 1. We assume that a word is neutral when it is not in the lexicon, assigning 0 to $L(w_i)$. For a word pair w_1w_2 we compute absolute difference $L^{diff}(w_1, w_2)$ between polarities of tokens in a word pair in order to catch examples such as *"inglorious success"*.

[13]E.g., expressions such as *"blue sky"* or *"white sand"* are not sentiment-laden, but do have positive connotation.

[14]We denote polarity from OpinionFinder as L_{subj} and connotation as L_{conn}

Measure	Corr Coeff	p-value*	adj p-value**
pointwise, noncontextual			
$Freq(w_1w_2)$	0.014	0.67	0.86
$PMI(w_1, w_2)$	0.011	0.75	0.86
information theoretic, contextual			
$E(w_1)$	-0.038	0.26	0.49
$E(w_2)$	**-0.126**	0.00019	0.00083
$E(w_1, w_2)$	0.013	0.71	0.86
$RH(w_1, w_2)$	**0.113**	0.00081	0.0024
$KL(w_1w_2, w_1)$	**0.134**	7.152-05	0.00054
$KL(w_1, w_2)$	**-0.080**	0.018	0.039
$MI(w_1, w_2)$	**0.125**	0.00022	0.00083
sentiment & connotation			
$L_{subj}(w_1)$	0.006	0.87	0.87
$L_{subj}(w_2)$	0.031	0.36	0.60
$L^{diff}_{subj}(w_1, w_2)$	**0.168**	6.67e-07	1.00e-05
$L_{conn}(w_1)$	0.023	0.49	0.74
$L_{conn}(w_2)$	0.008	0.80	0.86
$L^{diff}_{conn}(w_1, w_2)$	**0.082**	0.015	0.038

Table 4: Pearson correlation between various measures and creativity of word pairs. Boldface denotes statistical significance ($p \leq 0.05$).
note *: Two-tailed p-value, 394 word pairs per class
note **: We used Benjamini-Hochberg method to adjust p-values for multiple tests

Table 4 shows Pearson coefficient for sentiment and connotation based measures. It turns out that polarity of each word on its own does not have a high impact on the creativity of a word pair. Rather, it is the difference between the two words that gives rise the sense of creativity.

4.3 Learning to Recognize Creativity

Now we put together all measures explored in §4.1 and 4.2 in a supervised-learning framework. As expected, rather than either one alone, the combination of various measures leads to the best performance:

$$\vec{F}_{12} = [RH(w_1, w_2); KL(w_1, w_2); H(w_1w_2); L^{diff}_{conn}(w_1, w_2); PMI(w1, w_2); H(w_2); KL(w_1w_2, w_1); KL(w_2, w_1); L^{diff}_{subj}(w_1, w_2); MI(w_1, w_2); Freq(w_1w_2); H(w_1)]$$

Table 5 shows the performance of the above feature vector with 12 features using libsvm (Chang and Lin, 2011). We use C-Support Vector Classification (C-SVC). Performance is reported in accuracy using 5-fold cross validation.[15]

[15]Among these 12 features, the feature selection algorithm

5 Learning Creative Pairs with Distributional Semantic Vectors

The measures explored in §4 were largely uninformed of distributional semantic dimensions of each word. However, in order to pursue the conceptual aspect of creativity illustrated in §2.2, that is, the notion of semantic subspaces that are inherently futile or fruitful for creativity, we need to incorporate semantic representations more directly. We therefore explore the use of distributional vector space models. Another goal of this section will be additional learning-based investigation to the compositional nature of creative word pairs, complementing the investigation in §4, which focused on the compositional aspect of creativity described in §2.1.

With above goals in mind, in what follows, we explore three different ways to learn compositional aspect of creative word pairs: (1) learning with explicit compositional vector operations (§5.1), (2) learning nonlinear composition via kernels (§5.2), (3) learning nonlinear composition via deep learning (§5.3). Note that in all these approaches, the notion of creative semantic subspace is integrated indirectly, as the feature representation always incorporates the resulting (composed) vector representations.

Baseline & Configuration We consider the concatenation of two word vectors $[\vec{w}_1; \vec{w}_2]$ as the baseline, since it can be viewed as what simple bag-of-word features would be. Since the size of creative pair dataset is not at scale yet, we choose to work with vector space models that are in reduced dimensions. We experimented with both Non-Negative Sparse Embedding (Murphy et al. (2012)) and neural semantic vectors of Huang et al. (2012), but report experiments with the latter only as those gave us slightly better results.

5.1 Compositional Vector Operations

We consider the following compositional vector operations inspired by recent studies for compositional distributional semantics (e.g., Guevara (2011), Clarke (2012), Mitchell and Lapata (2008), Widdows (2008)).

- **ADD**: $\vec{w}_1 + \vec{w}_2$
- **DIFF**: $abs(\vec{w}_1 - \vec{w}_2)$
- **MULT**: $\vec{w}_1 .* \vec{w}_2$
- **MIN**: $\min\{\vec{w}_1, \vec{w}_2\}$
- **MAX**: $\max\{\vec{w}_1, \vec{w}_2\}$

All operations take two input vectors $\in R^n$, and output a vector $\in R^n$. Each operation is applied element-wise. We then perform binary classification over the composed vectors using linear SVM. Besides using features based on the composed vectors, we also experiment with features based on concatenating multiple composed vectors, in the hope to capture more diverse compositional operations. See Table 5 for more details and experimental results.

5.2 Learning Nonlinear Composition via Kernels

As an alternative to explicit vector compositions, we also probe implicit operations based on non-linear combinations of semantic dimensions using kernels (e.g., Schölkopf and Smola (2002), Shawe-Taylor and Cristianini (2004)), in particular:

- Polynomial: $K(x, y) = (\gamma x^T y + r)^d,\ \gamma > 0$
- RBF: $K(x, y) = \exp(-\gamma \left\|x - y\right\|^2),\ \gamma > 0$
- Laplacian: $K(x, y) = \exp(-\gamma \left\|x - y\right\|),\ \gamma > 0$

5.3 Learning Non-linear Composition via Deep Learning

Yet another alternative to model non-linear composition is deep learning. To learn the non-linear transformation of a pair of semantic vectors, we explore the use of autoencoders (e.g., Pollack (1990), Voegtlin and Dominey (2005)). We follow the formulation of vector composition proposed by Socher et al. (2011) except that we do not stack autoencoders for recursion. More specifically, given the two input words $\vec{w}_1, \vec{w}_2 \in R^n$, we want to learn a vector space representation of their combination $\vec{p} \in R^n$. The recursive auto encoder (RAE) of Socher et al. (2011) models the composition of a word pair as a non-linear transformation of their concatenation $[\vec{w}_1; \vec{w}_2]$:

$$\vec{p} = f(M_1[\vec{w}_1; \vec{w}_2] + \vec{b}_1) \quad (4)$$

where $M_1 \in R^{n \times 2n}$. After adding a bias term $\vec{b}_1 \in R^n$, a nonlinear element-wise function f such as $tanh$ is applied to the resulting vector. The representation $\vec{p}$ of the word pair is then fed into a reconstruction layer to reconstruct the two input vectors,

of Chen and Lin (2005) determines that the most two important ones are $RH(w_1, w_2)$ and $KL(w_1, w_2)$.

Methods	Accuracy
Creativity measures (§4.3)	
$\vec{F}_{12}$	62.30
Baseline: vector concatenation (no composition)	
$[\vec{w}_1; \vec{w}_2]$	67.51
Explicit vector composition (§5.1)	
$\vec{w}_1 + \vec{w}_2$	66.62
$abs(\vec{w}_1 - \vec{w}_2)$	60.03
$\min\{\vec{w}_1, \vec{w}_2\}$	66.08
$\max\{\vec{w}_1, \vec{w}_2\}$	64.97
$\vec{w}_1 .* \vec{w}_2$	56.34
$[abs(\vec{w}_1 - \vec{w}_2); \vec{w}_1; \vec{w}_2]$	**69.54**
$[\max\{\vec{w}_1, \vec{w}_2\}; \vec{w}_1; \vec{w}_2]$	68.02
Non-linear composition via kernels (§5.2)	
Polynomial	65.86
RBF	**69.16**
Laplacian	68.15
Non-linear composition via deep learning (§5.3)	
$f(M_1[\vec{w}_1; \vec{w}_2] + \vec{b}_1)$	67.25

Table 5: Performance comparison of creativity classifiers.

Incorrectly predicted word pairs	y^*	Semantically close word pairs	y^*
CONFUSION DUE TO WORD SIMILARITY (20/42)			
"entire carton"	-	*"whole angst"*	+
"outdated tax"	-	*"graconian tax"*	+
"dismissive way"	-	*"amorous way"*	+
"insidious part"	+	*"leather part"*	-
CONFUSION DUE TO SUBJECTIVE LABELING (8/42)			
"independent religion"	+	*"wonderful religion"*	-
WORD SENSE DISAMBIGUATION PROBLEMS (2/42)			
"fiscal cliff"	-	*"winding lake"*	+
"opera window"	+	*"work-shop floor"*	-

Table 6: Error analysis: y^* denotes the true label. For each incorrectly predicted word pair (left column), we show an example of semantically close word pairs (right column) with the opposite true label that might have confused learning.

and a softmax layer to predict the probability of the word pair being creative and not creative. We initialize the word vectors using the pre-learned vector space representations in Huang et al. (2012).

5.4 Experimental Results

Table 5 shows the performance comparison of different features sets and algorithms. In all cases, parameters are tuned from the training portion of the data. We see that simple vector composition alone does not perform better than vector concatenation $[\vec{w}_1; \vec{w}_2]$. However, combining $abs(\vec{w}_1 - \vec{w}_2)$ or $max\{\vec{w}_1; \vec{w}_2\}$ with $[\vec{w}_1; \vec{w}_2]$ perform better than concatenation. Kernels with non-linear transformation of feature space generally improve performance over linear SVM, suggesting that kernels capture some of the interesting compositional aspect of creativity that is not covered by some of the explicit vector compositions considered in §5.1. We also experimented with additional features driven from the creativity measures explored in §4, but we omit their results as those did not help improving the performance. Unfortunately learning nonlinear composition with deep learning did not yield better results. We conjecture that it is due to the small dataset we were able to obtain for this study, which may have not been enough to learn the rich parameter space of the nonlinear transformation matrix.

6 Analysis and Insight

Error analysis We manually inspected a randomly chosen 42 error cases, and characterize the potential causes of those errors. Examples of three types of errors are shown in Table 6. For each incorrectly predicted word pair, we also show a semantically close word pair with the opposite true label that might have confused the learning algorithm.

Visualization To gain additional insight, we project word pairs represented in their vector concatenations onto 2-dimensional space using t-Distributed Stochastic Neighbor Embedding (van der Maaten and Hinton (2008)). Figure 5 shows some of the interesting regions of the projection: some regions are relatively futile in having creative phrases (e.g., regions involving simple adjectives such as *"good"*, *"bad"*, regions corresponding to legal terms), while some regions are relatively more fruitful (e.g., regions involving abstract adjectives such as *"infinite"*, *"universal"*, *"fundamental"*). There are also many other regions (e.g., in the vicinity of *"true"*, *"perfect"* or *"intelligent"* in Figure 5) where the separation between creative and noncreative phrases are not as prominent. In those regions, compositional aspects would play a bigger role in determining creativity than memorizing fruitful semantic subspaces.

Figure 5: Creative (blue bold) and not creative (red italic) word pairs graph.

7 Related Work

Among computational approaches that touch on linguistic creativity, many focused on *metaphor* (e.g., Dunn (2013), Krishnakumaran and Zhu (2007), Mashal et al. (2007), Rumbell et al. (2008), Rentoumi et al. (2012), Mashal et al. (2009)). Other linguistic devices and phenomena related to creativity include *irony* (e.g., Davidov et al. (2010), González-Ibáñez et al. (2011), Filatova (2012)), *neologism* (e.g., Cartoni (2008)), *humor* (e.g., Mihalcea and Strapparava (2005), Purandare and Litman (2006)), and *similes* (e.g., Hao and Veale (2010)).

Veale (2011) proposed the new task of creative text retrieval to harvest expressions that potentially convey the same meaning as the query phrase in a fresh or unusual way. Our work contributes to the retrieval process of recognizing more creative phrases. Ozbal and Strapparava (2012) explored automatic creative naming of commercial products and services, focusing on the generation of creative phrases within a specific domain. Costello (2002) investigated the cognitive process that guides people's choice of words when making up a novel noun-noun compound. In contrast, we present a data-driven investigation to quantifying creativity in lexical composition. Memorability is loosely related to linguistic creativity (Danescu-Niculescu-Mizil et al. (2012)) as some of the creative quotes may be more memorable, but not all creative phrases are memorable and vice versa.

8 Conclusion

We presented the first study that focuses on learning and quantifying creativity in lexical compositions, exploring statistical techniques motivated by three different theories and hypotheses of creativity, ranging from divergent thinking, compositional structure, creative semantic subspace, and the connection to sentiment and connotation. Our experimental results suggest the viability of learning creative language, and point to promising directions for future research.

Acknowledgments This research was supported in part by the Stony Brook University Office of the Vice President for Research, and in part by gift from Google. We thank anonymous reviewers for insightful comments and suggestions.

References

T. Amabile. 1997. Motivating creativity in organizations: On doing what you love and loving what you do. *California Management Review*, 40(1):39–58.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193, Stroudsburg, PA, USA.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram corpus version 1.1. *Google Inc*.

Ronald Carter and Michael McCarthy. 2004. Talking, creating: interactional language, creativity, and context. *Applied Linguistics*, 25(1):62–88.

Bruno Cartoni. 2008. Lexical resources for automatic translation of constructed neologisms: the case study of relational adjectives. In *LREC*.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Yi-Wei Chen and Chih-Jen Lin. 2005. Combining svms with various feature selection strategies. In *Taiwan University*. Springer-Verlag.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*, volume 11. The MIT press.

Carol Chomsky. 1976. Creativity and innovation in child language. *Journal of Education, Boston*.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

W.M. Cohen and D.A. Levinthal. 1990. Absorptive Capacity: A New Perspective on Learning and Innovation. *Administrative Science Quarterly*, 35(1).

Fintan J. Costello. 2002. Investigating creative language: People's choice of words in the production of novel noun-noun compounds. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Arthur Cropley. 2006. In praise of convergent thinking. *Creativity Research Journal*, 18(3):391–404.

Cristian Danescu-Niculescu-Mizil, Justin Cheng, Jon Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 892–901. Association for Computational Linguistics.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Jonathan Dunn. 2013. What metaphor identification systems can tell us about metaphor-in-language. *Meta4NLP 2013*, page 1.

Song Feng, Jun Sak Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, Angust. Association for Computational Linguistics.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *LREC*, pages 392–398.

Stefan L Frank. 2010. Uncertainty reduction as a measure of cognitive processing effort. In *Proceedings of the 2010 workshop on cognitive modeling and computational linguistics*, pages 81–89. Association for Computational Linguistics.

Sigmund Freud. 1908. Creative writers and daydreaming. *Standard edition*, 9:143–153.

Susan R. Fussell and Mallie M. Moss. 1998. Figurative language in descriptions of emotional states. In *Social and cognitive approaches to interpersonal communication*.

Pablo Gervás. 2010. Engineering linguistic creativity: Bird flight and jet planes. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, pages 23–30. Association for Computational Linguistics.

Katie Glaskin. 2011. Dreams, memory, and the ancestors: creativity, culture, and the science of sleep. *Journal of the royal anthropological institute*, 17(1):44–62.

Andrew Goatly. 1997. *The language of metaphors*. Routledge.

Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: A closer look. In *ACL (Short Papers)*, pages 581–586. Citeseer.

Emiliano Guevara. 2011. Computing semantic compositionality in distributional semantics. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, pages 135–144. Citeseer.

John Hale. 2003. The information conveyed by words in sentences. *Journal of Psycholinguistic Research*, 32(2):101–123.

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):643–672.

Yanfen Hao and Tony Veale. 2010. An ironic fist in a velvet glove: Creative mis-representation in the construction of ironic similes. *Minds and Machines*, 20(4):635–650.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational approaches to Figurative Language*, pages 13–20. Association for Computational Linguistics.

George Lakoff and Mark Johnson. 1980. *Metaphors we Live by*. University of Chicago Press, Chicago.

N. Mashal, M. Faust, T Hendler, and M. Jung-Beeman. 2007. An fmri investigation of the neural correlates underlying the processing of novel metaphoric expressions. *Brain and Language*, pages 115 – 126.

N Mashal, M Faust, T Hendler, and M Jung-Beeman. 2009. An fmri study of processing novel metaphoric sentences. *Laterality*, (1):30–54.

Janet Maybin and Joan Swann. 2007. Everyday creativity in language: Textuality, contextuality, and critique. *Applied Linguistics*, 28(4):497–517.

Robert R McCrae. 1987. Creativity, divergent thinking, and openness to experience. *Journal of personality and social psychology*, 52(6):1258.

Susan M. McCurry and Steven C. Hayes. 1992. Clinical and experimental perspectives on metaphorical talk. *Clinical Psychology Review*, 12(7):763 – 785.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *In Proceedings of ACL-08: HLT*, pages 236–244.

Jeff Mitchell and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 430–439. Association for Computational Linguistics.

Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *COLING*, pages 1933–1950.

Edward Necka. 1999. Memory and creativity. *Encyclopedia of creativity, ed. by MA Runco, SR Pritzker*, 2:193–99.

Gozde Ozbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 703–711, Jeju Island, Korea, July. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

J. B. Pollack. 1990. Recursive distributed representation. *Artificial Intelligence*, 46:77–105.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for f* r* i* e* n* d* s*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215. Association for Computational Linguistics.

Vassiliki Rentoumi, George A. Vouros, Vangelis Karkaletsis, and Amalia Moser. 2012. Investigating metaphorical language in sentiment analysis: A sense-to-sentiment perspective. *ACM Trans. Speech Lang. Process.*, 9(3):6:1–6:31, November.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 907–916. Association for Computational Linguistics.

Tim Rumbell, John Barnden, Mark Lee, and Alan Wallington. 2008. Affect in metaphor: Developments with wordnet.

Bernhard Schölkopf and Alexander J Smola. 2002. *Learning with kernels*. The MIT Press.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel methods for pattern analysis*. Cambridge university press.

Ekaterina Shutova. 2010. Models of metaphor in nlp. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 688–697, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-sne.

Tony Veale, Yanfen Hao, and Guofu Li. 2008. Multilingual harvesting of cross-cultural stereotypes. In *ACL*, pages 523–531.

Tony Veale. 2011. Creative language retrieval: A robust hybrid of information retrieval and linguistic creativity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287, Portland, Oregon, USA, June. Association for Computational Linguistics.

Thomas Voegtlin and Peter F. Dominey. 2005. Linear recursive distributed representations. *Neural Netw.*, 18(7):878–895, September.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 34–35. Association for Computational Linguistics.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics.

Xiaojin Zhu, Zhiting Xu, and Tushar Khot. 2009. How creative is your writing? a linguistic creativity measure from computer science and cognitive psychology perspectives. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 87–93. Association for Computational Linguistics.

Sentiment Analysis: How to Derive Prior Polarities from SentiWordNet

Marco Guerini
Trento RISE
Via Sommarive 18
38123 Povo in Trento, Italy
m.guerini@trentorise.eu

Lorenzo Gatti
Trento RISE
Via Sommarive 18
38123 Povo in Trento, Italy
l.gatti@trentorise.eu

Marco Turchi
Fondazione Bruno Kessler
Via Sommarive 18
38123 Povo in Trento, Italy
turchi@fbk.eu

Abstract

Assigning a positive or negative score to a word out of context (i.e. a word's prior polarity) is a challenging task for sentiment analysis. In the literature, various approaches based on SentiWordNet have been proposed. In this paper, we compare the most often used techniques together with newly proposed ones and incorporate all of them in a learning framework to see whether blending them can further improve the estimation of prior polarity scores. Using two different versions of SentiWordNet and testing regression and classification models across tasks and datasets, our learning approach consistently outperforms the single metrics, providing a new state-of-the-art approach in computing words' prior polarity for sentiment analysis. We conclude our investigation showing interesting biases in calculated prior polarity scores when word Part of Speech and annotator gender are considered.

1 Introduction

Many approaches to sentiment analysis make use of lexical resources – i.e. lists of positive and negative words – often deployed as baselines or as features for other methods (usually machine learning based) for sentiment analysis research (Liu and Zhang, 2012). In these lexica, words are associated with their prior polarity, i.e. if that word out of context evokes something positive or something negative. For example, *wonderful* has a positive connotation – prior polarity – while *horrible* has a negative one. These approaches have the advantage of not needing deep semantic analysis or word sense disambiguation to assign an affective score to a word and are domain independent (they are thus less precise but more portable).

SentiWordNet (henceforth SWN) is one of these resources and has been widely adopted since it provides a broad-coverage lexicon – built in a semi-automatic manner – for English (Esuli and Sebastiani, 2006). Given that SWN provides polarities scores for each word sense (also called 'posterior polarities'), it is necessary to derive prior polarities from the posteriors. For example, the word *cold* has a posterior polarity for the meaning "having a low temperature" – like in "*cold* beer" – that is different from the one in "*cold* person" which refers to "being emotionless". This information must be considered when reconstructing the prior polarity of *cold*.

Several formulae to compute prior polarities starting from posterior polarities scores have been used in the literature. However, their performance varies significantly depending on the adopted variant. We show that researchers have not paid sufficient attention to this *posterior-to-prior polarity* issue. Indeed, we show that some variants outperform others on different datasets and can represent a fairer state-of-the-art approach using SWN. On top of this, we attempt to outperform the state-of-the-art formula using a learning framework that combines the various formulae together.

In detail, we will address five main research questions: (*i*) is there any relevant difference in the posterior-to-prior polarity formulae performance (both in regression and classification tasks), (*ii*) is there any relevant variation in prior polarity values

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1259–1269,
Seattle, Washington, USA, 18-21 October 2013.

if we use different releases of SWN (i.e. SWN_1 or SWN_3), (*iii*) can a learning framework boost performance of such formulae, (*iv*) considering word Part of Speech (PoS), is there any relevant difference in formulae performance, (*v*) considering the gender dimension of the annotators (male/female) and the sentiment dimension (positive/negative), is there any relevant difference in SWN performance.

In Section 2 we briefly describe our approach and how it differentiates from similar sentiment analysis tasks. Then, in Sections 3 and 4, we present SentiWordNet and overview various posterior-to-prior polarity formulae based on this resource that appeared in the literature (included some new ones we identified as potentially relevant). In Section 5 we describe the learning approach adopted on prior-polarity formulae. In Section 6 we introduce the ANEW and General Inquirer resources that will be used as gold standards. Finally, in the two last sections, we present a series of experiments, both in regression and classification tasks, that give an answer to the aforementioned research questions. The results support the hypothesis that using a learning framework we can improve on state-of-the-art performance and that there are some interesting phenomena connected to PoS and annotator gender.

2 Proposed Approach

In the broad field of Sentiment Analysis we will focus on the specific problem of posterior-to-prior polarity assessment, using both regression and classification experiments. A general overview on the field and possible approaches can be found in (Pang and Lee, 2008) or (Liu and Zhang, 2012).

For the regression task, we tackled the problem of assigning affective scores (along a continuum between -1 and 1) to words using the posterior-to-prior polarity formulae. For the classification task (assessing whether a word is either *positive* or *negative*) we used the same formulae, but considering just the sign of the result. In these experiments we will also use a learning framework which combines the various formulae together. The underlying hypothesis is that by blending these formulae, and looking at the same information from different perspectives (i.e. the posterior polarities provided by SWN combined in various ways), we can give a better prediction.

The regression task is harder than binary classification, since we want to assess not only that *pretty*, *beautiful* and *gorgeous* are positive words, but also to define a partial or total order so that *gorgeous* is more positive than *beautiful* which, in turn, is more positive than *pretty*. This is fundamental for tasks such as affective modification of existing texts, where words' polarity together with their score are necessary for creating multiple graded variations of the original text (Guerini et al., 2008). Some of the work that addresses the problem of sentiment strength are presented in (Wilson et al., 2004; Paltoglou et al., 2010), however, their approach is modeled as a multi-class classification problem (*neutral*, *low*, *medium* or *high* sentiment) at the sentence level, rather than a regression problem at the word level. Other works such as (Neviarouskaya et al., 2011) use a fine grained classification approach too, but they consider emotion categories (*anger*, *joy*, *fear*, etc.), rather than sentiment strength categories. On the other hand, even if approaches that go beyond pure prior polarities – e.g. using word bigram features (Wang and Manning, 2012) – are better for sentiment analysis tasks, there are tasks that are intrinsically based on the notion of words' prior polarity. Consider copywriting, where evocative names are a key element to a successful product (Özbal and Strapparava, 2012; Özbal et al., 2012). In such cases no context is given and the brand name alone, with its perceived prior polarity, is responsible for stating the area of competition and evoking semantic associations. For example *Mitsubishi* changed the name of one of its SUV for the Spanish market, since the original name *Pajero* had a very negative prior polarity, as it meant 'wanker' in Spanish (Piller, 2003).

To our knowledge, the only work trying to address the SWN posterior-to-prior polarity issue, comparing some of the approaches appeared in the literature is (Gatti and Guerini, 2012). However, in our previous study we only considered a regression framework, we did not use machine learning and we only tested SWN_1. So, we took this work as a starting point for our analysis and expanded on it.

3 SentiWordNet

SentiWordNet (Esuli and Sebastiani, 2006) is a lexical resource in which each entry is a set of

lemma-PoS pairs sharing the same meaning, called "synset". Each synset `s` is associated with the numerical scores `Pos(s)` and `Neg(s)`, which range from 0 to 1. These scores – automatically assigned starting from a bunch of seed terms – represent the positive and negative valence (or posterior polarity) of the synset and are inherited by each lemma-PoS in the synset. According to the structure of SentiWordNet, each pair can have more than one sense and each of them takes the form of `lemma#PoS#sense-number`, where the smallest sense-number corresponds to the most frequent sense.

Obviously, different senses can have different polarities. In Table 1, the first 5 senses of `cold#a` present all possible combinations, included mixed scores (`cold#a#4`), where positive and negative valences are assigned to the same sense. Intuitively, mixed scores for the same sense are acceptable, as in "*cold* beer" (positive) vs. "*cold* pizza" (negative).

PoS	Offset	Pos(s)	Neg(s)	SynsetTerms
a	1207406	0.0	0.75	`cold#a#1`
a	1212558	0.0	0.75	`cold#a#2`
a	1024433	0.0	0.0	`cold#a#3`
a	2443231	0.125	0.375	`cold#a#4`
a	1695706	0.625	0.0	`cold#a#5`

Table 1: First five *SentiWordNet* entries for `cold#a`

In our experiments we use two different versions of SWN: SentiWordNet 1.0 (SWN_1), the first release of SWN, and its updated version SentiWordNet 3.0 (Baccianella et al., 2010) – SWN_3. In SWN_3 the annotation algorithm used in SWN_1 was revised, leading to an increase in the accuracy of posterior polarities over the previous version.

4 Prior Polarities Formulae

In this section we review the main strategies for computing prior polarities used in previous studies. All the proposed approaches try to estimate the prior polarity score from the posterior polarities of all the senses for a single lemma-PoS. Given a lemma-PoS with n senses (`lemma#PoS#n`), every formula f is independently applied to all the `Pos(s)` and `Neg(s)` . This produces two scores, $f(posScore)$ and $f(negScore)$, for each lemma-PoS. To obtain a unique prior polarity for each lemma-PoS, $f(posScore)$ and $f(negScore)$ can be mapped according to different strategies:

$$f_m = \begin{cases} f(posScore) & \text{if } f(posScore) \geq f(negScore) \\ -f(negScore) & \text{otherwise} \end{cases}$$

$$f_d = f(posScore) - f(negScore)$$

where f_m computes the absolute maximum of the two scores, while f_d computes the difference between them. It is worth noting that $f(negScore)$ is always positive by construction. To obtain a final prior polarity that ranges from -1 to 1, the negative sign is imposed. So, considering the first 5 senses of `cold#a` in Table 1, $f(posScore)$ will be derived from the `Pos(s)` values $<0.0, 0.0, 0.0, 0.125, 0.625>$, while $f(negScore)$ from $<0.750, 0.750, 0.0, 0.375, 0.0>$. Then, the final polarity strength returned will be either f_m or f_d.

The formulae (f) we tested are the following:

fs. In this formula only the first (and thus most frequent) sense is considered for the given `lemma#PoS`. This is equivalent to considering only the SWN score for `lemma#PoS#1`. Based on (Neviarouskaya et al., 2009; Agrawal and Siddiqui, 2009; Guerini et al., 2008; Chowdhury et al., 2013), this is the most basic form of prior polarities.

mean. It calculates the mean of the positive and negative scores for all the senses of the given `lemma#PoS`. This formula has been used in (Thet et al., 2009; Denecke, 2009; Devitt and Ahmad, 2007; Sing et al., 2012).

uni. Based on (Neviarouskaya et al., 2009), it considers only those senses that have a `Pos(s)` greater than or equal to the corresponding `Neg(s)`, and greater than 0 (the $stronglyPos$ set). In case $posScore$ is equal to $negScore$, the one with the highest weight is returned, where weights are defined as the cardinality of $stronglyPos$ divided by the total number of senses. The same applies for the negative senses. This is the only method, together with rnd, for which we cannot apply f_d, as it returns a positive or negative score according to the weight.

uniw. Like uni but without the weighting system.

w1. This formula weighs each sense with a geometric series of ratio 1/2. The rationale behind this choice is based on the assumption that more frequent

senses should bear more "affective weight" than rare senses when computing the prior polarity of a word. The system presented in (Chaumartin, 2007) uses a similar approach of weighted mean.

w2. Similar to the previous one, this formula weighs each lemma with a harmonic series, see for example (Denecke, 2008).

On top of these formulae, we implemented some new formulae that were relevant to our task and have not been implemented before. These formulae mimic the ones discussed previously, but they are built under a different assumption: that the saliency (Giora, 1997) of a word's prior polarity might be more related to its posterior polarities score, rather than to sense frequencies. Thus we ordered $posScore$ and $negScore$ by strength, giving more relevance to 'valenced' senses. For instance, in Table 1, $posScore$ and $negScore$ for `cold#a` become $<0.625, 0.125, 0.0, 0.0, 0.0>$ and $<0.750, 0.750, 0.375, 0.0, 0.0>$ respectively.

w1s and **w1n**. Like $w1$ and $w2$, but senses are ordered by strength (sorting Pos(s) and `Neg(s)` independently).

w1n and **w2n**. Like $w1$ and $w2$ respectively, but without considering senses that have a 0 score for both `Pos(s)` and `Neg(s)`. Our motivation is that "empty" senses are mostly noise.

w1sn and **w2sn**. Like $w1s$ and $w2s$, but without considering senses that have a 0 score for both `Pos(s)` and `Neg(s)`.

median: return the median of the senses ordered by polarity score.

All these prior polarities formulae are compared against two gold standards (one for regression, one for classification) both one by one, as in the works mentioned above, and combined together in a learning framework (to see whether combining these features – that capture different aspect of prior polarities – can further improve the results).

Finally, we implemented two variants of a prior polarity random baseline to asses possible advantages of approaches using SWN:

rnd. This formula represents the basic baseline random approach. It simply returns a random number between -1 and 1 for any given `lemma#PoS`.

swnrnd. This formula represents an advanced random approach that incorporates some "knowledge" from SWN. It takes the scores of a random sense for the given `lemma#PoS`. We believe this is a fairer baseline than rnd since SWN information can possibly constrain the values. A similar approach has been used in (Qu et al., 2008).

5 Learning Algorithms

We used two non-parametric learning approaches, Support Vector Machines (SVMs) (Shawe-Taylor and Cristianini, 2004) and Gaussian Processes (GPs) (Rasmussen and Williams, 2006), to test the performance of all the metrics in conjunction. SVMs are non-parametric deterministic algorithms that have been widely used in several fields, in particular in NLP where they are the state-of-the-art for various tasks. GPs, on the other hand, are an extremely flexible non-parametric probabilistic framework able to explicitly model uncertainty, that, despite being considered state-of-the-art in regression, have rarely been used in NLP. To our knowledge only two previous works did so (Polajnar et al., 2011; Cohn and Specia, 2013).

Both methods take advantage of the *kernel trick*, a technique used to embed the original feature space into an alternative space where data may be linearly separable. This is performed by the kernel function that transforms the input data in a new structure, called *kernel*. How it is used to produce the prediction is one of the main differences between SVMs and GPs. In classification SVMs use the geometric mean to discriminate between the positive and negative classes, while the GP model uses the posterior probability distribution over each class. Both frameworks support learning algorithms for regression and classification. An exhaustive explanation of the two methodologies can be found in (Shawe-Taylor and Cristianini, 2004) and (Rasmussen and Williams, 2006).

In the SVM experiments, we use C-SVM and ϵ-SVM implemented in the LIBSVM toolbox (Chang and Lin, 2011). The selection of the kernel (linear, polynomial, radial basis function and sigmoid) and the optimization of the parameters are carried out through grid search in 10-fold cross-validation.

GP regression models with Gaussian noise are a rare exception where the exact inference with likelihood functions is tractable, see §2 in (Rasmussen

and Williams, 2006). Unfortunately, this is not valid for the classification task – see §3 in (Rasmussen and Williams, 2006) – where an approximation method is required. In this work, we use the Laplace approximation method proposed in (Williams and Barber, 1998). Different kernels are tested (covariance for constant functions, linear with and without automatic relevance determination (ARD)[1], Matern, neural network, etc.[2]) and the linear logistic (lll) and probit regression (prl) likelihood functions are evaluated in classification. In our classification experiments we tried all possible combinations of kernels and likelihood functions, while in the regression tests we ranged only on different kernels. All the GP models were implemented using the GPML Matlab toolbox[3]. Unlike SVMs, the optimization of the kernel parameters can be performed without using grid search, but the optimal parameters can be obtained iteratively, by maximizing the marginal likelihood (or in classification, the Laplace approximation of the marginal likelihood). We fix at 100 the maximum number of iterations.

An interesting property of the GPs is their capability of weighting the features differently according to their importance in the data. This is referred to as the automatic variance determination kernel. As demonstrated in (Weston et al., 2000), SVMs can benefit from the application of feature selection techniques especially when there are highly redundant features. Since the prior polarities formulae tend to cluster in groups that provide similar results (Gatti and Guerini, 2012) – creating noise for the learner – we want to understand whether feature selection approaches can boost the performance of SVMs. For this reason, we also test feature selection prior to the SVM training. For that we used Randomized Lasso, or stability selection (Meinshausen and Bühlmann, 2010). Re-sampling of the training data is performed several times and a Lasso regression model is fit on each sample. Features that appear in a given number of samples are retained. Both the fraction of the data to be sampled and the threshold to select the features can be configured. In our experiments we set the sampling fraction to 75%, the selection threshold to 25% and the number of resamples to 1,000. We refer to these as *SVMfs*.

[1] $linone$ and $linard$ in the result tables, respectively.

[2] More detailed information on the available kernels are in §4 (Rasmussen and Williams, 2006)

[3] `http://www.gaussianprocess.org/gpml/code/matlab/doc/`

6 Gold Standards

To assess how well prior polarity formulae perform, a gold standard with word polarities provided by human annotators is needed. There are many such resources in the literature, each with different coverage and annotation characteristics. ANEW (Bradley and Lang, 1999) rates the valence score of 1,034 words, which were presented in isolation to annotators. The SO-CAL entries (Taboada et al., 2011) were collected from corpus data and then manually tagged by a small number of annotators with a multi-class label. These ratings were further validated through crowdsourcing. Other resources, such as the General Inquirer lexicon (Stone et al., 1966), provide a binomial classification (either *positive* or *negative*) of sentiment-bearing words. The resource presented in (Wilson et al., 2005) uses a similar binomial annotation for single words; another interesting resource is WordNetAffect (Strapparava and Valitutti, 2004) but it labels words senses and it cannot be used for the prior polarity validation task.

In the following we describe in detail the two resources we used for our experiments, namely ANEW for the regression experiments and the General Inquirer (GI) for the classification ones.

6.1 ANEW

ANEW (Bradley and Lang, 1999) is a resource developed to provide a set of normative emotional ratings for a large number of words (roughly 1 thousand) in the English language. It contains a set of words that have been rated in terms of pleasure (affective valence), arousal, and dominance. In particular for our task we considered the valence dimension. Since words were presented to subjects in isolation (i.e. no context was provided) this resource represents a human validation of prior polarities scores for the given words, and can be used as a gold standard. For each word ANEW provides two main metrics: $anew_{\mu}$, which correspond to the average of annotators votes, and $anew_{\sigma}$, which gives the variance in annotators scores for the given word. In the same way these metrics are also provided for

the male/female annotator groups.

6.2 General Inquirer

The Harvard General Inquirer dictionary is a widely used resource, built for automatic text analysis (Stone et al., 1966). Its latest revision[4] contains 11789 words, tagged with 182 semantic and pragmatic labels, as well as with their part of speech. Words and their categories were initially taken from the Harvard IV-4 Psychosociological Dictionary (Dunphy et al., 1974) and the Lasswell Value Dictionary (Lasswell and Namenwirth, 1969). For this paper we consider the `Positiv` and `Negativ` categories (1,915 words the former, 2,291 words the latter, for a total of 4,206 affective words).

7 Experiments

In order to use the ANEW dataset to measure prior polarities formulae performance, we had to assign a PoS to all the words to obtain the SWN `lemma#PoS` format. To do so, we proceeded as follows: for each word, check if it is present among both SWN_1 and SWN_3 lemmas; if not, lemmatize the word with the TextPro tool suite (Pianta et al., 2008) and check if the lemma is present instead[5]. If it is not found (i.e., the word cannot be aligned automatically), remove the word from the list (this was the case for 30 words of the 1,034 present in ANEW). The remaining 1,004 lemmas were then associated with all the PoS present in SWN to get the final `lemma#PoS`. Note that a lemma can have more than one PoS, for example, *writer* is present only as a noun (`writer#n`), while *yellow* is present as a verb, a noun and an adjective (`yellow#v`, `yellow#n`, `yellow#a`). This gave us a list of 1,484 words in the `lemma#PoS` format.

In a similar way we pre-processed the GI words that uses the generic `modif` label to indicate either adjective or adverb (`noun` and `verb` PoS were instead consistently used). Finally, all the sense-disambiguated words in the `lemma#PoS#n` format were discarded (1,114 words out of the 4,206 words with positive or negative valence).

After the two datasets were built this way, we removed the words for which the $posScore$ and $negScore$ contained all 0 in both SWN_1 and SWN_3 (523 `lemma#PoS` for ANEW and 484 for the GI dataset), since these words are not informative for our experiments. The final dataset included 961 entries for ANEW and 2,557 for GI. For each `lemma#PoS` in GI and ANEW, we then applied the prior polarity formulae described in Section 4, using both SWN_1 and SWN_3 and annotated the results.

According to the nature of the human labels (real numbers or -1/1), we ran several regression and classification experiments. In both cases, each dataset was randomly split into 70% for training and the remaining for test. This process was repeated 5 times to generate different splits. For each partition, optimization of the learning algorithm parameters was performed on the training data (in 10-fold cross-validation for SVMs). Training and test sets were normalized using the z-score.

To evaluate the performance of our regression experiments on ANEW we used the Mean Absolute Error (MAE), that averages the error over a given test set. Accuracy was used for the classification experiments on GI instead. We opted for accuracy – rather than F1 – since for us True Negatives have same importance as True Positives. For each experiments we reported the average performance and the standard deviation over the 5 random splits. In the following sections, to check if there was a statistically significant difference in the results, we used Student's t-test for regression experiments, while an approximate randomization test (Yeh, 2000) was used for the classification experiments.

In Tables 2 and 3, the results of regression experiments over the ANEW dataset, using SWN_1 and SWN_3, are presented. The results of the classification experiments over the GI dataset, using SWN_1 and SWN_3 are shown in Tables 4 and 5. For the sake of interpretability, results are divided according to the main approaches: randoms, posterior-to-prior formulae, learning algorithms. Note that for classification we report the generics f and not the f_m and f_d variants. In fact, both versions always return the same classification answer (we are classifying according to the sign of f result and not its strength). For the GPs, we report the two best configurations only.

[4] `http://www.wjh.harvard.edu/~inquirer/`

[5] We did not lemmatize everything to avoid duplications (for example, if we lemmatize the ANEW entry *addicted*, we obtain *addict*, which is already present in ANEW).

	MAE μ	MAE σ
rnd	0.652	0.026
$swnrnd_m$	0.427	0.011
$swnrnd_d$	0.426	0.009
$uniw_m$	0.420	0.009
max_m	0.419	0.009
fs_d	0.413	0.011
fs_m	0.412	0.009
uni	0.410	0.010
$uniw_d$	0.406	0.007
$w1sn_m$	0.405	0.011
max_d	0.404	0.005
$w2sn_m$	0.402	0.011
$median_d$	0.401	0.014
$w1_d$	0.401	0.010
$w1n_d$	0.399	0.008
$mean_d$	0.398	0.010
$w2_d$	0.398	0.010
$median_m$	0.397	0.015
$w1sn_d$	0.397	0.008
$w2sn_d$	0.397	0.008
$w2n_d$	0.397	0.008
$w1s_m$	0.396	0.010
$w1_m$	0.396	0.010
$w1n_m$	0.394	0.009
$mean_m$	0.393	0.011
$w2s_d$	0.393	0.008
$w1s_d$	0.393	0.009
$w2s_m$	0.392	0.010
$w2_m$	0.391	0.011
$w2n_m$	0.391	0.012
GP_{linard}	0.398	0.014
GP_{linone}	0.398	0.014
SVM	0.367	0.010
$SVMfs$	0.366	0.011
AVERAGE	0.398	0.010

Table 2: MAE results for metrics using SWN_1

	MAE μ	MAE σ
rnd	0.652	0.026
$swnrnd_d$	0.404	0.013
$swnrnd_m$	0.402	0.010
max_m	0.393	0.009
fs_d	0.382	0.008
$uniw_m$	0.382	0.015
fs_m	0.381	0.010
$median_m$	0.377	0.008
$uniw_d$	0.377	0.012
$median_d$	0.377	0.011
uni	0.376	0.010
max_d	0.372	0.011
$mean_d$	0.371	0.010
$w1sn_m$	0.371	0.011
$w2sn_m$	0.369	0.010
$w1_d$	0.368	0.010
$w2_d$	0.367	0.010
$mean_m$	0.367	0.010
$w1_m$	0.365	0.010
$w2sn_d$	0.364	0.011
$w1sn_d$	0.364	0.010
$w1s_m$	0.363	0.009
$w1n_d$	0.362	0.009
$w2s_d$	0.362	0.010
$w2_m$	0.362	0.010
$w1s_d$	0.362	0.009
$w1n_m$	0.362	0.007
$w2n_d$	0.361	0.010
$w2s_m$	0.360	0.009
$w2n_m$	0.359	0.009
GP_{linone}	0.356	0.008
GP_{linard}	0.355	0.008
SVM	0.333	0.004
$SVMfs$	0.333	0.003
AVERAGE	0.366	0.009

Table 3: MAE results for regression using SWN_3

8 General Discussion

In this section we sum up the main results of our analysis, providing an answer to the various questions we introduced at the beginning of the paper:

SentiWordNet improves over random. One of the first things worth noting – in Tables 2, 3, 4 and 5 – is that the random approach (*rnd*), as expected, is the worst performing metric, while all other approaches, based on *SWN*, have statistically significant improvements both for MAE and for Accuracy ($p < 0.001$). So, using *SWN* for posterior-to-prior polarity computation brings benefits, since it increases the performance above the baseline in words' prior polarity assessment.

SWN$_3$ is better than SWN$_1$. With respect to SWN_1, using SWN_3 enhances performance, both in regression (MAE μ 0.398 vs. 0.366, $p < 0.001$) and classification (Accuracy μ 0.710 vs. 0.771, $p < 0.001$) tasks. Since many of the approaches described in the literature use SWN_1 their results should be revised and SWN_3 should be used as standard. This difference in performance can be partially explained by the fact that, even after preprocessing, for the ANEW dataset 137 `lemma#PoS` have all senses equal to 0 in SWN_1, while in SWN_3 they are just 48. In the GI lexicon the numbers are 233 for SWN_1 and 69 for SWN_3.

Not all formulae are created equal. The formulae described in Section 4 have very different results, along a continuum. While inspecting every differ-

	Acc. μ	Acc. σ
rnd	0.447	0.019
swn_rnd_m	0.639	0.026
swn_rnd_d	0.646	0.021
fs_m	0.659	0.020
uni	0.684	0.017
$median$	0.686	0.022
$uniw$	0.702	0.019
max	0.710	0.022
$w1$	0.712	0.021
$w1n$	0.713	0.022
$w2n$	0.714	0.023
$w2$	0.715	0.021
$mean$	0.718	0.023
$w2s$	0.719	0.023
$w2sn$	0.719	0.023
$w1s$	0.719	0.023
$w1sn$	0.719	0.023
GP^{lll}_{linard}	0.721	0.026
GP^{prl}_{linard}	0.722	0.025
SVM	0.733	0.021
$SVMfs$	0.743	0.021
Average	0.710	0.022

Table 4: Accuracy results for classification using SWN_1

	Acc. μ	Acc. σ
rnd	0.447	0.019
swn_rnd_d	0.700	0.030
swn_rnd_m	0.706	0.034
fs	0.723	0.014
$medianm$	0.742	0.016
uni	0.750	0.015
$uniw$	0.762	0.023
max	0.769	0.019
$w2s$	0.777	0.017
$w2sn$	0.777	0.017
$w1s$	0.777	0.017
$w1sn$	0.777	0.017
$w1n$	0.780	0.021
$w2n$	0.780	0.022
$mean$	0.781	0.018
$w1$	0.781	0.021
$w2$	0.781	0.021
SVM	0.779	0.016
GPl	0.779	0.018
GPg	0.781	0.018
$SVMfs$	0.792	0.014
Average	0.771	0.018

Table 5: Accuracy results for classification using SWN_3

ence in performance is out of the scope of the present paper, we can see that there is a strong difference between best and worst performing formulae both in regression (in Table 2 $w2n_m$ is better than $uniw_m$, in Table 3 $w2n_m$ is better than max_m) and classification (in Table 4 $w1sn_m$ is better than fs_m,in Table 5 $w2_m$ is better than fs_m) and these differences are all statistically significant ($p < 0.001$). Again, these results indicate that the previous experiments in the literature that use SWN as a baseline should be revised to take these results into account. Furthermore, the new formulae we introduced, based on the "posterior polarities saliency" hypothesis, proved to be among the best performing in all experiments. This entails that there is room for inspecting new formulae variants other than those already proposed in the literature.

Selecting just one sense is not a good choice. On a side note, the approaches that rely on only one sense polarity (namely fs, $median$ and max) have similar results which do not differ significantly from $swnrnd$ (for max_m, fs_d and fs_m in Table 2, and for max_m in Table 3). These same approaches are also far from the best performing formulae: in Table 3, $median_d$ differs from $w2n_m$ ($p < 0.05$), as do max_m, max_d, fs_m and fs_d ($p < 0.001$); in Table 3, fs, max and $median$ in both their f_m and f_d variants are significantly different from the best performing $w2n_m$ ($p < 0.001$). For classification, in Table 4 and 5 the difference between the corresponding best performing formula and the single senses formulae is always significant (at least $p < 0.01$). Among other things, this finding entails, surprisingly, that taking the first sense of a `lemma#PoS` in some cases has no improvement over taking a random sense, and that in all cases it is one of the worst approaches with SWN. This is surprising since in many NLP tasks, such as word sense disambiguation, algorithms based on most frequent sense represent a very strong baseline[6].

Learning improvements. Combining the formulae in a learning framework further improves the results over the best performing formulae, both in regression (MAEμ with SWN_1 0.366 vs. 0.391, $p < 0.001$; MAEμ with SWN_3 0.333 vs. 0.359, $p < 0.001$) and in classification (Accuracyμ for SWN_1 is 0.743 vs. 0.719, $p < 0.001$; Accuracyμ for SWN_3 is 0.792 vs. 0.781, not significant $p = 0.07$). Another thing worth noting is that, in regression, GPs are outperformed by both versions of

[6]In SemEval 2010, only 5 participants out of 29 performed better than the most frequent threshold (Agirre et al., 2010).

SVM ($p < 0.001$), see Tables 2 and 3. This is in contrast with the results presented in (Cohn and Specia, 2013), where GPs on the single task are on average better than SVMs. In classification, GPs have similar performance to SVM without feature selection, and in some cases (see Table 5) even slightly better. Analyzing the selected kernels for GPs and SVMs, we notice that in most of the splits SVMs prefer the radial based function, while the best performance with the GPs are obtained with linear kernels with and without ARD. There is no significant difference in using linear logistic and probit regression likelihoods. In all our experiments, SVM with feature selection leads to the best performance. This is not surprising due the high level of redundancy in the formulae scores. Interestingly, inspecting the most frequent selected features by $SVMfs$, we see that features from different groups are selected, and even the worst performing formulae can add information, confirming the idea that viewing the same information from different perspectives (i.e. the posterior polarities provided by SWN combined in various ways) can give better predictions.

To sum up: the new state-of-the-art performance level in prior-polarity computation is represented by the $SVMfs$ approach using SWN_3, and this should be used as the reference from now on.

9 PoS and Gender Experiments

Next, we wanted to understand if the performance of our approach, using SWN_3, was consistent across word PoS. In Table 6 we report the results for the best performing formulae and learning algorithm on the GI PoS classes. In particular for ADJ there are 1,073 words, 922 for NOUN and 508 for VERB. We discarded adverbs since the class was too small to allow reliable evaluation and efficient learning (only 54 instances). The results show a greater accuracy for adjectives ($p < 0.01$), while performance for nouns and verbs are similar.

	SVMfs		$best\_f$	
	Acc. μ	Acc. σ	Acc. μ	Acc. σ
ADJ	0.829	0.019	0.821	0.016
NOUN	0.784	0.021	0.765	0.023
VERBS	0.782	0.052	0.744	0.046

Table 6: Accuracy results for PoS using SWN_3

Finally we test against the male and female ratings provided by ANEW. As can be seen from Table 7, SWN approaches are far more precise in predicting Male judgments rather than Female ones (MAEμ goes from 0.392 to 0.323 with the best formula and from 0.369 to 0.292 with $SVMfs$, both differences are significant $p < 0.001$). Instead, in Table 8 – which displays the results along gender and polarity dimensions – there is no statistically significant difference in MAE on positive words between male and female, while there is a strong statistical significance for negative words ($p < 0.001$).

Interestingly, there is also a large difference between positive and negative affective words (both for male and female dimensions). This difference is maximum for male scores on positive words compared to female scores on negative words (0.283 vs. 0.399, $p < 0.001$). Recent work by Warriner et al. (2013) inspected the differences in prior polarity assessment due to gender.

At this stage we can only note that prior polarities calculated with SWN are closer to ANEW male annotations than female ones. Understanding why this happens would require an accurate examination of the methods used to create WordNet and SWN (which will be the focus of our future work).

	Male		female	
	MAE μ	MAE σ	MAE μ	MAE σ
SVMfs	0.292	0.020	0.369	0.008
best_f	0.323	0.022	0.392	0.010

Table 7: MAE results for Male vs Female using SWN_3

	Male		female	
	MAE μ	MAE σ	MAE μ	MAE σ
Pos	0.283	0.022	0.340	0.009
Neg	0.301	0.029	0.399	0.013

Table 8: MAE for Male/Female - Pos/Neg using SWN_3

10 Conclusions

We have presented a study on the posterior-to-prior polarity issue, i.e. the problem of computing words' prior polarity starting from their posterior polarities. Using two different versions of SentiWordNet and 30 different approaches that have been proposed in the literature, we have shown that researchers have not paid sufficient attention to this issue. Indeed, we

showed that the better variants outperform the others on different datasets both in regression and classification tasks, and that they can represent a fairer state-of-art baseline approach using SentiWordNet. On top of this, we also showed that these state-of-the-art formulae can be further outperformed using a learning framework that combines the various formulae together. We conclude our analysis with some experiments investigating the impact of word PoS and annotator gender in gold standards, showing interesting phenomena that requires further investigation.

Acknowledgments

The authors thanks José Camargo De Souza for his help with feature selection. This work has been partially supported by the Trento RISE PerTe project.

References

E. Agirre, O.L. De Lacalle, C. Fellbaum, S.K. Hsieh, M. Tesconi, M. Monachini, P. Vossen, and R. Segers. 2010. Semeval-2010 task 17: All-words word sense disambiguation on a specific domain. In *Proceedings of the 5th International Workshop on Semantic Evaluation (IWSE '10)*, pages 75–80, Uppsala, Sweden.

S. Agrawal and T.J. Siddiqui. 2009. Using syntactic and contextual information for sentiment polarity analysis. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09)*, pages 620–623, Seoul, Republic of Korea.

S. Baccianella, A. Esuli, and F. Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC '10)*, pages 2200–2204, Valletta, Malta.

M.M. Bradley and P.J. Lang. 1999. Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical Report C-1, University of Florida.

C.C. Chang and C.J. Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

F.R. Chaumartin. 2007. UPAR7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the 4th International Workshop on Semantic Evaluations (IWSE '07)*, pages 422–425, Prague, Czech Republic.

F.M. Chowdhury, M. Guerini, S. Tonelli, and A. Lavelli. 2013. Fbk: Sentiment analysis in twitter with tweetsted. In *Second Joint Conference on Lexical and Computational Semantics (*SEM): Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval '13)*, volume 2, pages 466–470, Atlanta, Georgia, USA, June.

T. Cohn and L. Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL '13)*, pages 32–42, Sofia, Bulgaria.

K. Denecke. 2008. Accessing medical experiences and information. In *Proceedings of the 18th European Conference on Artificial Intelligence, Workshop on Mining Social Data (MSoDa '08)*, Patras, Greece.

K. Denecke. 2009. Are SentiWordNet scores suited for multi-domain sentiment classification? In *Proceedings of the 4th International Conference on Digital Information Management (ICDIM '09)*, pages 32–37, Ann Arbor, MI, USA.

A. Devitt and K. Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 984–991, Prague, Czech Republic.

D.C. Dunphy, C.G. Bullard, and E.E.M. Crossing. 1974. Validation of the General Inquirer Harvard IV Dictionary. Paper presented at the Pisa Conference on Content Analysis.

A. Esuli and F. Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on International Language Resources and Evaluation (LREC '06)*, pages 417–422, Genova, Italy.

L. Gatti and M. Guerini. 2012. Assessing sentiment strength in words prior polarities. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 361–370, Mumbai, India.

R. Giora. 1997. Understanding figurative and literal language: The graded salience hypothesis. *Cognitive Linguistics*, 8:183–206.

M. Guerini, O. Stock, and C. Strapparava. 2008. Valentino: A tool for valence shifting of natural language texts. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC '08)*, pages 243–246, Marrakech, Morocco.

H.D. Lasswell and J.Z. Namenwirth. 1969. The Lasswell value dictionary. *New Haven*.

B. Liu and L. Zhang. 2012. A survey of opinion mining and sentiment analysis. *Mining Text Data*, pages 415–463.

N. Meinshausen and P. Bühlmann. 2010. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473.

A. Neviarouskaya, H. Prendinger, and M. Ishizuka. 2009. Sentiful: Generating a reliable lexicon for sentiment analysis. In *Proceedings of the 3rd Affective Computing and Intelligent Interaction (ACII '09)*, pages 363—368, Amsterdam, Netherlands.

A. Neviarouskaya, H. Prendinger, and M. Ishizuka. 2011. Affect analysis model: novel rule-based approach to affect sensing from text. *Natural Language Engineering*, 17(1):95.

G. Özbal and C. Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*, pages 703–711, Jeju Island, Korea.

G. Özbal, C. Strapparava, and M. Guerini. 2012. Brand Pitt: A corpus to explore the art of naming. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC '12)*, pages 1822–1828, Istanbul, Turkey.

G. Paltoglou, M. Thelwall, and K. Buckley. 2010. Online textual communications annotated with grades of emotion strength. In *Proceedings of the 3rd International Workshop of Emotion: Corpora for research on Emotion and Affect (satellite of LREC '10)*, pages 25–31, Valletta, Malta.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

E. Pianta, C. Girardi, and R. Zanoli. 2008. The TextPro tool suite. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC '08)*, pages 2603–2607, Marrakech, Morocco.

I. Piller. 2003. Advertising as a site of language contact. *Annual Review of Applied Linguistics*, 23:170–183.

T. Polajnar, S. Rogers, and M. Girolami. 2011. Protein interaction detection in sentences via gaussian processes: a preliminary evaluation. *International journal of data mining and bioinformatics*, 5(1):52–72.

L. Qu, C. Toprak, N. Jakob, and I. Gurevych. 2008. Sentence level subjectivity and sentiment analysis experiments in NTCIR-7 MOAT challenge. In *Proceedings of the 7th NTCIR Workshop Meeting (NTCIR '08)*, pages 210–217, Tokyo, Japan.

C.E. Rasmussen and C.K.I. Williams. 2006. *Gaussian processes for machine learning*. MIT Press.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel methods for pattern analysis*. Cambridge university press.

J.K. Sing, S. Sarkar, and T.K. Mitra. 2012. Development of a novel algorithm for sentiment analysis based on adverb-adjective-noun combinations. In *Proceedings of the 3rd National Conference on Emerging Trends and Applications in Computer Science (NCETACS '12)*, pages 38–40, Shillong, India.

P.J. Stone, D.C. Dunphy, and M.S. Smith. 1966. *The General Inquirer: A Computer Approach to Content Analysis.* MIT press.

C. Strapparava and A. Valitutti. 2004. WordNet-Affect: an affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC '04)*, pages 1083 – 1086, Lisbon, Portugal.

M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

T.T. Thet, J.C. Na, C.S.G. Khoo, and S. Shakthikumar. 2009. Sentiment analysis of movie reviews on discussion boards using a linguistic approach. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion (TSA '09)*, pages 81–84, Hong Kong.

S. Wang and C.D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*, pages 90–94, Jeju Island, Korea.

A.B. Warriner, V. Kuperman, and M. Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, pages 1–17.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. 2000. Feature selection for SVMs. In *Proceedings of the 14th Conference on Neural Information Processing Systems (NIPS '00)*, pages 668–674, Denver, CO, USA.

C.K.I. Williams and D. Barber. 1998. Bayesian classification with gaussian processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(12):1342–1351.

T. Wilson, J. Wiebe, and R. Hwa. 2004. Just how mad are you? Finding strong and weak opinion clauses. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI '04)*, pages 761–769, San Jose, CA, USA.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP '05)*, pages 347–354, Vancouver, Canada.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING '00)*, pages 947–953, Saarbrücken, Germany.

Simulating Early-Termination Search for Verbose Spoken Queries

Jerome White
IBM Research
Bangalore, KA India
jerome.white@in.ibm.com

Douglas W. Oard
University of Maryland
College Park, MD USA
oard@umd.edu

Nitendra Rajput
IBM Research
New Delhi, India
rnitendra@in.ibm.com

Marion Zalk
University of Melbourne
Melbourne, VIC Australia
m.zalk@student.unimelb.edu.au

Abstract

Building search engines that can respond to spoken queries with spoken content requires that the system not just be able to find useful responses, but also that it know when it has heard enough about what the user wants to be able to do so. This paper describes a simulation study with queries spoken by non-native speakers that suggests that indicates that finding relevant content is often possible within a half minute, and that combining features based on automatically recognized words with features designed for automated prediction of query difficulty can serve as a useful basis for predicting when that useful content has been found.

1 Introduction

Much of the early work on what has come to be called "speech retrieval" has focused on the use of text queries to rank segments that are automatically extracted from spoken content. While such an approach can be useful in a desktop environment, half of the world's Internet users can access the global information network only using a voice-only mobile phone. This raises two challenges: 1) in such settings, both the query and the content must be spoken, and 2) the language being spoken will often be one for which we lack accurate speech recognition.

The Web has taught us that the "ten blue links" paradigm can be a useful response to short queries. That works because typed queries are often fairly precise, and tabular responses are easily skimmed. However, spoken queries, and in particular open-domain spoken queries for unrestricted spoken content, pose new challenges that call for new thinking about interaction design. This paper explores the potential of a recently proposed alternative, in which the spoken queries are long, and only one response can be played at a time by the system. This approach, which has been called Query by Babbling, requires that the user ramble on about what they are looking for, that the system be able to estimate when it has found a good response, and that the user be able to continue the search interaction by babbling on if the first response does not fully meet their needs (Oard, 2012).

One might question whether users actually will "babble" for extended periods about their information need. There are two reasons to believe that some users might. First, we are particularly interested in ultimately serving users who search for information in languages for which we do not have usable speech recognition systems. Speech-to-speech matching in such cases will be challenging, and we would not expect short queries to work well. Second, we seek to principally serve users who will be new to search, and thus not yet conditioned to issue short queries. As with Web searchers, we can expect them to explore initially, then to ultimately settle on query strategies that work well enough to meet their needs. If longer queries work better for them, it seems reasonable to expect that they would use longer queries. Likewise, if systems cannot effectively use longer queries to produce useful results, then people will not use them.

To get a sense for whether such an interaction modality is feasible, we performed a simulation

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1270–1280,
Seattle, Washington, USA, 18-21 October 2013.

study for this paper in which we asked people to babble on some topic for which we already have relevance judgments results. We transcribe those babbles using automatic speech recognition (ASR), then note how many words must be babbled in each case before an information retrieval system is first able to place a relevant document in rank one. From this perspective, our results show that people are indeed often able to babble usefully; and, moreover, that current information retrieval technology could often place relevant results at rank one within half a minute or so of babbling even with contemporary speech recognition technology.

The question then arises as to whether a system can be built that would recognize when an answer is available at rank one. Barging in with an answer before that point wastes time and disrupts the user; barging in long after that point also wastes time, but also risks user abandonment. We therefore want a "Goldilocks" system that can get it just about right. To this end, we introduce an evaluation measure that differentially penalizes early and late responses. Our experiments using such a measure show that systems can be built that, on average, do better than could be achieved by any fixed response delay.

The remainder of this paper is organized as follows: We begin in Section 2 with a brief review of related work. Section 3 then describes the design of the ranking component of our experiment; Section 4 follows with some exploratory analysis of the ranking results using our test collection. Section 6 completes the description of our methods with an explanation of how the stopping classifier is built; Section 7 then presents end-to-end evaluation results using a new measure designed for this task. Section 8 concludes the paper with some remarks on future work.

2 Background

The rapid adoption of remarkably inexpensive mobile telephone services among low-literacy users in developing and emerging markets has generated considerable interest in so-called "spoken forum" projects (Sherwani et al., 2009; Agarwal et al., 2010; Medhi et al., 2011; Mudliar et al., 2012). It is relatively straightforward to collect and store spoken content regardless of the language in which it is spoken; organizing and searching that content is, however, anything but straightforward. Indeed, the current lack of effective search services is one of the key inhibitors that has, to date, limited spoken forums to experimental settings with at most a few hundred users. If a "spoken web" is to achieve the same degree of impact on the lives of low-literacy users in the developing world that the World Wide Web has achieved over the past decade in the developed world, we will need to develop the same key enabler: an effective search engine.

At present, spoken dialog systems of conventional design, such as Siri, rely on complex and expensive language-specific engineering, which can easily be justified for the "languages of wealth" such as English, German, and Chinese; but perhaps not for many of the almost 400 languages that are each spoken by a million or more people.[1] An alternative would be to adopt more of an "information retrieval" perspective by directly matching words spoken in the query with words that had been spoken in the content to be searched. Some progress has been made on this task in the MediaEval benchmark evaluation, which has included a spoken content matching task each year since 2011 (Metze et al., 2012). Results for six low-resource Indian and African languages indicate that miss rates of about 0.5 can be achieved on individual terms, with false alarm rates below 0.01, by tuning acoustic components that had originally been developed for languages with reasonably similar phonetic inventories. Our goal in this paper is to begin to explore how such capabilities might be employed in a complete search engine for spoken forum content, as will be evaluated for the first time at MediaEval 2013.[2] The principal impediment to development in this first year of that evaluation is the need for relevance judgments, which are not currently available for spoken content of the type we wish to search. That consideration has motivated our design of the simulation study reported in this paper.

[1] http://www.ethnologue.com/statistics/size

[2] http://www.multimediaeval.org/mediaeval2013/qa4sw2013/

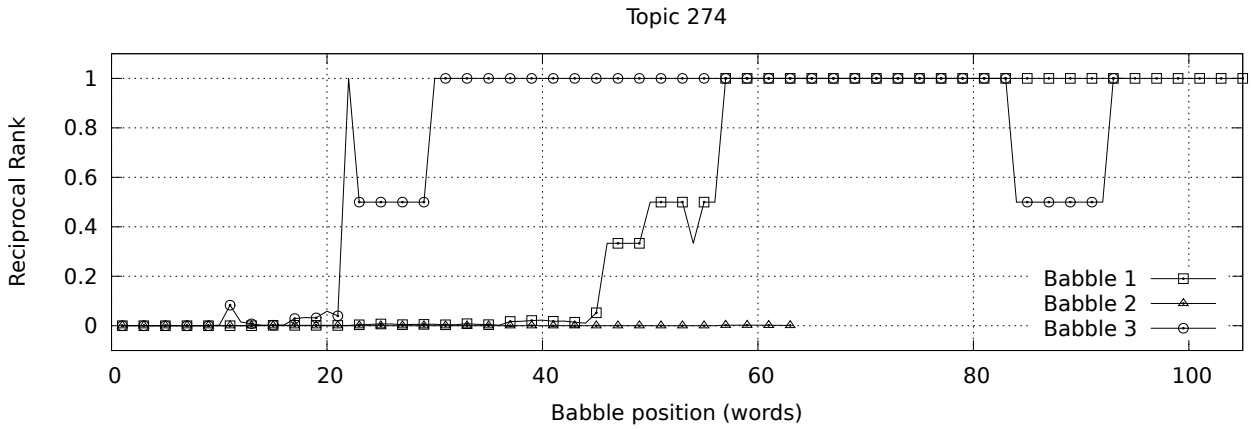


Figure 1: Reciprocal ranks at for each query making up a given babble. When retrieving results, a babbler either "latches" on to a relevant document (Babble 1), moves back-and-forth between relevant documents (Babble 3), or fails to elicit a relevant document at all (Babble 2).

3 Setup and Method

The approach taken in this paper is to simulate, as closely as possible, babbling about topics for which we a) already have relevance judgments available, and b) have the ability to match partial babbles with potential answers in ways that reflect the errors introduced by speech processing. To this end, we chose to ask non-native English speakers to babble, in English, about an information need that is stimulated by an existing English Text Retrieval Conference (TREC) topic for which we already have relevance judgments. An English Automatic Speech Recognition (ASR) system was then used to generate recognized words for those babbles. Those recognized words, in turn, have been used to rank order the (character-coded written text) news documents that were originally used in TREC, the documents for which we have relevance judgments. Our goal then becomes twofold: to first rank the documents in such a way as to get a relevant document into rank one; and then to recognize when we have done so.

Figure 1 is a visual representation of retrieval results as a person babbles. For three different babbles prompted by TREC Topic 274, it shows the reciprocal rank for the query that is posed after each additional word is recognized. We are primarily interested in cases where the reciprocal rank is one.[3] In these three babbles we see all cases that the retrieval system must take into account: babbles that never yield a relevant first-ranked document (Babble 2); babbles that eventually yield a relevant first-rank document, and that continue to do so as the person speaks (Babble 1); and babbles that alternate between good and bad results as the speaker continues (Babble 3).

3.1 Acquiring Babbles

Ten TREC-5 Ad Hoc topics were selected for this study: 255, 257, 258, 260, 266, 271, 274, 276, 287, and 297 based on our expectation of which of the 50 TREC 5 topics would be most suitable for prompted babbles. In making this choice, we avoided TREC topics that we felt would require specialized domain knowledge, experience with a particular culture, or detailed knowledge of an earlier time period, such as when the topics had been crafted. For each topic, three babbles were created by people speaking at length about the same information need that the TREC topic reflected. For convenience, the people who created the babbles were second-language speakers of English selected from information technology companies. There were a total of ten babblers; each recorded, in English, babbles for three topics, yielding a total of thirty babbles. We maintained a balance across topics when assigning topic

[3] A reciprocal rank of one indicates that a known relevant document is in position one; a reciprocal rank of 0.5 indicates that the most highly ranked known relevant document is in position two; 0.33 indicates position three; and so on.

Transcribed babble	Text from ASR
So long time back one of my friend had a Toyota Pryus it uses electric and petrol to increase the to reduce the consumption and increase the mileage I would now want to get information about why car operators manufacturers or what do they think about electric vehicles in the US well this is what the stories say that the car lobby made sure that the electric vehicles do not get enough support and the taxes are high by the government but has it changed now are there new technologies that enable to lower cost and also can increase speed for electric vehicles I am sure something is being done because of the rising prices of fuel these days	So long time at one of my friends headed towards the previous accuses electric in petrol to increase the to reduce the consumption and increase the minutes and would now want to get information about why car operator manufacturers on what to think about electric vehicles in the us versus what the story said that the car lobby make sure that the electric vehicles to not get enough support to an attack and I try to comment but has changed now arctic new technologies that enabled to cover costs and also can increase speak for electric vehicles I'm sure some clinton gore carls junior chef

Table 1: Text from an example babble (274-1). The left is transcribed through human comprehension; the right is the output from an automatic speech recognition engine.

numbers to babblers. All babblers had more than sixteen years of formal education, had a strong command on the English language, and had some information about the topics that they selected. They were all briefed about our motivation for collecting this data, and about the concept of query by babbling.

The babbles were created using a phone interface. Each subject was asked to call an interactive voice response (IVR) system. The system prompted the user for a three digit topic ID. After obtaining the topic ID, the system then prompted the user to start speaking about what they were looking for. TREC topics contain a short title, a description, and a narrative. The title is generally something a user might post as an initial Web query; the description is something one person might say to another person who might then help them search; the narrative is a few sentences meant to reflect what the user might jot down as notes to themselves on what they were actually looking for. For easy reference, the system provided a short description—derived from the description and narrative of the TREC topics—that gave the user the context around which to speak. The user was expected to begin speaking after hearing a system-generated cue, at which time their speech was recorded. Two text files were produced from the audio babbles: one produced via manual transcription,[4] and one produced by an ASR system; Table 1 presents an example. The ASR transcripts of the babbles were used by our system as a basis for ranking, and as a basis for making the decision on when to barge-in, what we call the "stopping point." The manual transcriptions were used only for scoring the Word Error Rate (WER) of the ASR transcript for each babble.

TREC Topic		WER	
ID	Title	Mean	SD
255	Environmental protect.	0.434	0.203
257	Cigarette consumption	0.623	0.281
258	Computer security	0.549	0.289
260	Evidence of human life	0.391	0.051
266	Prof. scuba diving	0.576	0.117
271	Solar power	0.566	0.094
274	Electric automobiles	0.438	0.280
276	School unif./dress code	0.671	0.094
287	Electronic surveillance	0.519	0.246
297	Right to die pros/cons	0.498	0.181
	Average	0.527	0.188

Table 2: Average ASR Word Error Rate over 3 babbles per topic (SD=Standard Deviation).

[4] The transcriber is the third author of this paper.

Babble	Words	Judgment at First Rank			Scorable	First Rel	Last Rel	WER
		Relevant	Not Relevant	Unknown				
257-3	74	5	64	5	93%	@13	@66	0.414
276-3	61	7	46	8	87%	@36	@42	0.720
258-1	146	2	118	26	82%	@28	@29	0.528
297-1	117	58	19	40	66%	@56	@117	0.594
274-3	94	57	0	47	61%	@22	@94	0.250
274-1	105	49	13	43	59%	@57	@105	0.437
257-1	191	104	0	87	54%	@52	@188	0.764
271-1	145	42	26	76	48%	@38	@109	0.556
287-2	61	26	0	35	43%	@33	@61	0.889
260-2	93	22	8	63	32%	@69	@93	0.500
276-2	69	11	2	56	19%	@47	@69	0.795
260-3	82	6	8	68	17%	@17	@62	0.370
258-2	94	14	1	79	16%	@24	@60	0.389
297-3	90	4	2	84	7%	@52	@56	0.312
266-2	115	6	0	109	5%	@47	@52	0.745

Table 3: Rank-1 relevance ("Rel") judgments and position of first and last scorable guesses.

3.2 System Setup

The TREC-5 Associated Press (AP) and Wall Street Journal (WSJ) news stories were indexed by Indri (Strohman et al., 2004) using the Krovetz stemmer (Krovetz, 1993), standard English stopword settings, and language model matching. Each babble was turned into a set of nested queries by sequentially concatenating words. Specifically, the first query contained only the first word from the babble, the second query only the first two words, and so on. Thus, the number of queries presented to Indri for a given babble was equivalent to the number of words in the babble, with each query differing only by the number of words it contained. The results were scored using `trec_eval` version 9.0. For evaluation, we were interested in the reciprocal rank; in particular, where the reciprocal rank was one. This measure tells us when Indri was able to place a known relevant document at rank one.

4 Working with Babbles

Our experiment design presents three key challenges. The first is ranking well despite errors in speech processing. Table 2 shows the average Word Error Rate (WER) for each topic, over three babbles. Averaging further over all thirty babbles, we see that about half the words are correctly recognized. While this may seem low, it is in line with observations from other spoken content retrieval research: over classroom lectures (Chelba et al., 2007), call center recordings (Mamou et al., 2006), and conversational telephone speech (Chia et al., 2010). Moreover, it is broadly consistent with the reported term-matching results for low density languages in MediaEval.

The second challenge lies in the scorability of the system guesses. Table 3 provides an overview of where relevance was found within our collection of babbles. It includes only the subset of babbles for which, during the babble, at least one known relevant document was found at the top of the ranked list. The table presents the number of recognized words—a proxy for the number of potential stopping points—and at how many of those potential stopping points the document ranked in position 1 is known to be relevant, known not to be relevant, or of unknown relevance. Because of the way in which TREC relevance judgments were created, unknown relevance indicates that no TREC system returned the document near the top of their ranked list. At TREC, documents with unknown relevance are typ-

ically scored as if they are not relevant;[5] we make the same assumption.

Table 3 also shows how much we would need to rely on that assumption: the "scorable" fraction for which the relevance of the top-ranked document is known, rather than assumed, ranges from 93 per cent down to 5 per cent. In the averages that we report below, we omit the five babbles with scorable fractions of 30 per cent or less. On average, over the 10 topics for which more than 30 per cent of the potential stopping points are scorable, there are 37 stopping points at which our system could have been scored as successful based on a known relevant document in position 1. In three of these cases, the challenge for our stopping classifier is extreme, with only a handful—between two and seven—of such opportunities.

A third challenge is knowing when to interrupt to present results. The ultimate goal of our work is to predict when the system should interrupt the babbler and barge-in to present an answer in which they might be interested. Table 3 next presents the word positions at which known relevant documents first and last appear in rank one ("First Rel"). This are the earliest and latest scorable successful stopping points. As can be seen, the first possible stopping point exhibits considerable variation, as does the last. For some babbles—babble 274-3, for example—almost any choice of stopping points would be fine. In other cases—babble 258-1, for example—a stopping point prediction would need to be spot on to get any useful results at all. Moreover, we can see both cases in different babbles for the same topic despite the fact that both babblers were prompted by the same topic; for example, babbles 257-1 and 257-3, which are, respectively, fairly easy and fairly hard.

Finally, we can look for interaction effects between speech processing errors and scorability. The rightmost column of Table 3 shows the measured WER for each scorable babble. Of the 10 scorable babbles for which more than 30 per cent of the potential stopping points are scorable, three turned out to be extremely challenging for ASR, with word error rates above 0.7. Overall, however, the WER for the 10 babbles on which we focus is 0.56, which is about the same as the average WER over all 30 babbles.

In addition to the 15 babbles shown in Table 3, there are another 15 babbles for which no relevant document was retrievable. Of those, only a single babble—babble 255-2, at 54 per cent scorable and a WER of 0.402—had more than 30 per cent of the potential stopping points scorable.

5 Learning to Stop

There are several ways in which we could predict when to stop the search and barge-in with an answer—in this paper, we consider a machine learning approach. The idea is that by building a classifier with enough information about known good and bad babbles, a learner can make such predictions better than other methods. Our stopping prediction models uses four types of features for each potential stopping point: the number of words spoken so far, the average word length so far, some "surface characteristics" of those words, and some query performance prediction metrics. The surface characteristics that we used were originally developed to quantify writing style—they are particularly useful for generating readability grades of a given document. Although many metrics for readability have been proposed, we choose a subset: Flesch Reading Ease (Flesch, 1948), Flesch-Kincaid Grade Level (Kincaid et al., 1975), Automated Readability Index (Senter and Smith, 1967), Coleman-Liau index (Coleman and Liau, 1975), Gunning fog index (Gunning, 1968), LIX (Brown and Eskenazi, 2005), and SMOG Grading (McLaughlin, 1969). Our expectation was that a better readability value should correspond to use of words that are more succinct and expressive, and that a larger number of more expressive words should help the search engine to get good responses highly ranked.

As post-retrieval query difficulty prediction measures, we choose three that have been prominent in information retrieval research: clarity (Cronen-Townsend et al., 2002), weighted information gain (Zhou and Croft, 2007), and normalized query commitment (Shtok et al., 2012). Although each takes a distinct approach, the methods all compare some aspect of the documents retrieved by a query

[5] On the assumption that the TREC systems together span the range of responses that are likely to be relevant.

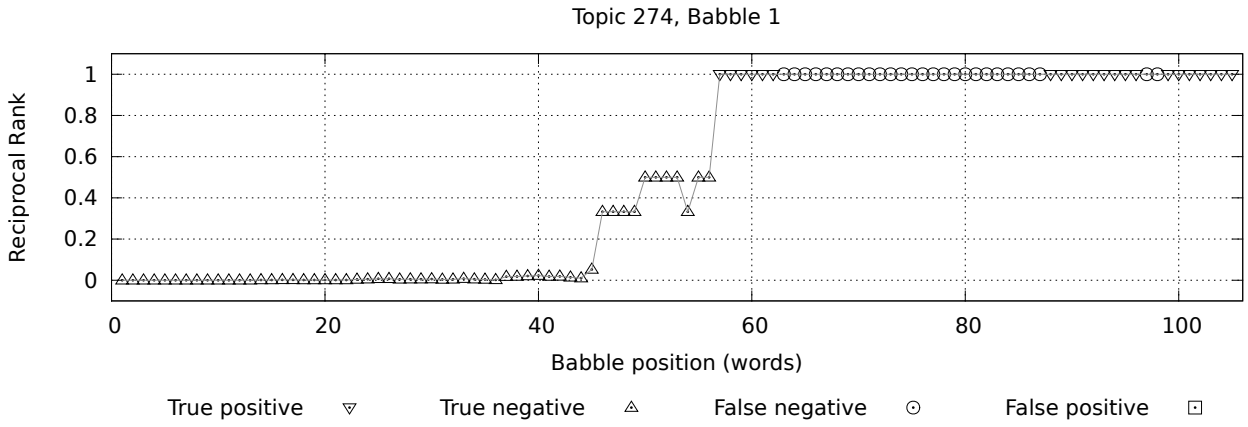


Figure 2: Predictions for babble 274-1 made by a decision tree classifier trained on 27 babbles for the nine other topics. For each point, the mean reciprocal rank is annotated to indicate the correctness of the guess made by the classifier. Note that in this case, the classifier never made a false positive. See Figure 1 for an unannotated version of this same babble.

	Confusion Matrix					
Class.	T_n	F_p	F_n	T_p	F_1	Acy.
Bayes	1288	1259	61	291	0.31	55%
Reg.	2522	25	253	99	0.42	90%
Trees	2499	48	70	282	0.83	96%

Table 4: Cross validation accuracy ("Acy.") measures for stop-prediction classifiers: naive Bayes, logistic regression, and Decision trees.

with the complete collection of documents in the collection from which that retrieval was performed. They seek to provide some measure of information about how likely a query is to have ranked the documents well when relevance judgments are not available. Clarity measures the difference in the language models induced by the retrieved results and the corpus as a whole. Weighted information gain and normalized query commitment look at the scores of the retrieved documents, the former comparing the mean score of the retrieved set with that of the entire corpus; the latter measuring the standard deviation of the scores for the retrieved set.

Features of all four types were were created for each query that was run for each babble; that is after receiving each new word. A separate classifier was then trained for each topic by creating a binary objective function for all 27 babbles for the nine other topics, then using every query for every one of those babbles as training instances. The objective function produces 1 if the query actually retrieved a relevant document at first rank, and 0 otherwise. Figure 2 shows an example of how this training data was created for one babble, and Table 4 shows the resulting hold-one-topic-out cross-validation results for intrinsic measures of classifier accuracy for three Weka classifiers[6]. As can be seen, the decision tree classifier seems to be a good choice, so in Section 7 we compare the stopping prediction model based on a decision tree classifier trained using hold-one-topic-out cross-validation with three baseline models.

6 Evaluation Design

This section describes our evaluation measure and the baselines to which we compared.

6.1 Evaluation Measure

To evaluate a stopping prediction model, the fundamental goal is to stop with a relevant document in rank one, and to do so as close in time as possible to the first such opportunity. If the first guess is bad, it would be reasonable to score a second guess, with some penalty.

Specifically, there are several things that we

[6]Naive Bayes, logistic regression, and decision trees (J48)

would like our evaluation framework to describe. Keeping in mind that ultimately the system will interrupt the speaker to notify them of results, we first want to avoid the interruption before we have found a good answer. Our evaluation measure gives no credit for such a guess. Second, we want to avoid interrupting long after finding the first relevant answer. Credit is reduced with increasing delays after the first point where we could have barged in. Third, when we do barge-in, there must indeed be a good answer in rank one. This will be true if we barge-in at the first opportunity, but if we barge-in later the good answer we had found might have dropped back out of the first position. No credit is given if we barge-in such a case. Finally, if a bad position for first barge-in is chosen, we would like at least to get it right the second time. Thus, we limit ourselves to two tries, awarding half the credit on the second try that we could have received had we barged in at the same point on the first try.

The delay penalty is modeled using an exponential distribution that declines with each new word that arrives after the first opportunity. Let q_0 be the first point within a query where the reciprocal rank is one. Let p_i be the first "yes" guess of the predictor after point q_0. The score is thus $e^{\lambda(q_0-p_i)}$, where λ is the half-life, or the number of words by which the exponential decay has dropped to one-half. The equation is scaled by 0.5 if i is the second element (guess) of p, and by 0.25 if it is the third. From Figure 1, some cases the potential stopping points are consecutive, while in others they are intermittent—we penalize delays from the first good opportunity even when there is no relevant document in position one because we feel that best models the user experience. Unjudged documents in position one are treated as non-relevant.

6.2 Stopping Prediction Baselines

We chose one deterministic and one random baseline for comparison. The deterministic baseline made its first guess at a calculated point in the babble, and continued to guess at each word thereafter. The initial guess was determined by taking the average of the first scorable point of the other 27 out-of-topic babbles.

The random baseline drew the first and second words at which to guess "yes" as samples from a uniform distribution. Specifically, drawing samples uniformly, without replacement, across the average number of words in all other out-of-topic babbles.

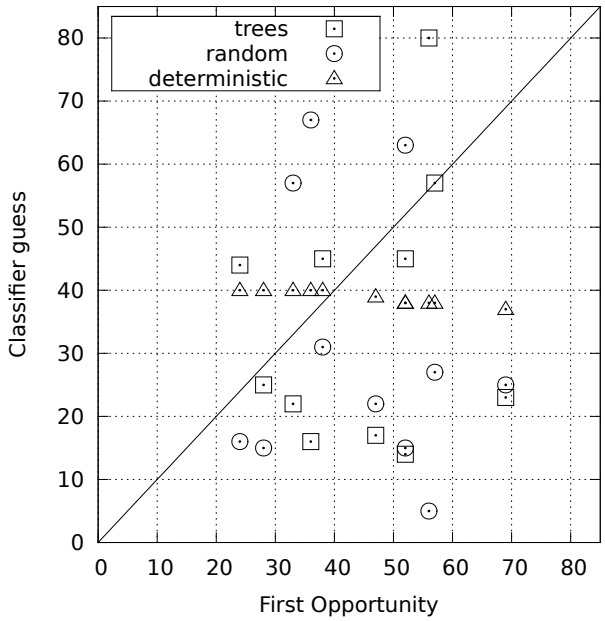


Figure 3: First guesses for various classifiers plotted against the first instance of rank one documents within a babble. Points below the diagonal are places where the classifier guessed too early; points above are guesses too late. All 11 babbles for which the decision tree classifier made a guess are shown.

7 Results

Figure 3 shows the extent to which each classifiers first guess is early, on time, or late. These points falls, respectively, below the main diagonal, on the main diagonal, or above the main diagonal. Early guesses result in large penalties from our scoring function, dropping the maximum score from 1.0 to 0.5; for late guesses the penalty depends on how late the guess is. As can be seen, our decision tree classifier ("trees") guesses early more often than it guesses late. For an additional four cases (not plotted), the decision tree classifier never makes a guess.

Figure 4 shows the results for scoring at most three guesses. These results are averaged over all eleven babbles for which the decision tree classifier made at least one guess; no guess was made on babbles 257-3, 266-2, 260-3, or 274-3. These re-

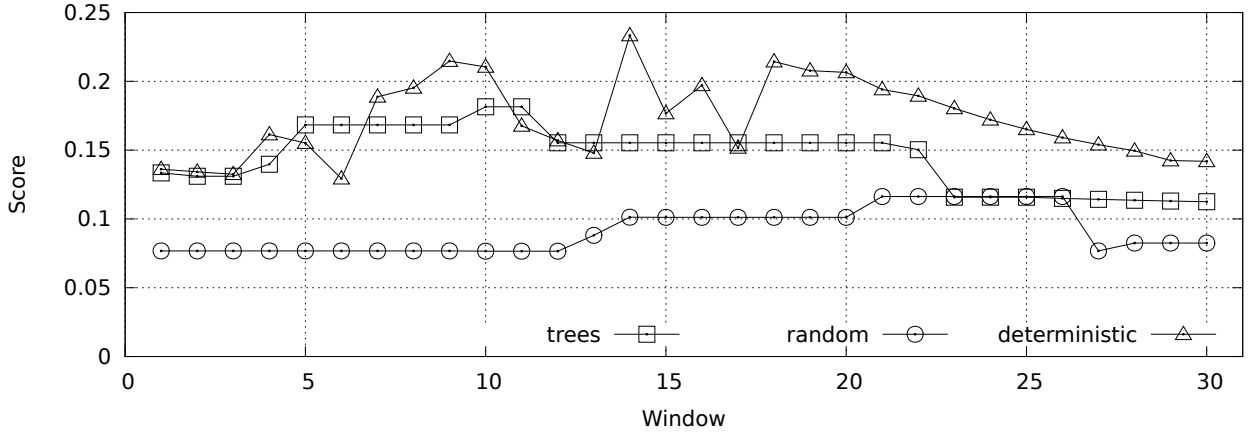


Figure 4: Evaluation using all available babbles in which the tree classifier made a guess.

sults are shown for a half-life of five words, which is a relatively steep penalty function, essentially removing all credit after about ten seconds at normal speaking rates. The leftmost point in each figure, plotted at a "window size" of one, shows the results for the stopping prediction models as we have described them. It is possible, and indeed not unusual, for our decision tree classifier to make two or three guesses in a row, however, in part because it has no feature telling it how long is has been since its most recent guess. To see whether adding a bit of patience would help, we added a deterministic period following each guess in which no additional guess would be allowed. We call the point at which this delay expires, and a guess is again allowed, the delay "window."

As can be seen, a window size of ten or eleven—allowing the next guess no sooner than the tenth or eleventh subsequent word—is optimal for the decision tree classifier when averaged over these eleven babbles. The random classifier has an optimal point between window sizes of 21 and 26, but is generally not as good as the other classifiers. The deterministic classifier displays the most variability, but for window sizes greater than 14, it is the best solution. Although it has fewer features available to it—knowing only the mean number of words to the first opportunity for other topics—it is able to outperform the decision tree classifier for relatively large window sizes.

From this analysis we conclude that our decision tree classifier shows promise; and that going forward, it would likely be beneficial to integrate features of the deterministic classifier. We can also conclude that these results are, at best, suggestive—a richer test collection will ultimately be required. Moreover, we need some approach to accommodate the four cases in which the decision tree classifier never guesses. Setting a maximum point at which the first guess will be tried could be a useful initial heuristic, and one that would be reasonable to apply in practice.

8 Conclusions and Future Work

We have used a simulation study to show that building a system for query by babbling is feasible. Moreover, we have suggested a reasonable evaluation measure for this task, and we have shown that several simple baselines for predicting stopping points can be beaten by a decision tree classifier. Our next step is to try these same techniques with spoken questions and spoken answers in a low-resource language using the test collection that is being developed for the MediaEval 2013 Question Answering for the Spoken Web task.

Another potentially productive direction for future work would be to somehow filter the queries in ways that improve the rankings. Many potential users of this technology in the actual developing region settings that we wish to ultimately serve will likely have no experience with Internet search engines, and thus they may be even less likely to fo-

cus their babbles on useful terms to the same extent that our babblers did in these experiments. There has been some work on techniques for recognizing useful query terms in long queries, but of course we will need to do that with spoken queries, and moreover with queries spoken in a language for which we have at lest limited speech processing capabilities available. How best to model such a situation in a simulation study is not yet clear, so we have deferred this question until the MediaEval speech-to-speech test collection becomes available.

In the long term, many of the questions we are exploring will also has implications for open-domain Web search in other hands- or eyes-free applications such as driving a car or operating an aircraft.

Acknowledgments

We thank Anna Shtok for her assistance with the understanding and implementation of the various query prediction metrics. We also thank the anonymous babblers who provided data that was imperative to this study. Finally, we would like to thank the reviewers, whose comments helped to improve the work overall.

References

[Agarwal et al.2010] Sheetal K. Agarwal, Anupam Jain, Arun Kumar, Amit A. Nanavati, and Nitendra Rajput. 2010. The spoken web: A web for the underprivileged. *SIGWEB Newsletter*, pages 1:1–1:9, June.

[Brown and Eskenazi2005] Jonathan Brown and Maxine Eskenazi. 2005. Student, text and curriculum modeling for reader-specific document retrieval. In *Proceedings of the IASTED International Conference on Human-Computer Interaction. Phoenix, AZ*.

[Chelba et al.2007] Ciprian Chelba, Jorge Silva, and Alex Acero. 2007. Soft indexing of speech content for search in spoken documents. *Computer Speech and Language*, 21(3):458–478.

[Chia et al.2010] Tee Kiah Chia, Khe Chai Sim, Haizhou Li, and Hwee Tou Ng. 2010. Statistical lattice-based spoken document retrieval. *ACM Transactions on Information Systems*, 28(1):2:1–2:30, January.

[Coleman and Liau1975] Meri Coleman and TL Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

[Cronen-Townsend et al.2002] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 299–306, New York, NY, USA. ACM.

[Flesch1948] Rudolf Flesch. 1948. A new readability yardstick. *The Journal of applied psychology*, 32(3):221.

[Gunning1968] Robert Gunning. 1968. *The technique of clear writing*. McGraw-Hill New York.

[Kincaid et al.1975] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

[Krovetz1993] Robert Krovetz. 1993. Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 191–202, New York, NY, USA. ACM.

[Mamou et al.2006] Jonathan Mamou, David Carmel, and Ron Hoory. 2006. Spoken document retrieval from call-center conversations. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 51–58, New York, NY, USA. ACM.

[McLaughlin1969] G Harry McLaughlin. 1969. Smog grading: A new readability formula. *Journal of reading*, 12(8):639–646.

[Medhi et al.2011] Indrani Medhi, Somani Patnaik, Emma Brunskill, S.N. Nagasena Gautama, William Thies, and Kentaro Toyama. 2011. Designing mobile interfaces for novice and low-literacy users. *ACM Transactions on Computer-Human Interaction*, 18(1):2:1–2:28.

[Metze et al.2012] Florian Metze, Etienne Barnard, Marelie Davel, Charl Van Heerden, Xavier Anguera, Guillaume Gravier, Nitendra Rajput, et al. 2012. The spoken web search task. In *Working Notes Proceedings of the MediaEval 2012 Workshop*.

[Mudliar et al.2012] Preeti Mudliar, Jonathan Donner, and William Thies. 2012. Emergent practices around cgnet swara, voice forum for citizen journalism in rural india. In *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, ICTD '12, pages 159–168, New York, NY, USA. ACM.

[Oard2012] Douglas W. Oard. 2012. Query by babbling. In *CIKM Workshop on Information and Knowledge Management for Developing Regions*, October.

[Senter and Smith1967] RJ Senter and EA Smith. 1967. Automated readability index. Technical report, DTIC Document.

[Sherwani et al.2009] Jahanzeb Sherwani, Sooraj Palijo, Sarwat Mirza, Tanveer Ahmed, Nosheen Ali, and Roni Rosenfeld. 2009. Speech vs. touch-tone: Telephony interfaces for information access by low literate users. In *International Conference on Information and Communication Technologies and Development*, pages 447–457.

[Shtok et al.2012] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems*, 30(2):11:1–11:35, May.

[Strohman et al.2004] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. 2004. Indri: A language model-based search engine for complex queries. In *International Conference on Intelligence Analysis*.

[Zhou and Croft2007] Yun Zhou and W. Bruce Croft. 2007. Query performance prediction in web search environments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 543–550, New York, NY, USA. ACM.

Summarizing Complex Events: a Cross-modal Solution of Storylines Extraction and Reconstruction

Shize Xu xsz@pku.edu.cn
Shanshan Wang cheers_echo_mch@163.com
Yan Zhang* zhy@cis.pku.edu.cn

Department of Machine Intelligence, Peking University, Beijing, China
Key Laboratory on Machine Perception, Ministry of Education, Beijing, China

Abstract

The rapid development of Web2.0 leads to significant information redundancy. Especially for a complex news event, it is difficult to understand its general idea within a single coherent picture. A complex event often contains branches, intertwining narratives and side news which are all called storylines. In this paper, we propose a novel solution to tackle the challenging problem of storylines extraction and reconstruction. Specifically, we first investigate two requisite properties of an ideal storyline. Then a unified algorithm is devised to extract all effective storylines by optimizing these properties at the same time. Finally, we reconstruct all extracted lines and generate the high-quality story map. Experiments on real-world datasets show that our method is quite efficient and highly competitive, which can bring about quicker, clearer and deeper comprehension to readers.

1 Introduction

News reports usually consist of various modalities of tremendous information, especially all kinds of textual information and visual information, which make web users dazzled and lost. The situation gets worse on complex news events. To help readers quickly grasp the general information of the news, a more concise and convenient system over multi-modality information should be provided. For example, given a large collection of texts and images related to a specified news event (e.g., *East Japan Earthquake*), such a system should present a terse and brief summarization about the event by showing different clues of its development, and thus helping readers to effectively find out "when, where, what, how and why" at a glance.

The researches (Goldstein et al., 2000) on automatic multi-document summarization (MDS) have helped a lot when we generate a description for a specific event. However, it traditionally exhibits in a very simple style like a "0-dimensional" point. The appearance of *Timeline* (Allan et al., 2001) brings about a visual progress for massive documents analyses. Readers can not only get the most important ideas, but also browse the story evolution in chronological order. Previous news summarization systems with structured output (Yan et al., 2011) have focused on timeline generation. Timeline becomes a "1-dimensional" line. This style of summarization only works for simple stories, which are linear in nature. However, the structure of complex stories usually turns out to be non-linear. These stories branch into storylines, dead ends, intertwining narratives and side news. To explore these lines, we need a map to reorganize all the information. Therefore, a "2-dimensional" story map is in bad need. Figure 1 shows a part of the story map generated by our system for representing *East Japan Earthquake*. We notice that the whole event evolves into 4 branches. Each of them focuses on a specific sub-topic and is distinct from other lines. Figure 2 takes a close look at the 4 nodes from different lines, and they differ a lot from each other as expected.

Text information is more precise and exquisite when compared with images. Nevertheless, as the

*Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1281–1291,
Seattle, Washington, USA, 18-21 October 2013.

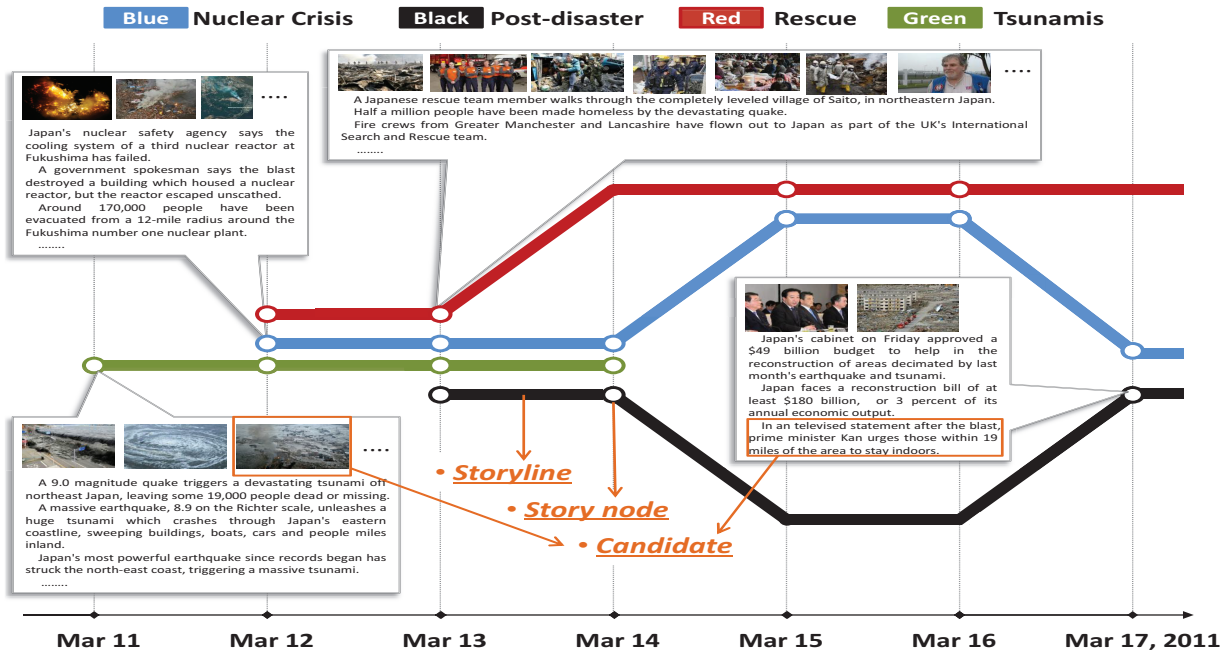


Figure 1: Four storylines are obtained in the story map of "East Japan Earthquake". They focus on *Tsunamis*, *Nuclear Crisis*, *Rescue* and *Post-disaster* respectively.

saying goes, "a picture paints a thousand words", an image could provide far more information than words do. In fact, a summarization including both texts and images will absolutely yield a more powerful and intuitive description about the news event. Under this motivation, we study on extracting and reconstructing tracks with different sub-topics for a complex event. To the best of our knowledge, the exploration and analysis of 2-dimensional cross-modal summarization is academically novel.

We are faced with two main problems. The first is how to select the most important sentences and images to make up the final story map. The previous work by Shahaf et al. presents a 2-D story map called "metro map", which summarizes the complex topics (Shahaf et al., 2012). They study on the document-level, and use the entire news document as one story node. But on real web, this may confront some difficulties. On one hand, news articles may report the event from different perspectives, especially those reviews or retrospective reports. This kind of documents contains many useful saliency information of different sub-topics, but they cannot be further subdivided to help understand each better. On the other hand, some documents, such as

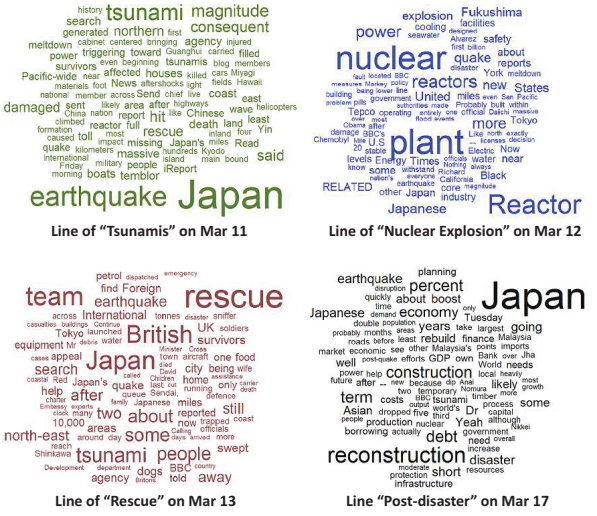


Figure 2: Word distributions vary a lot among nodes in different storylines

cover news and interviewing reports, contain one or two famous remarks. Since these documents also include too much useless information, it's inappropriate to use the whole document as a story node. So our work, the sentence-level story map extraction, is just aimed at this brand new problem. The second problem is the way to utilize the cross-modal

information suitably. Our goal is not only to fuse sentences and images together, but also to provide a unified framework to improve them mutually.

In this paper, we introduce a novel solution for the story map summarization problem. All the sentences and images are the candidates for making up the final map. The analysis of complicated information usually requires a semantic-level knowledge study. We address this in the pre-processing of data in Section 3.1. The key task of our research is the extraction of storylines. We reveal two fundamental properties of an ideal storyline, and propose an optimization algorithm in Section 3.2. A highly compatible MDS sub-algorithm is also fused in and offers help to the sentences/images selection. In Section 3.3, the extracted storylines are reconstructed as a final story map. The experimental results conducted on four real datasets show that our approach can perform effectively.

The rest of this paper is organized as follows. Some related researches are demonstrated in Section 2. We introduce our methodology in Section 3. The experimental results in Section 4 prove the effectiveness of our approach. Finally, we conclude this paper and present our future work in Section 5.

2 Related Work

Generally speaking, multi-document summarization can be either extractive or abstractive. Researchers mainly focus on the former which extracts the information deemed most important to the summary. Various techniques have been used for this type of MDS (Haghighi and Vanderwende, 2009; Contractor et al., 2012). Graph-based text summarization techniques have been widely used for years. The algorithms, used in TextRank (Mihalcea and Tarau, 2005) and LexPageRank (Radev et al., 2004), which are meant to compute sentence importance, are similar to those in PageRank and HITS.

Recently, timeline becomes a popular style to present a schedule of events and attracts many researchers consequently. For example, Yan et al. make use of timestamps to generate an evolutionary timeline (Yan et al., 2011). Shahaf et al. present a 2-D story map called "metro map" to summarize the complex topic (Shahaf et al., 2012). But previous work only studies on the document level, which inevitably brings about much information redundancy.

Previous studies show that the use of visual materials not only leads to the conservation of information but also promotes comprehension (Panjwani et al., 2009). Thus the cross-modal fusion is necessary. Wu et al. propose a framework of multimodal information fusion for multimedia data analysis by learning the optimal combination of multimodal information with the superkernel fusion (Wu et al., 2004). Borrowing the idea of recommendation in heterogeneous network into the cross-modal news summarization is also a convincing research. Xu et al. tackle this task and bring out an 1-D cross-media timeline generation framework (Xu et al., 2013).

Summarization of multimedia involves researches on information retrieval of multimedia. Since textual and visual information are quite different from each other, how to make a good transformation to mine latent knowledge from unannotated images is of great concern. Feng and Lapata (2010) use visual words to describe visual features and then propose a probabilistic model based on the assumption that images and their co-occurring textual data are generated by mixtures of latent topics.

However, to the best of our knowledge, no existing research manages to generate a 2-dimensional story map automatically and integrate images and texts into a unified framework at the same time.

3 Methodology

The original data of one event is a collection of news documents on different days, or in a finer granularity, a set of sentences and images with different timestamps. Each item in the data collection is a candidate for selection to form the final map. We denote the data collection as C, and $C = C_s \cup C_v$, where C_s is the subset containing all sentences and C_v contains all images. In the following elaboration of our method, dateset C is the important knowledge base. As the ultimate goal for a specific event, we would like to generate the "2-D" story map $\mathcal{M}$, whose main component is a set of "1-D" storylines $\mathcal{L} = \{L_1, L_2, \ldots\}$. Each storyline $L \in \mathcal{L}$ is made up of a set of "0-D" story nodes, $L = \{I_1, I_2, \ldots\}$, each of which is composed of a set of candidates sharing the same timestamp $I = \{c_1, c_2, \ldots\}$. For more concise, our method can be scheduled with the

following three steps:

1. Prepare semantic knowledge for each candidate, and then purify data collection C by eliminating the noisy candidates;

2. Conduct OPT-LSH algorithm to extract $\mathcal{L}$ that contains all qualified storylines;

3. Reconstruct the storylines as the final map $\mathcal{M}$.

3.1 Pre-processing

Before we extract the storylines, we have to solve two problems first. Since our work is cross-modal, the semantic knowledge under the literal and visual surface is basically required. Recall from Section 2, there exist many effective ways to dig into the semantic level of image and text. In this paper, we employ the approach proposed by Jiang and Tan (2006). They present a convincing approach which employs a multilingual retrieval model to apply knowledge mining on semantic level. The first is the sentence feature vector generation. With the preprocessing such as stemming and stop words removal, they extract the textual TF-IDF feature Vec_s of each sentence. The second, also the challenging part, is the image feature vector generation. In this step, for each region they extract visual features that are consisting of 6 color features and 60 gabor features which have been proven to be useful in many applications. Color features are the means and variances of the RGB color spaces. Gabor features are extracted by calculating the means and variations of the filtered image regions on 6 orientations. After the visual feature vectors of the image regions are extracted, all image regions are clustered using the k-means algorithm. The generated clusters, called "visterms" or "visual words", are treated as a vocabulary for the images. Besides visual features, they also utilize the context textual feature of each image as the semantic supplement to generate to final feature vector Vec_v.

Based on these feature vector of each cantidate, they further calculating the intra-modal similarity with classical IR methods, they obtain the inter-modal similarity through the vague transformation (Mandl T, 1998). We also note that translation tools such as VIPS (Cai et al., 2003) and WebKit[1] can help us to segment web documents and pick those text blocks whose coordinates are neighboring to each specified image. In this way we can successfully obtain images, text contexts and content sentences. Three kinds of semantic similarity are now ready. They are uniformly denoted as $sim(c_i, c_j)$, representing the similarity between two candidates. c_i and c_j can be any type of modalities.

Another problem is the noise from irregular data. We would like to utilize the sentences and images of high quality. The intuitive assumption is that a good candidate should have substance in speech, and be coherent with other good candidates. Fortunately, many useful measures are now available. They can be used to choose better candidates. Inspired by the analysis analogy to information retrieval, we extend the idea of the classical *PageRank* algorithm to estimate the authority for each candidate. The similarity between two candidates is regarded as the weighted "link" between them.

Inspired by the idea of classic "update summarization" task, we try to avoid those chronologically ordered documents sets focusing on a constant topic. Therefore, given the particularity of our task, we also have to develop the weighting function Γ with the temporal factor before starting the ranking algorithm. Our fundamental assumption is that the inter-date and inter-modal "links" have different influence comparing with the intra ones. Therefore the core formula calculating the authority of c_i is adapted as follows to make it become weight-compatible:

$$Auth(c_i) = \frac{1-q}{|C|} + q \cdot [\, \alpha \sum_{c_j \in C'} \Gamma(c_i, c_j) \cdot \frac{Auth(c_j)}{O(c_j)} + (1-\alpha) \sum_{c_k \in C''} \Gamma(c_i, c_k) \cdot \frac{Auth(c_k)}{O(c_k)} \,]$$

C' is the subset of C which only includes the candidates of the same modality with c_i, and C'' is the cross-modal candidates subset. $O(c_j)$ denotes the out-degree of c_j, and smoothing parameter q is set as the common value 0.85. Parameter α is used to balance the biases of intra- and inter-modal impact. If α is set to 1, it means the cross-modal information is abandoned, and vice versa. $\Gamma(c_i, c_j)$ contains two terms as follows:

$$\Gamma(c_i, c_j) = sim(c_i, c_j) \cdot e^{\frac{-\|c_i.t - c_j.t\|}{2\sigma^2}}$$

[1] http://www.webkit.org

The second term of Γ's formula is Gaussian Kernel (Aliguliyev R, 2009), which is used to measure the temporal gap between two candidates ($*.t$ denotes the date-based timestamp). Note that the similarity metric is content-based and time-independent, since the time decay function is only used to adjust the ranking impact strength. In this way, we can give higher authority to the more informative and coherent candidates. The optimal value of σ, which controls the spread of kernel curves, is sensitive to datasets and will be discussed later.

We eliminate the candidates whose authority goes under threshold. The data collection C is then significantly downsized and purified for later work. Authority is also used to determine the presentation sequence inside each story node of final map.

3.2 Extraction: LSH-OPT Algorithm

Other traditional relative methods need to pre-decide how many sub-topics are going to be obtained, like clustering or other supervised models. They fail in the unsupervised automatic storylines extraction problem. In this Section, we propose a concrete algorithm to extract storyline set $\mathcal{L}$ from C. Our proposed LSH-based algorithm can automatically optimize the number of storylines according to their self-evaluation.

3.2.1 Task Formalization

Let's investigate the storyline first. We may think of some basic attributes as well as many extension properties. The number of nodes in the storyline L (denoted as $|L|$) is intuitively one of its basic attributes. We would like to define another basic attribute called the SUPPORT of L. In detail, $Support(L) = \min_{I_k \in L} |I_k|$, which denotes the smallest size among all the story nodes it has.

The most challenging part is to properly model extension properties. We observe that an effective storyline should meet three key requirements: (1) **Coherence**. Within one storyline, news changes gradually as time goes and the evolution indicates consistency among component story nodes. We rely on the notion of coherence developed in *Connect-the-Dots* (Shahaf and Guestrin, 2010) and transform it to what we exactly need in this research; (2) **Diversity**. According to MMR principle (Goldstein et al., 1999), though the work is about summary, we still can draw an analogy and derive that a good storyline should be concise and contain redundant information as few as possible, i.e., two sentences providing information of similar content should not be presented in different storylines; (3) **Coverage**. The extracted storyline set $\mathcal{L}$ should keep alignment with the source collection C, which is intuitive and even proved to be significant as proposed in (Li et al., 2009). However, *Coverage* in some ways is technically redundant in front of *Diversity*. We decide to use the first two criteria in extraction process and use the last one to verify the effectiveness.

Since a storyline is composed of several nodes, we can select or abandon nodes mainly according to these two requirements. In fact, both of them involve a measurement of similarity between two story nodes, denoted by two word distributions (see Figure 2). Specifically, for story node I_i, its distribution probability of word w is estimated as $p(w|I_i) = \frac{\sum_{c \in I_i} TF(w)}{\sum_w \sum_{c \in I_i} TF(w)}$ where the denominator is used for normalization. Then *Kullback-Leibler* divergence is employed to denote the distance between two nodes I_i and I_j:

$$D_{KL}(I_i, I_j) = \sum_w p(w|I_i) \log \frac{p(w|I_i)}{p(w|I_j)}$$

In addition, we introduce the decreasing and increasing variants based on logistic functions, $D^{\flat}_{KL} = 1/(1 + e^{D_{KL}})$ and $D^{\sharp}_{KL} = e^{D_{KL}}/(1 + e^{D_{KL}})$, to map the distance into $[0, 1]$. Given the measurement, we can formulate the two properties.

For *Coherence*, a storyline L_i consists of a series of individual but correlated nodes, which do not necessarily have the serial timestamps. We would like to choose such a set of nodes $\{I_1, I_2, \ldots\}$, and at the same time guarantee this criterion:

$$Cor(L_i) = \frac{1}{|L_i|} \sum\nolimits_{1 \le k < |L_i|} D^{\flat}_{KL}(I_k, I_{k+1})$$

For *Diversity*, each storyline $Li \in \mathcal{L}$ should demonstrate quite different subtopics with other storylines. This is the most essential motivation for us to step into 2-dimensional field. This criterion can be used to maximize the minimum diversity value among all storylines:

$$Div(\mathcal{L}) = \min_{L_i, L_j \in \mathcal{L}} \{ \frac{\sum_{I_k \in L_i} \sum_{I_{k'} \in L_j} D^{\sharp}_{KL}(I_k, I_{k'})}{|L_i| \cdot |L_j|} \}$$

Then the problem can be transformed into the following optimization problem. Parameters θ_1, θ_2 and θ_3 denote the minimum number of nodes in each line, the smallest size of candidates in each node and the coherence lower bound respectively. The task is to extract an optimal $\mathcal{L}$ out of C, such that:

$$\forall L \in \mathcal{L},\ |L| \geq \theta_1\ \&\ Support(L) \geq \theta_2;$$
$$\forall L \in \mathcal{L},\ Cor(L) \geq \theta_3;$$
$$Div(\mathcal{L})\ is\ maximized.$$

3.2.2 Optimization Algorithm

It can be proved that finding the optimal set $\mathcal{L}$ is an NP-Complete problem (not presented due to the limited space). Thus the brute-force exhaustive approach is crashed. We develop a near-optimal algorithm based on locality sensitive hashing (OPT-LSH). The original LSH solution is a popular technique used to solve the nearest neighbor search problems in high dimensions. Its basic idea is to hash similar input items into the same bucket (i.e., uniquely definable hash signature) with high probability. All potential storylines can be targeted fast if we make good use of this idea.

LSH performs probabilistic dimension reduction of high dimensional data by projecting a higher d-dimensional vector Vec_c (recall from Section 3.1) to a lower d'-dimensional vector ($d'<<d$), such that the candidates which are in close proximity in the higher dimension get mapped into the same item in the lower dimensional space with high probability. It guarantees a lower bound on the probability that two similar input items fall into the same bucket in the projected space and also the upper bound on the probability that two dissimilar vectors fall into the same bucket (Indyk and Motwani, 1998).

One of the key requirements for good performance of LSH is the careful selection of the family of hashing functions. In OPT-LSH, we use the hashing scheme proposed by Charikar (Charikar M, 2002). In detail, d' random unit d-dimensional vectors $\vec{r_1}, \vec{r_2}, \ldots, \vec{r_{d'}}$ are generated first. Each of the d entries of $\vec{r_i}$ is drawn from a standardized normal distribution N(0,1). Then the d' hashing functions are defined as:

$$h_i(c) = \begin{cases} 1, & if\ \vec{r_i} \cdot Vec(c) \geq 0 \\ 0, & if\ \vec{r_i} \cdot Vec(c) < 0 \end{cases} \quad 1 \leq i \leq d'$$

We represent the d'-dimensional bucket feature $\mathbf{h}$ for c, $\mathbf{h}(c) := [h_1(c), \ldots, h_{d'}(c)]$. There are $2^{d'}$ different buckets at most. Each denotes a potential storyline, so we have to verify the probability of similar candidates falling into the same bucket, whose lower bound is given by Charikar (Charikar M, 2002).

Simply filtering and searching among all potential lines in single pass may lead to empty result set if in post-processing no bucket satisfies all constraints. This could probably happen because the input parameter d' is set so large that the optimal set of candidates is separated into different buckets. However, we will get a suboptimal result in turn when d' is too small. We are then motivated to tune LSH by iterative relaxation that varies d' in each iteration. Changing the value of d' balances the leverage between expected number of potential storylines and their properties. We perform a binary search between 1 and d' to identify the ideal number of hash functions to employ. Algorithm 1 shows the pseudo code of our OPT-LSH algorithm. The LSH time is bounded by $O(d'|C| \log |C|)$ since the binary search relaxation iteration runs for $\log |C|$ times in the worst case and the hashing time is $O(d'|C|)$.

3.3 Storylines Reconstruction

At last, we manage to reconstruct all the storylines in $\mathcal{L}^{opt}$. A real-world storyline may sometimes intertwine with another, educe other branches, and end its own evolvement. The way to reconstruct a more effective layout of the story map requires further study and provides a good research direction in the future. However, in this paper we order the sentences/images in each story node according to their authority scores. Next, all storylines are arranged to proceed along the timestamps, thus a storyline never turns back in the map. Then we adjust the structure to make story nodes sharing the same timestamp stay close, though they belong to different lines. Figure 1 shows the sample output of our system.

4 Experiments

4.1 Dataset

There is no existing standard evaluation data set for 2-dimensional cross-modal summarization methods. We randomly choose 4 news topics from 4 selected news websites: *New York Times*, *BBC*, *CNN* and

Algorithm 1 OPT-LSH Algorithm

Input: Candidate set C, similarity function sim, bucket dimensions d'

Output: A near-optimal storylines set $\mathcal{L}^{opt}$

// Main Algorithm

Initialize $left = 1, right = d', max = -1$
repeat
  $d' = (left + right)/2$
  $\mathcal{L} \leftarrow LSH(C, d')$
  Revise all word distributions $p(w|I)$ in $\mathcal{L}$
  if $\forall L \in \mathcal{L}$, $|L| \geq \theta_1$ **and** $Support(L) \geq \theta_2$ **and** $Cor(L) \geq \delta_3$ **then**
    if $Div(\mathcal{L}) > max$ **then**
      $\mathcal{L}^{opt} \leftarrow \mathcal{L}, max \leftarrow Div(\mathcal{L})$
    end if
    $left = d' + 1$
  else
    $right = d' - 1$
  end if
until $left > right$
return $\mathcal{L}^{opt}$

// $LSH(C, d') : Buckets$

Generate d' unit vectors randomly
for $j = 1$ to $|C|$ **do**
  for $i = 1$ to d' **do**
    if $\vec{r_i} \cdot Vec(c_j) \geq 0$ **then**
      $h_i(c_j) \leftarrow 1$
    else
      $h_i(c_j) \leftarrow 0$
    end if
  end for
  $\mathbf{h}(c_j) = [h_1(c_j), \ldots, h_{d'}(c_j)]$
end for
return $Buckets \leftarrow \{\mathbf{h}(c_j) | c_j \in C\}$

Reuters. We query each event confined to these sites and crawl webpages' html docs. Referring to Section 3.1, timestamps, text contents, images and their text contexts are extracted. Table 1 shows the details. These 4 datasets all contain massive information and complex evolutions.

4.2 Analysis of Our System

Since there is no standard for us to verify the effectiveness of our solution, we have to utilize convincing criteria based on manual evaluation of experts, and then we can compare them with other approaches. In order to setup the system, based on the optimization problem shown in Section 3.2, we assign the value of 1 to both θ_1 and θ_2, and empirically set θ_3 as 0.6 which can balance the number of potential lines and the quality of story map as well. Then the problem can be re-interpreted in natural language as follows. Given the data collection, we manage to find out a set of storylines such that every line in it contains at least one non-null story node and keeps self-coherence not less than 0.6. What's more, the diversity of the whole set is maximized. Before comparing OPT-LSH with other systems, we do some further analysis of inherent properties first.

Table 1: Statistics of Datasets.

Event (Query)	Document	Time Span	σ	α
EJE	504	Mar 11-Apr 8, 2011	3	0.9
OWS	638	Sept 17-Dec 10, 2011	12	0.7
NBA	489	July 1-Dec 28, 2011	17	0.6
ME	437	June 21-Aug 31, 2011	9	0.7

* Abbreviations EJE, OWS, NBA, ME denote *East Japan Earthquake*, *Occupy Wall Street*, *NBA Lockout* and *Murdock's Eavesdropping* respectively.

4.2.1 Compactness

The essential idea of summarization is to reduce the data size, so that a more concise representation will be generated and help users to fast grasp the main points. Therefore the *Compactness* of a story map needs to be guaranteed. In the pre-processing module, we have already excluded significant number of inferior candidates with the extended PageRank. Nevertheless what we really care is the compactness that OPT-LSH brings about. In the candidates and story nodes selection processes, only the most saliency and coherent candidates can appear in the final representation. We count the number of sentences and images in $\mathcal{L}^{opt}$, denoted as $||\mathcal{L}^{opt}||$, and then we compare it with the collection size $|C|$ (both before and after pre-processing) to test the compactness. Table 2 shows that OPT-LSH further reduces the representation scale significantly.

4.2.2 Coverage

Obviously, only the verification of compactness is far from enough. As mentioned in Section 3.2, the storyline set $\mathcal{L}$ we extract should keep alignment with the source collection, and contain informative as well as comprehensive information in C. Thus we

Table 2: The compactness of OPT-LSH

Dataset	EJE	OWS	NBA	ME
$\|C\|$-before	12049	22890	20403	10237
$\|C\|$-after	1454	2357	1187	1042
$\|\|\mathcal{L}\|\|$	87	168	113	92
Downsizing	94.0%	92.9%	90.5%	91.2%

need to verify another property of our summarization, the *Coverage*. Inspired by (Shannon C, 2001), we employ the *Information Entropy* to represent the information quantity based on solid mathematical theory. The less information quantity decreases after summarizing, the more story comprehensiveness is maintained. In this way we verify the property of coverage. Particularly, *Shannon* denotes the entropy H as follows of a discrete random variable X, which in fact is a word distribution. The base knowledge in our work is the global probability mass function $P(W_C)$ based on the entire vocabulary of C, with possible values $\{p(w_1), \ldots, p(w_{|W_C|})\}$.

$$H(X) = E\{-\log(P(W_C))\|X\} = -\sum_{w_i \in X} p(w_i)\log(p(w_i))$$

Although different word distributions may have the same H, we do not focus on the similarity of two corpora, but the difference of information quantities they are carrying. So H is an ideal criterion.

Besides the comparison of the entropies of C and $\mathcal{L}$, it gives us a chance to study different modules' contributions in our solution. There are two places that we may simplify the solution. One is the feature of temporal gap. If we set the parameter σ to $infinity$, then we can remove the second term of the calculation of Γ (i.e. $\sigma \leftarrow \infty$) and then bring out a *time-insensitive* system. The other is the cross-modal feature. We set parameter α as 1 to make the system work in one single modality, and ignore all images (only the textual sentences are available) to make up story map. The work then becomes the study with *text-bias*. We also implement the *simplest* system that blocks images as well as temporal feature. Figure 3 highlights the good performance of our system. We are maintaining information by a larger proportion of the original data collection. Considering the property of high compactness, our solution tackles the information redundancy quite well, and promisingly delivers entire knowledge with compact structure.

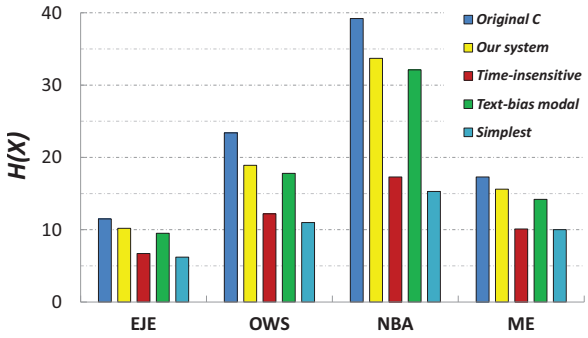


Figure 3: The y-axis denotes the entropy. And the larger H is, the richer information it brings.

During our experiments on *Coverage*, we have some interesting findings. Datasets perform differently when we take different values of σ and α, which controls the temporal decaying rate and cross-modal learning respectively. The events with short life-cycles prefer a smaller value of σ to dominate the influence from neighbors, as well as the intra-modal bias. On the contrary, long-living events prefer lager σ and more inter-modal bias to get information replenishment from different dates and modality. Due to the limited space we don't present the tuning details, but the optimal values are shown in Table 1. In fact, using the cross-modal mutual influence can, more or less, help to improve the effectiveness of information extraction and summarization.

4.3 User Study

Before we introduce other existing methods that can also tackle the cross-modal 2-dimensional summarization problem, we have to setup the appropriate standards to quantify users' evaluation.

4.3.1 Metrics

In the user study, we evaluate the effectiveness of our story maps in aiding users to integrate different aspects of multi-faceted information. (Shahaf et al., 2012) also focuses on story map generation and puts forward two convincing metrics to answer the following questions:

- *Micro-Knowledge*: Can the maps help users retrieve information faster than other methods?
- *Macro-Knowledge*: Can the maps help users understand the big picture better than other methods?

For *micro-knowledge*, we wish to see how maps help users answer specific questions. We compare the level of knowledge attained by users using our method with two other systems: *Google News* and *TDT*. Google News is a computer-generated site that aggregates headlines from news sources worldwide. News-viewing tools are dominated by portal and search approaches, and *Google News* is a typical representative of those tools. *TDT* (Nallapati et al., 2004) is a successful system which captures the rich structure and dependencies of news events.

We have noticed that making comparisons between different systems is not convincing, since the output of Google News and TDT is different both in content and in presentation (and in particular, cannot be double-blind). In order to isolate the effects of sentence selection vs. map organization, we introduce a hybrid system into the study: the system with *structureless* story map displays the same sentences and images as our system but with none of the structure. Its output is basically the same with our full system but with a single storyline and merges node content for each date. And each story nodes are sorted chronologically and displayed similarly to Google News. We implement TDT based on (Nallapati et al., 2004) (cos + TD + SimpleThresholding), and pick a representative article from each cluster. The purpose of the study is to test a single query. We also obtain the results from Google News using the same queries.

We recruit 64 volunteers to browse all the four events, and one of the four systems is assigned to each person randomly. After browsing, users are asked to answer a short questionnaire (8 questions), composed by domain experts. Users answer as many questions as possible in limited time (8 minutes). The statistics of their answers are promised to evaluate the micro-knowledge on different systems. In order to aid in comprehension, we give some examples about those asked questions. For the event of EJE, we ask that

1. *How many magnitude was initially reported by the USGS, and what about the finally report?*
2. *List at least six countries that had dispatched their rescue teams.*
3. ...

And for OWS, we ask that

Table 3: *Macro-knowledge* performance on four datasets

Dataset	Our System	Google News	TDT
EJE	56.3%	23.2%	20.5%
OWS	62.2%	22.1%	15.7%
NBA	58.3%	18.9%	22.8%
ME	47.2%	26.3%	26.5%

1. *What was the attitude of President Obama about the protesters on October?*
2. *When did the protesters begin dressing "corporate zombies" in New York?*
3. ...

These questions can effectively help us to investigate users' $micro\text{-}knowledge$ about the events.

As for *macro-knowledge*, unlike the retrieval study that evaluates users' ability to answer questions, we are interested in the use of story maps as high-level overviews, allowing users to understand the big picture. We believe that the ability to explain a certain issue is the only proof of understanding. Therefore, the 64 volunteers are then asked to write four paragraphs to summarize the four events respectively. This time, all three systems' (the structureless system presents the same content as our system) results are provided and we let users choose the sentences with complete freedom. Then we count the number of sentences they employed from each system and derive the average proportions. According to the results we can research on the macro-knowledge that different systems deliver.

4.3.2 Results

We take the time cost and the average numbers of correct answers of different systems to evaluate on the micro-knowledge. Figure 4 shows the results.

We can find out that our system outperforms the others significantly when users taking less time to learn the knowledge. The failure of structureless system proves that our work of storyline reconstruction makes lots of advantages to help reading.

On the other hand, Table 3 analyzes the statistics of macro-knowledge. It's also obvious that users would like to refer to the sentences that our system provides. A reasonable explanation is that the story maps we generate can clarify users' thoughts and views on the complicated events.

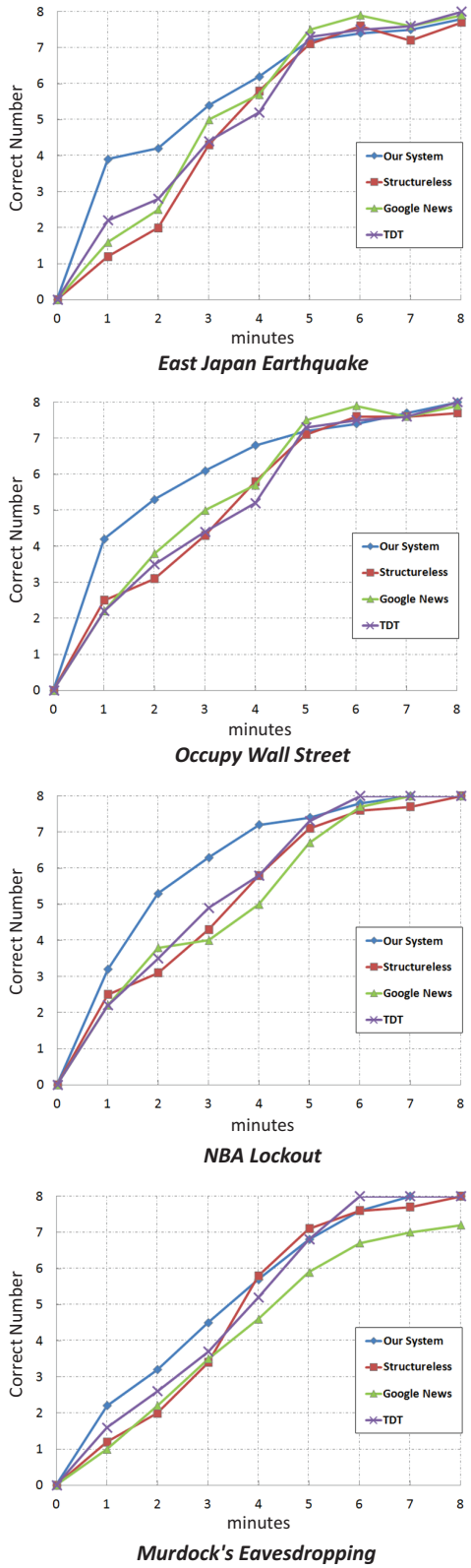


Figure 4: *Micro-knowledge* performance on four datasets

Table 4: Average runtime of different datasets

Dataset	EJE	OWS	NBA	ME
$\|C\|$	1454	2357	1187	1042
Iterations	3.5	4.7	5.4	3.7
Runtime (ms)	1582	3214	3089	1314

4.4 Runtime Analysis

At last, we analyze the time performance of our OPT-LSH algorithm on a PC server (16G RAM, 2.67GHz 4-processors CPU). The average iterations for different initial value of d' and the runtime are shown in Table 4. The results are acceptable.

5 Conclusions

In this paper, we study the feasibility of automatically generating cross-modal story maps and present a novel solution to this challenging problem. Our works mainly tackle the problems of storylines extraction and reconstruction. Specifically, we investigate two requisite properties of an ideal storyline, *Coherence* and *Diversity*. Then the convincing criteria are devised to model both. We formalize the task as an optimization problem and design an algorithm to solve it. Classical IR and text analyzing techniques like PageRank are fused into the unified framework, and a near-optimal solution is employed to deal with the NP-complete problem. Experiments on web datasets show that our method is quite efficient and competitive. We also verify that it brings quicker, clearer and deeper comprehension to users.

As a future work, we plan to adapt parameters automatically on the basis of different types of datasets. Improving the layout quality of story map by concerning the interactivity of different media (e.g. images order) is also significant. Furthermore, our framework is universal, so that the media other than text and image can be adopted as well.

Acknowledgments

We sincerely thank all the anonymous reviewers for their valuable comments, which have helped to improve this paper greatly.

This work is supported by NSFC with Grant No. 61073081 and 61370054, and 973 Program with Grant No. 2014CB340405.

References

Agrawal R, Gollapudi S, Kannan A, et al. 2011 *Enriching textbooks with images*. Proceedings of the 20th ACM International Conference on Information and Knowledge Management, ACM, pages 1847-1856.

Aliguliyev R M. 2009 *A new sentence similarity measure and sentence based extractive technique for automatic text summarization*. Expert Systems with Applications, 2009, 36(4): 7764-7772.

Allan J, Gupta R, Khandelwal V. 2001 *Temporal summaries of new topics*. Proceedings of the 24th Annual International ACM Conference on SIGIR, pages 10-18.

Cai D, Yu S, Wen J R, et al. 2003 *VIPS: a visionbased page segmentation algorithm*. Microsoft Technical Report, MSR-TR-2003-79.

Charikar M S. 2002 *Similarity estimation techniques from rounding algorithms*. Proceedings of the 34th Annual ACM Symposium on Theory of Computing, ACM, pages 380-388.

Chen Y, Jin O, Xue G R, et al. 2010 *Visual contextual advertising: Bringing textual advertisements to images*. Proceedings of the 24th AAAI Conference, AAAI, pages 1314-1320.

Contractor D, Guo Y, Korhonen A. 2012. *Using Argumentative Zones for Extractive Summarization of Scientific Articles*. COLING, pages 663-678.

Evans D K, McKeown K, Klavans J L. 2005 *Similarity-based multilingual multi-document summarization*. IEEE Transactions on Information Theory, pages 1858C1860.

Feng Y, Lapata M. 2010 *Topic models for image annotation and text illustration*. Human Language Technologies: The 2010 Annual Conference of NAACL, pages 831-839.

Goldstein J, Mittal V, Carbonell J, Kantrowitz M. 2000 *Multi-document summarization by sentence extraction*. Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4. Association for Computational Linguistics, pages 40-48.

Goldstein J, Kantrowitz M, Mittal V, et al. 1999 *Summarizing text documents: sentence selection and evaluation metrics*. Proceedings of the 22nd Annual International ACM Conference on SIGIR, ACM, pages 121-128.

Haghighi A, Vanderwende L. 2009. *Exploring content models for multi-document summarization*. Proceedings of Human Language Technologies: The 2009 Annual Conference of NAACL, Association for Computational Linguistics, pages 362-370.

Indyk P, Motwani R. 1998 *Approximate nearest neighbors: towards removing the curse of dimensionality*. Proceedings of the 30th Annual ACM Symposium on Theory of Computing, ACM, pages 604-613.

Jiang T, Tan A H. 2006 *Discovering image-text associations for cross-media web information fusion*. Knowledge Discovery in Databases: PKDD 2006, pages 561-568.

Li L, Zhou K, Xue G R, et al. 2009 *Enhancing diversity, coverage and balance for summarization through structure learning*. Proceedings of the 18th International Conference on World Wide Web, ACM, pages 71-80.

Mandl T. 1998 *Vague transformations in information retrieval*. ISI 1998, pages 312-325.

Mihalcea R, Tarau P. 2005 *A language independent algorithm for single and multiple document summarization*. Proceedings of IJCNLP 2005.

Nallapati R, Feng A, Peng F, et al. 2004 *Event threading within news topics*. Proceedings of the 13rd ACM International Conference on Information and Knowledge Management, ACM, pages 446-453.

Panjwani S, Micallef L, Fenech K, et al. 2009 *Effects of integrating digital visual materials with textbook scans in the classroom*. International Journal of Education and Development using ICT, 2009, 5(3).

Radev D R, Jing H, et al. 2004 *Centroid-based summarization of multiple documents*. Information Processing & Management, 2004, 40(6): 919-938.

Radev D, Winkel A, Topper M. 2002 *Multi document centroid-based text summarization*. ACL Demo Session, 2002.

Shahaf D, Guestrin C. 2010 *Connecting the dots between news articles*. Proceedings of the 16th ACM Conference on SIGKDD, ACM, pages 623-632.

Shahaf D, Guestrin C, Horvitz E. 2012 *Trains of thought: Generating information maps*. Proceedings of the 21st International Conference on World Wide Web, ACM, pages 899-908.

Shannon C E. 2001 *A mathematical theory of communication*. ACM SIGMOBILE Mobile Computing and Communications Review, 2001, 5(1): 3-55.

Wu Y, Chang E Y, Chang K C C, et al. 2004 *Optimal multimodal fusion for multimedia data analysis*. Proceedings of the 12th annual ACM International Conference on Multimedia, ACM, 2004: 572-579.

Xu S, Kong L, Zhang Y. 2013 *A cross-media evolutionary timeline generation framework based on iterative recommendation*. Proceedings of the 3rd ACM conference on International Conference on Multimedia Retrieval, ACM, pages 73-80.

Yan R, Wan X, Otterbacher J, et al. 2011 *Evolutionary timeline summarization: a balanced optimization framework via iterative substitution*. Proceedings of the 34th International ACM Conference on SIGIR, ACM, pages 745-754.

Image Description using Visual Dependency Representations

Desmond Elliott
School of Informatics
University of Edinburgh
d.elliott@ed.ac.uk

Frank Keller
School of Informatics
University of Edinburgh
keller@inf.ed.ac.uk

Abstract

Describing the main event of an image involves identifying the objects depicted and predicting the relationships between them. Previous approaches have represented images as unstructured bags of regions, which makes it difficult to accurately predict meaningful relationships between regions. In this paper, we introduce visual dependency representations to capture the relationships between the objects in an image, and hypothesize that this representation can improve image description. We test this hypothesis using a new data set of region-annotated images, associated with visual dependency representations and gold-standard descriptions. We describe two template-based description generation models that operate over visual dependency representations. In an image description task, we find that these models outperform approaches that rely on object proximity or corpus information to generate descriptions on both automatic measures and on human judgements.

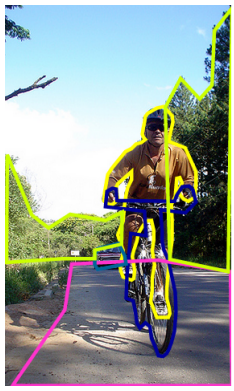
1 Introduction

Humans are readily able to produce a description of an image that correctly identifies the objects and actions depicted. Automating this process is useful for applications such as image retrieval, where users can go beyond keyword-search to describe their information needs, caption generation for improving the accessibility of existing image collections, story illustration, and in assistive technology for blind and partially sighted people. Automatic image description presents challenges on a number of levels: recognizing the objects in an image and their attributes are difficult computer vision problems; while determining how the objects interact, which relationships hold between them, and which events are depicted requires considerable background knowledge.

Previous approaches to automatic description generation have typically tackled the problem using an object recognition system in conjunction with a natural language generation component based on language models or templates (Kulkarni et al., 2011; Li et al., 2011). Some approaches have utilised the visual attributes of objects (Farhadi et al., 2010), generated descriptions by retrieving the descriptions of similar images (Ordonez et al., 2011; Kuznetsova et al., 2012), relied on an external corpus to predict the relationships between objects (Yang et al., 2011), or combined sentence fragments using a tree-substitution grammar (Mitchell et al., 2012).

A common aspect of existing work is that an image is represented as a bag of image regions. Bags of regions encode which objects co-occur in an image, but they are unable to express how the regions relate to each other, which makes it hard to describe what is happening. As an example, consider Figure 1a, which depicts a man riding a bike. If the man was instead repairing the bike, then the bag-of-regions representation would be the same, even though the image would depict a different action and would have to be described differently. This type of co-occurrence of regions indicates the need for a more structured image representation; an image description system that has access to structured repre-

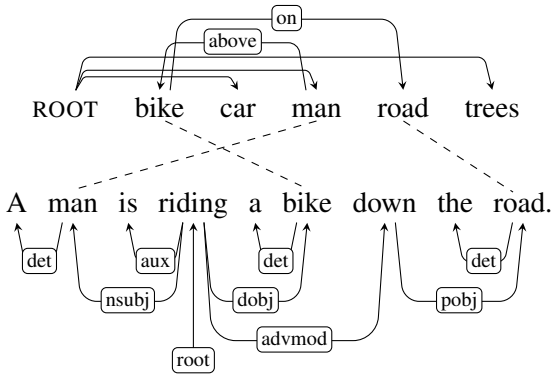
Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1292–1302,
Seattle, Washington, USA, 18-21 October 2013.



(a)

A man is riding a bike down the road.
A car and trees are in the background.

(b)



(c)

Figure 1: (a) Image with regions marked up: BIKE, CAR, MAN, ROAD, TREES; (b) human-generated image description; (c) visual dependency representation expressing the relationships between MAN, BIKE, and ROAD aligned to the syntactic dependency parse of the first sentence in the human-generated description (b).

sentations would be able to correctly infer the action that is taking place, such as the distinction between repairing or riding a bike, which would greatly improve the descriptions it is able to generate.

In this paper, we introduce *visual dependency representations* (VDRs) to represent the structure of images. This representation encodes the geometric relations between the regions of an image. An example can be found in Figure 1c, which depicts the VDR for Figure 1a. It encodes that the MAN is above the BIKE, and that the BIKE is on the ROAD. These relationships make it possible to infer that the man is riding a bike down the road, which corresponds to the first sentence of the human-generated image description in Figure 1b.

In order to test the hypothesis that structured image representations are useful for description generation, we present a series of template-based image description models. Two of these models are based on approaches in the literature that represent images as bags of regions. The other two models use visual dependency representations, either on their own or in conjunction with gold-standard image descriptions at training time.

We find that descriptions generated using the VDR-based models are significantly better than those generated using bag-of-region models in automatic evaluations using smoothed BLEU scores and in human judgements. The BLEU score improvements are found at bi-, tri-, and four-gram levels, and humans rate VDR-based image descriptions 1.2 points above the next-best model on a 1–5 scale.

Finally, we also show that the benefit of the visual dependency representation is maintained when image descriptions are generated from automatically parsed VDRs. We use a modified version of the edge-factored parser of McDonald et al. (2005) to predict VDRs over a set of annotated object regions. This result reaffirms the potential utility of this representation as a means to describe events in images. Note that throughout the paper, we work with gold-standard region annotations; this makes it possible to explore the effect of structured image representations independently of automatic object detection.

2 Visual Dependency Representation

In analogy to dependency grammar for natural language syntax, we define *Visual Dependency Grammar* to describe the spatial relations between pairs of image regions. A directed arc between two regions is labelled with the spatial relationship between those regions, defined in terms of three geometric properties: pixel overlap, the angle between regions, and the distance between regions. Table 1 presents a detailed explanation of the spatial relationships defined in the grammar.

A visual dependency representation of an image is constructed by creating a directed acyclic graph

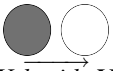
 X $\overrightarrow{on}$ Y	More than 50% of the pixels of region X overlap with region Y.[1]
 X $\overrightarrow{surrounds}$ Y	The entirety of region X overlaps with region Y.
 X $\overrightarrow{beside}$ Y	The angle between the centroid of X and the centroid of Y lies between 315° and 45° or 135° and 225°.
 X $\overrightarrow{opposite}$ Y	Similar to *beside*, but used when there X and Y are at opposite sides of the image.
 X $\overrightarrow{above}$ Y	The angle between X and Y lies between 225° and 315°.
 X $\overrightarrow{below}$ Y	The angle between X and Y lies between 45° and 135°.
 X $\overrightarrow{infront}$ Y	The Z-plane relationship between the regions is dominant.
 X $\overrightarrow{behind}$ Y	Identical to *infront* except X is behind Y in the Z-plane.

Table 1: Visual Dependency Grammar defines eight relations between pairs of annotated regions. To simplify explanation, all regions are circles, where *X* is the grey region and *Y* is the white region. All relations are considered with respect to the centroid of a region and the angle between those centroids. We follow the definition of the unit circle, in which 0° lies to the right and a turn around the circle is counter-clockwise.

over the set of regions in an image using the spatial relationships in the Visual Dependency Grammar. It is created from a region-annotated image and a corresponding image description by first identifying the central actor of the image. The central actor is the person or object carrying out the depicted action; this typically corresponds to the subject of the sentence describing the image. The region corresponding to the central actor is attached to the ROOT node of the graph. The remaining regions are then attached based on their relationship with either the actor or the other regions in the image as they are mentioned in the description. Each arc introduced is labelled with one of the spatial relations defined in the grammar, or with no label if the region is not described in relation to anything else in the image.

As an example of the output of this annotation process, consider Figure 1a, its description in 1b, and its VDR in 1c. Here, the MAN is the central actor in the image, as he is carrying out the depicted action (riding a bike). The region corresponding to MAN is therefore attached to ROOT without a spatial relation. The BIKE region is then attached to the MAN region using the $\overrightarrow{above}$ relation and BIKE is attached to the ROAD with the $\overrightarrow{on}$ relation. In the second sentence of the description, CAR and TREES are mentioned without a relationship to anything else in the image, so they are attached to the ROOT node. If these regions were attached to other regions, such as CAR $\overrightarrow{above}$ ROAD then this would imply structure in the image that is not conveyed in the description.

2.1 Data

Our data set uses the images from the PASCAL Visual Object Classification Challenge 2011 action recognition taster competition (Everingham et al., 2011). This is a closed-domain data set containing images of people performing ten types of actions, such as making a phone call, riding a bike, and taking a photo. We annotated the data set in a three-step process: (1) collect a description for each image; (2) annotate the regions in the image; and (3) create a visual dependency representation of the image. Note that Steps (2) and (3) are dependent on the image description, as both the region labels and the relations between them are derived from the description.

2.2 Image Descriptions

We collected three descriptions of each image in our data set from Amazon Mechanical Turk. Workers were asked to describe an image in two sentences. The first sentence describes the action in the image, the person performing the action and the region involved in the action; the second sentence describes any other regions in the image not directly involved in the action. An example description is given in Figure 1b.

A total of 2,424 images were described by three workers each, resulting in a total of 7,272 image de-

[1] As per the PASCAL VOC definition of overlap in the object detection task (Everingham et al., 2011).

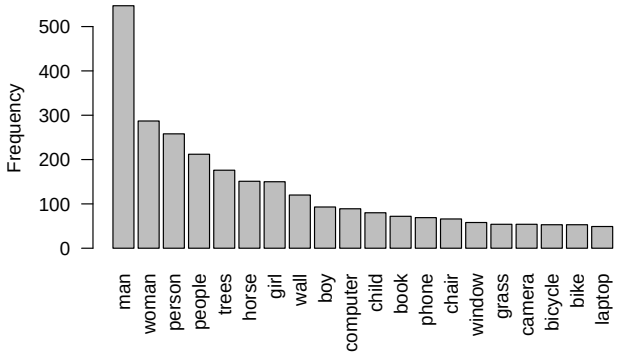


Figure 2: Top 20 annotated regions.

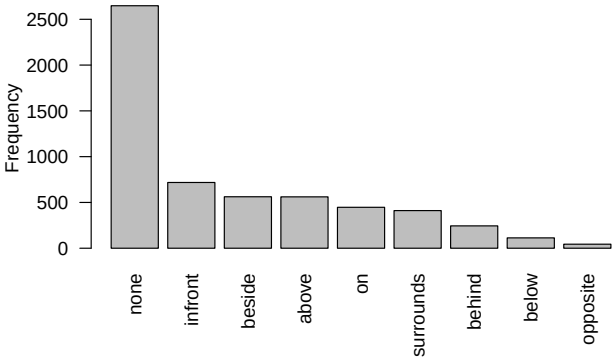


Figure 3: Distribution of the spatial relations.

scriptions. The workers, drawn from those registered in the US with a minimum HIT acceptance rate of 95%, described an average of 145 $\pm$ 93 images; they were encouraged to describe fewer than 300 images each to ensure a linguistically diverse data set. They were paid \$0.04 per image and it took on average 67 $\pm$ 123 seconds to describe a single image. The average length of a description was 19.9 $\pm$ 6.5 words in a range of 8–50 words. Dependency parses of the descriptions were produced using the MST-Parser (McDonald et al., 2005) trained on sections 2-21 of the WSJ portion of the Penn Treebank.

2.3 Region Annotations

We trained two annotators to draw polygons around the outlines of the regions in an image using the LabelMe annotation tool (Russell et al., 2008). The regions annotated for a given image were limited to those mentioned in the description paired with the image. Region annotation was performed on a subset of 341 images and resulted in a total of 5,034 annotated regions with a mean of 4.19 $\pm$ 1.94 annotations per image. A total of 496 distinct labels were used to label regions. Figure 2 shows the distribution of the top 20 region annotations in the data; people-type regions are the most commonly annotated regions. Given the prevalence of labels referring to the same types of regions, we defined 26 sets of equivalent labels to reduce label sparsity (e.g., BIKE was considered equivalent to BICYCLE). This normalization process reduced the size of the region label vocabulary from 496 labels to 362 labels. Inter-annotator agreement was 74.3% for region annotations, this was measured by computing polygon overlap over the annotated regions.

2.4 Visual Dependency Representations

The same two annotators were trained to construct gold-standard visual dependency representations for annotated image–description pairs. The process for creating a visual dependency representation of an image is described earlier in this section of the paper. The 341 region-annotated images resulted in a set of 1,023 visual dependency representations. The annotated data set comprised a total of 5,748 spatial relations, corresponding to a mean of 4.79 $\pm$ 3.51 relations per image. Figure 3 shows the distribution of spatial relation labels in the data set. It can be seen that the majority of regions are attached to the ROOT node, i.e., they have the relation label *none*. Inter-annotator agreement on a subset of the data was measured at 84% agreement for labelled dependency accuracy and 95.1% for unlabelled dependency accuracy. This suggests the task of generating visual dependency representations can be performed reliably by human annotators. We induced an alignment between the annotated region labels and words in the image description using simple lexical matching augmented with WordNet hyponym lookup. See Figure 1c for an example of the alignments.

3 Image Description Models

We present four template-based models for generating image descriptions in this section. Table 2

	Regions	VDR	External Corpus	Parallel text
PROXIMITY	✓			
CORPUS	✓		✓	
STRUCTURE	✓	✓		
PARALLEL	✓	✓		✓

Table 2: The data available to each model at training time.

T_1	DT O_i AUX REL DT O_j. T_5?
T_2	There AUX also $\{\text{DT } O_i\}_{i=1}^{\lvert unrelated \rvert}$ in the image.
T_3	DT O_i AUX REL DT O_j REL DT O_k. T_5?
T_4	REL DT O_j.
T_5	PRP AUX $\{\text{REL DT } O_i\}_{i=1}^{\lvert dependents \rvert}$.

Table 3: The language generation templates.

presents an overview of the amount of information available to each model at training time, ranging from only the annotated regions of an image to using visual dependency representation of an image aligned with the syntactic dependency representation of its description. At test time, all models have access to image regions and their labels, and use these to generate image descriptions. Two of the models also have access to VDRs at test time, allowing us to test the hypothesis that image structure is useful for generating good image descriptions.

The aim of each model is to determine what is happening in the image, which regions are important for describing it, and how these regions relate to each other. Recall that all our images depict actions, and that the gold-standard annotation was performed with this in mind. A good description therefore is one that relates the main actors depicted in the image to each other, typically through a verb; a mere enumeration of the regions in the image is not sufficient. All models attempt to generate a two-sentence description, as per the gold standard descriptions.

In the remainder of this section, we will use Figure 1 as a running example to demonstrate the type of language each model is capable of generating. All models share the set of templates in Table 3.

3.1 PROXIMITY

PROXIMITY is based on the assumption that people describe the relationships between regions that are near each other. It has access to only the annotated image regions and their labels.

Region–region relationships that are potentially relevant for the description are extracted by calculating the proximity of the annotated regions. Here, o_i is the subject region, o_j is the object region, and s_{ij} is the spatial relationship between the regions. Let $R = \{(o_i, s_{ij}, o_j), \dots\}$ be the set of possible region–region relationships found by calculating the nearest neighbour of each region in Euclidean space between the centroids of the polygons that mark the region boundaries. The tuple with the subject closest to the centre of the image is used to describe what is happening in the image, and the remaining regions are used to describe the background.

The first sentence of the description is realised with template T_1 from Table 3. o_i is the label of the subject region and o_j is the label of the object region. DT is a simple determiner chosen from {the, a}, depending on whether the region label is a plural noun; AUX is either {is, are}, depending on the number of the region label; and REL is a word to describe the relationship between the regions. For this model, REL is the spatial relationship between the centroids chosen from {above, below, beside}, depending on the angle formed between the region centroids, using the definitions in Table 1. The second sentence of the description is realised with template T_2 over the subjects o_i in R that were not used in the first sentence. An example of the language generated is:

(1) The man is beside the bike. There is also a road, a car, and trees in the image.

With the exception of visual attributes to describe size, colour, or texture, this model is based on the approach described by Kulkarni et al. (2011).

3.2 CORPUS

The biggest limitation of PROXIMITY is that regions that are near each other are not always in a relevant relationship for a description. For example, in Figure 1, the BIKE and the CAR regions are nearest neighbours but they are unlikely to be described as being in an relationship by a human annotator. The model CORPUS addresses this issue by using an

external text corpus to determine which pairs of regions are likely to be in a describable relationship. Furthermore, CORPUS can generate verbs instead of spatial relations between regions, leading to more human-like descriptions. CORPUS is based on Yang et al. (2011), except we do not use scene type (indoor, outdoor, etc.) as part of the model. At training time, the model has access to the annotated image regions and labels, and to the dependency-parsed version of the English Gigaword Corpus (Napoles et al., 2012). The corpus is used to extract subject–verb–object subtrees, which are then used to predict the best pairs of regions, as well as the verb that relates the regions.

The set of region–region relationships $\mathrm{R} = \{(\mathrm{o}_i, \mathrm{v}_{ij}, \mathrm{o}_j), \dots\}$ is determined by searching for the most likely o_j^*, v^* given an o_i over a set of verbs $\mathcal{V}$ extracted from the corpus and the other regions in the image. This is shown in Equation 1.

$$o_j^*, v^* | o_i = \arg\max_{o_j, v} p(o_i) \cdot p(v|o_i) \cdot p(o_j|v, o_i) \quad (1)$$

We can easily estimate $p(o_i)$, $p(v|o_i)$, and $p(o_j|v, o_i)$ directly from the corpus. If we cannot find an o_j^*, v^* for a region, we back-off to the spatial relationship calculation as defined in PROXIMITY. When we have found the best pairs of regions, we select the most probable pair and generate the first sentence of the description using that pair an template T_1. The second sentence is realised with template T_2 over the subjects in R not used in generating the first sentence. An example of the language generated is:

(2) The man is riding the bike. There is also a car, a road, and trees in the image.

In comparison to PROXIMITY, this model will only describe pairs of regions that have observed relations in the external corpus. The corpus also provides a verb that relates the regions, which produces descriptions that are more in line with human-generated text. However, since noun co-occurrence in the corpus controls which regions can be mentioned in the description, this model will be prone to relating regions simply because their labels occur together frequently in the corpus.

3.3 STRUCTURE

The model STRUCTURE exploits the visual dependency representation of an image to generate language for only the relationships that hold between pairs of regions. It has access to the image regions, the region labels, and the visual dependency representation of an image.

Region–region relationships are generated during a depth-first traversal of the VDR using templates T_1, T_3, T_4, and T_5. The VDR of an image is traversed and language fragments are generated and then combined depending on the number of children of a node in the tree. If a node has only one child then we use T_1 to generate text for the head-child relationship. If a node has more than one child, we need to decide how to order the language generated by the model. We generate sentence fragments using T_4 for each child independently and combine them later. In STRUCTURE, the sentence fragments are sorted by the Euclidean distance of the children from the parent. In order to avoid problematic descriptions such as *"The woman is above the horse is above the field is beside the house"*, we include a special case for when a node has more than one child. In these cases, the nearest region is realized in direct relation to the head using either T_3 (two children) or T_1 (more than two children), and the remaining regions form a separate sentence using T_5. This sorting and combing process would result in *"The woman is above the horse. She is above field and beside the house"* for the case mentioned above.

An example of the type of description that can be generated during a traversal is:

(3) The man is above the bike above the road. There is also a car and trees in the image.

In comparison to PROXIMITY, this model can exploit a representation of an image that encodes the relationships between regions in an image (the VDR). However, it is limited to generating spatial relations, because it cannot predict verbs to relate regions.

3.4 PARALLEL

The model PARALLEL is an extension of STRUCTURE that uses the image descriptions available to

predict verbs that relate regions in parent-child relationships in a VDR. At training time it has access to the annotated regions and labels, the visual dependency representations, and the gold-standard image descriptions. Recall from Section 2.1 that the descriptions were dependency-parsed using the parser of McDonald et al. (2005) and alignments were calculated between the nodes in the VDRs and the words in the parsed image descriptions.

We estimate two distributions from the image descriptions using the alignments: $p(verb|o_{head}, o_{child}, rel_{head-child})$ and $p(verb|o_{head}, o_{child})$. The second distribution is used as a backoff when we do not observe the arc label between the regions in the training data. The generation process is similar to that used in STRUCTURE, with two exceptions: (1) it can generate verbs during the generation steps, and (2) when a node has multiple dependents, the sentence fragments are sorted by the probability of the verb associated with them. This sorting step governs which child is in a relationship with its parent. When the model generates text, it only generates a verb for the most probable sentence fragment. The remaining fragments revert back to spatial relationships to avoid generating language that places the subject region in multiple relationships with other regions. An example of the language generated is:

(4) The man is riding the bike on the road. There is also a car and trees in the image.

In comparison to CORPUS, this model generates descriptions in which the relations between the regions determined by the image itself and not by an external corpus. In comparison to PROXIMITY and STRUCTURE, this model generates descriptions that express meaningful relations between the regions and not simple spatial relationships.

4 Image Parsing

The STRUCTURE and PARALLEL models rely on visual dependency representations, but it is unrealistic to assume gold-standard representations will always be available because they are expensive to construct. In this section we describe an image parser that can induce VDRs automatically from region-annotated images, providing the input for the STRUCTURE-PARSED and PARALLEL-PARSED models at test time.

The parser is based on the arc-factored dependency parsing model of McDonald et al. (2005). This model generates a dependency representation by maximizing the score s computed over all edges of the representation. In our notation, $\mathbf{x}_{vis}$ is the set of annotated regions and y_{vis} is a visual dependency representation of the image; (i, j) is a directed arc from node i to node j in $\mathbf{x}_{vis}$, $\mathbf{f}(i, j)$ is a feature representation of the arc (i, j), and $\mathbf{w}$ is a vector of feature weights to be learned by the model. The overall score of a visual dependency representation is:

$$s(\mathbf{x}_{vis}, y_{vis}) = \sum_{(i,j) \in y_{vis}} \mathbf{w} \cdot \mathbf{f}(i, j) \quad (2)$$

The features in the model are defined over region labels in the visual dependency representation as well as the relationship labels. As our dependency representations are unordered, none of the features encode the linear order of region labels, unlike the feature set of the original model. Unigram features describe how likely individual region labels are to appear as either heads or arguments and bigram feature captures which region labels are in head-argument relationships. All features are conjoined with the relationship label.

We evaluate our parser on the 1,023 visual dependency representations from the data set. The evaluation is run over 10 random splits into 80% training, 10% development, and 10% test data.[2] Performance is measured with labelled and unlabelled directed dependency accuracy. The parser achieves 58.2% $\pm$ 3.1 labelled accuracy and 65.5% $\pm$ 3.3 unlabelled accuracy, significantly better than the baseline of 51.6% $\pm$ 2.5 for both labelled and unlabelled accuracy (the baseline was calculated by attaching all image regions to the root node; this is the most frequent form of attachment in our data).

5 Language Generation Experiments

We evaluate the image description models in an automatic setting and with human judgements. In

[2]Different visual dependency representations of the same image are never split between the training and test data.

	Automatic Evaluation				Human Judgements		
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Grammar	Action	Scene
PARALLEL-PARSED	45.4 ± 2.0	$\mathbf{16.1 \pm 0.9}$	$\mathbf{6.4 \pm 0.7}$	$\mathbf{2.7 \pm 0.5}$	4.2 ± 1.3	$\mathbf{3.3 \pm 1.7}$	3.5 ± 1.3
PROXIMITY	45.1 ± 0.8	$10.2 \pm 1.0^\star$	$2.1 \pm 0.6^\star$	$0.4 \pm 0.2^\star$	$3.7 \pm 1.5^\star$	$2.1 \pm 0.3^\star$	$3.0 \pm 1.4^\star$
CORPUS	46.1 ± 1.1	$12.4 \pm 1.3^\star$	$3.1 \pm 0.8^\star$	$0.7 \pm 0.3^\star$	4.4 ± 1.1	$2.2 \pm 1.3^\star$	3.4 ± 1.3
STRUCTURE	$40.2 \pm 3.0^\star$	$11.5 \pm 1.2^\star$	$3.5 \pm 0.5^\star$	$0.3 \pm 0.1^\star$	4.1 ± 1.4	$2.1 \pm 1.4^\star$	$3.0 \pm 1.4^\star$
STRUCTURE-PARSED	$41.1 \pm 2.1^\star$	$12.2 \pm 0.9^\star$	$3.6 \pm 0.4^\star$	$0.4 \pm 0.2^\star$	4.0 ± 1.4	$1.6 \pm 1.3^\star$	3.2 ± 1.3
PARALLEL	44.6 ± 3.1	16.0 ± 1.5	6.8 ± 1.0	2.9 ± 0.7	$4.5 \pm 1.0^\star$	3.4 ± 1.6	3.7 ± 1.3
GOLD	-	-	-	-	$4.8 \pm 0.4^\star$	$4.8 \pm 0.6^\star$	$4.6 \pm 0.7^\star$

Table 4: Automatic evaluation results averaged over 10 random test splits of the data, and human judgements on the median scoring BLEU-4 test split for PARALLEL. We find significant differences ($^\star p < 0.05$) in the descriptions generated by PARALLEL-PARSED compared to models that operate over an unstructured bag of image regions representation. **Bold** means PARALLEL-PARSED is significantly better than PROXIMITY, CORPUS, and STRUCTURE.

the automatic setting, we follow previous work and measure how close the model-generated descriptions are to the gold-standard descriptions using the BLEU metric. Human judgements were collected from Amazon Mechanical Turk.

5.1 Methodology

The task is to produce a description of an image. The PROXIMITY and CORPUS models have access to gold-standard region labels and region boundaries at test time. The STRUCTURE and PARALLEL models have additional access to the visual dependency representation of the image. These representations are either the gold-standard, or in the case of STRUCTURE-PARSED and PARALLEL-PARSED, produced by the image parser described in Section 4. Table 2 provides a reminder of the information the different models have access to at training time.

Our data set of 1,023 image–description–VDR tuples was randomly split into 10 folds of 80% training data, 10% development data, and 10% test data. The results we report are means computed over the 10 splits. The image parser used for models STRUCTURE-PARSED and PARALLEL-PARSED is trained on the gold-standard VDRs of the training splits, and then predicts VDRs on the development and test splits. Significant differences were measured using a one-way ANOVA with PARALLEL-PARSED as the reference[3], with differences between pairs of mean checked with a Tukey HSD test.

5.2 Automatic Evaluation

The model-generated descriptions are compared against the human-written gold-standard descriptions using the smoothed BLEU measure (Lin and Och, 2004). BLEU is commonly used in machine translation experiments to measure the effective overlap between a reference sentence and a proposed translation sentence. Table 4 shows the results on the test data and Figure 4 shows sample outputs for two images. PARALLEL, the model with access to both image structure and aligned image descriptions at training time outperforms all other models on higher-order BLEU measures. One reason for this improvement is that PARALLEL can formulate sentence fragments that relate the subject, a verb, and an object without trying to predict the best object, unlike CORPUS. The probability associated with each fragment generated for nodes with multiple children also tends to lead to a more accurate order of mentioning image regions. It can also be seen that PARALLEL-PARSED remains significantly better than the other models when the VDRs of images are predicted by an image parser, rather than being gold-standard.

[3]Recall that PARALLEL uses gold-standard VDRs and PARALLEL-PARSED uses the output of the image parser described in Section 4.

The weakest results are obtained from a model that relies on the proximity of regions to generate descriptions. PROXIMITY achieves competitive BLEU-1 scores but this is mostly due to it correctly generating region names and determiners. CORPUS is better than PROXIMITY at correctly producing higher-order n-grams than because it has a better model of the region–region relationships in an image. However, it has difficulties guessing the correct verb for a description, as it relies on corpus co-occurrences for this (see the second example in Table 4). STRUCTURE uses the VDR of an image to generate the description, which this leads to an improvement over PROXIMITY on some of the BLEU metrics; however, it is not sufficient to outperform CORPUS.

5.3 Human Judgements

We conducted a human judgement study on Mechanical Turk to complement the automatic evaluation. Workers were paid $0.05 to rate the quality of an image–description pair generated by one of the models using three criteria on a scale from 1 to 5:

1. Grammaticality: give high scores if the description is correct English and doesn't contain any grammatical mistakes.
2. Action: give high scores if the description correctly describes what people are doing in the image.
3. Scene: give high scores if the description correctly describes the rest of the image (background, other objects, etc).

A total of 101 images were used for this evaluation and we obtained five judgments for each image-description pair, resulting in a total of 3,535 judgments. To ensure a fair evaluation, we chose the images from the split of the data that gave median BLEU-4 accuracy for PARALLEL, the best performing model in the automatic evaluations.

The right side of Table 4 shows the mean judgements for each model for across the three evaluation criteria. The gold-standard descriptions elicited judgements around five, and were significantly better than the model outputs on all aspects. Furthermore, all models produce highly grammatical output, with mean ratings of between 3.7 and 4.5. This can be explained by the fact that the models all relied on templates to ensure grammatical output.

The ratings of the action descriptions reveal the usefulness of structural information. PROXIMITY, CORPUS, and STRUCTURE all perform badly with mean judgements around two, PARALLEL, which uses both image structure and aligned descriptions, significantly outperforms all other models with the exception of PARALLEL-PARSED, which has very similar performance. The fact that PARALLEL and PARALLEL-PARSED perform similarly on all three human measures confirms that automatically parsed VDRs are as useful for image description as gold-standard VDRs.

When we compare the quality of the scene descriptions, we notice that all models perform similarly, around the middle of the scale. This is probably due to the fact that they all have access to gold-standard region labels, which enables them to correctly refer to regions in the scene most of the time. The additional information about the relationships between regions that STRUCTURE and PARALLEL have access to does not improve the quality of the background scene description.

6 Related Work

Previous work on image description can be grouped into three approaches: description-by-retrieval, description using language models, and template-based description. Ordonez et al. (2011), Farhadi et al. (2010), and Kuznetsova et al. (2012) generate descriptions by retrieving the most similar image from a large data set of images paired with descriptions. These approaches are restricted to generating descriptions that are only present in the training set; also, they typically require large amounts of training data and assume images that share similar properties (scene type, objects present) should be described in a similar manner.

Kulkarni et al. (2011) and Li et al. (2011) generate descriptions using n-gram language models trained on a subset of Wikipedia. Both approaches first determine the attributes and relationships between regions in an image as region–preposition–region triples. The disadvantage of relying on region–preposition–region triples is that they cannot distinguish between the main event of the image and the

PROXIMITY	A man is beside a phone. There is also a wall and a sign in the image.
CORPUS	A man is holding a sign. There is also a wall and a phone in the image.
STRUCTURE	A wall is above a wall. A man is beside a sign.
PARALLEL	A man is holding a phone. A wall is beside a sign.
GOLD	A foreign man with sunglasses talking on a cell phone. A large building and a mountain in the background.

PROXIMITY	A beach is above a beach. There are also horses, a woman, and a man in the image.
CORPUS	A woman is outnumbering a man. There are also horses and beaches in the image.
STRUCTURE	A man is beside a woman above a horse. A horse is beside a woman beside a beach.
PARALLEL	A man is riding a horse above a beach. A horse is beside a beach beside a woman.
GOLD	There is a man and women both on horses. They are on a beach during the day.

Figure 4: Some example descriptions produced by PROXIMITY, CORPUS, STRUCTURE and PARALLEL.

background regions. Kulkarni et al. (2011) is closely related to our PROXIMITY baseline.

Yang et al. (2011) fill in a sentence template by selecting the likely objects, verbs, prepositions, and scene types based on a Hidden Markov Model. Verbs are generated by finding the most likely pairing of object labels in an external corpus. This model is closely related to our CORPUS baseline. Mitchell et al. (2012) over-generates syntactically well-formed sentence fragments and then recombines these using a tree-substitution grammar.

Previous research has relied extensively on automatically detecting object regions in an image using state-of-the art object detectors (Felzenszwalb et al., 2010). We use gold-standard region annotations to remove this noisy component from the description generation pipeline, allowing us to focus on the utility of image structure for description generation.

7 Conclusion

In this paper we introduced a novel representation of an image as a set of dependencies over its annotated regions. This *visual dependency representation* encodes which regions are related to each other in an image, and can be used to infer the action or event that is depicted. We found that image description models based on visual dependency representations significantly outperform competing models in both automatic and human evaluations. We showed that visual dependency representations can be induced automatically using a standard dependency parser and that the descriptions generated from the induced representations are as good as the ones generated from gold-standard representations. Future work will focus on improvements to the image parser, on exploring this representation in open-domain data sets, and on using the output of an object detector to obtain a fully automated model.

Acknowledgments

The authors would like to thank M. Lapata and S. Frank for feedback on an earlier draft of the paper and the anonymous reviewers for their feedback. A. M. Enoch, N. Ghahremani-Azghandi, L. S. McAlpine, and K. Tsagkaridis helped annotate the data. The research presented here was supported by the European Research Council under award 203427 Synchronous Linguistic and Visual Processing.

References

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2011. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: generating sentences from images. In *ECCV '10*, pages 15–29, Heraklion, Crete, Greece.

P F Felzenszwalb, R B Girshick, D McAllester, and D Ramanan. 2010. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *CVPR '11*, pages 1601–1608, Colorado Springs, Colorado, U.S.A.

Polina Kuznetsova, Vicente Ordonez, Alexander C. Berg, Tamara L. Berg, and Yejin Choi. 2012. Collective Generation of Natural Image Descriptions. In *ACL '12*, pages 359–368, Jeju Island, South Korea.

Siming Li, Girish Kulkarni, Tamara L. Berg, Alexander C. Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In *CoNLL '11*, pages 220–228, Portland, Oregon, U.S.A.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL '04*, pages 605–612, Barcelona, Spain.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL '05*, pages 91–98, University of Michigan, U.S.A.

Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Alyssa Mensch, Alex Berg, Tamara Berg, and Hal Daum. 2012. Midge : Generating Image Descriptions From Computer Vision Detections. In *EACL '12*, pages 747–756, Avignon, France.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *AKBC-WEKEX Workshop at NAACL-HLT '12*, Montreal, Canada.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2Text: Describing Images Using 1 Million Captioned Photographs. In *NIPS 24*, Granada, Spain.

Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. 2008. LabelMe: A Database and Web-Based Tool for Image Annotation. *IJCV*, 77(1-3):157–173.

Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. 2011. Corpus-Guided Sentence Generation of Natural Images. In *EMNLP '11*, pages 444–454, Edinburgh, Scotland, UK.

Semi-supervised Feature Transformation for Dependency Parsing

Wenliang Chen[†], **Min Zhang**[†,*], **and Yue Zhang**[‡]
[†]School of Computer Science and Technology, Soochow University, China
[‡]Singapore University of Technology and Design, Singapore
{wlchen, mzhang}@suda.edu.cn
yue_zhang@sutd.edu.sg

Abstract

In current dependency parsing models, conventional features (i.e. base features) defined over surface words and part-of-speech tags in a relatively high-dimensional feature space may suffer from the data sparseness problem and thus exhibit less discriminative power on unseen data. In this paper, we propose a novel semi-supervised approach to addressing the problem by transforming the base features into high-level features (i.e. meta features) with the help of a large amount of automatically parsed data. The meta features are used together with base features in our final parser. Our studies indicate that our proposed approach is very effective in processing unseen data and features. Experiments on Chinese and English data sets show that the final parser achieves the best-reported accuracy on the Chinese data and comparable accuracy with the best known parsers on the English data.

1 Introduction

In recent years, supervised learning models have achieved lots of progress in the dependency parsing task, as can be found in the CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). The supervised models take annotated data as training data, utilize features defined over surface words, part-of-speech tags, and dependency trees, and learn the preference of features via adjusting feature weights.

In the supervised learning scenarios, many previous studies explore rich feature representation that leads to significant improvements. McDonald and Pereira (2006) and Carreras (2007) define second-order features over two adjacent arcs in second-order graph-based models. Koo and Collins (2010) use third-order features in a third-order graph-based model. Bohnet (2010) considers information of more surrounding words for the graph-based models, while Zhang and Nivre (2011) define a set of rich features including the word valency and the third-order context features for transition-based models. All these models utilize richer and more complex feature representations and achieve better performance than the earlier models that utilize the simpler features (McDonald et al., 2005; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004). However, the richer feature representations result in a high-dimensional feature space. Features in such a space may suffer from the data sparseness problem and thus have less discriminative power on unseen data. If input sentences contain unknown features that are not included in training data, the parsers can usually give lower accuracy.

Several methods have been proposed to alleviate this problem by using large amounts of unannotated data, ranging from self-training and co-training (McClosky et al., 2006; Sagae and Tsujii, 2007) to more complex methods that collect statistical information from unannotated sentences and use them as additional features (Koo et al., 2008; Chen et al., 2009).

In this paper, we propose an alternative approach to semi-supervised dependency parsing via feature transformation (Ando and Zhang, 2005). More

[*]Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1303–1313,
Seattle, Washington, USA, 18-21 October 2013.

specifically, we transform base features to a higher-level space. The base features defined over surface words, part-of-speech tags, and dependency trees are high dimensional and have been explored in the above previous studies. The higher-level features, which we call meta features, are low dimensional, and newly defined in this paper. The key idea behind is that we build connections between known and unknown base features via the meta features. From another viewpoint, we can also interpret the meta features as a way of doing feature smoothing.

Our feature transfer method is simpler than that of Ando and Zhang (2005), which is based on splitting the original problem into multiple auxiliary problems. In our approach, the base features are grouped and each group relates to a meta feature. In the first step, we use a baseline parser to parse a large amount of unannotated sentences. Then we collect the base features from the parse trees. The collected features are transformed into predefined discrete values via a transformation function. Based on the transformed values, we define a set of meta features. Finally, the meta features are incorporated directly into parsing models.

To demonstrate the effectiveness of the proposed approach, we apply it to the graph-based parsing models (McDonald and Nivre, 2007). We conduct experiments on the standard data split of the Penn English Treebank (Marcus et al., 1993) and the Chinese Treebank Version 5.1 (Xue et al., 2005). The results indicate that the approach significantly improves the accuracy. In summary, we make the following contributions:

- We define a simple yet useful transformation function to transform base features to meta features automatically. The meta features build connections between known and unknown base features, and relieve the data sparseness problem.
- Compared to the base features, the number of meta features is remarkably small.
- We build semi-supervised dependency parsers that achieve the best accuracy on the Chinese data and comparable accuracy with the best known systems on the English data.

The rest of this paper is organized as follows. Section 2 introduces the graph-based parsing model. Section 3 describes the meta features and meta parser. Section 4 describes the experiment settings and reports the experimental results on English and Chinese data sets. Section 5 discusses related work. Finally, in Section 6 we summarize the proposed approach.

2 Baseline parser

In this section, we introduce a graph-based parsing model proposed by McDonald et al. (2005) and build a baseline parser.

2.1 Graph-based parsing model

Given an input sentence, dependency parsing is to build a dependency tree. We define X as the set of possible input sentences, Y as the set of possible dependency trees, and $D = (x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)$ as a training set of n pairs of $x_i \in X$ and $y_i \in Y$. A sentence is denoted by $x = (w_0, w_1, ..., w_i, ..., w_m)$, where w_0 is ROOT and does not depend on any other word and w_i refers to a word.

In the graph-based model, we define ordered pair $(w_i, w_j) \in y$ as a dependency relation in tree y from word w_i to word w_j (w_i is the head and w_j is the dependent), G_x as a graph that consists of a set of nodes $V_x = \{w_0, w_1, ..., w_i, ..., w_m\}$ and a set of arcs (edges) $E_x = \{(w_i, w_j) | i \neq j, w_i \in V_x, w_j \in (V_x - \{w_0\})\}$. The parsing model of McDonald et al. (2005) is to search for the maximum spanning tree (MST) in graph G_x. We denote $Y(G_x)$ as the set of all the subgraphs of G_x that are valid dependency trees (McDonald and Nivre, 2007) for sentence x.

We define the score of a dependency tree $y \in Y(G_x)$ to be the sum of the subgraph scores,

$$score(x, y) = \sum_{g \in y} score(x, g) \quad (1)$$

where g is a spanning subgraph of y, which can be a single arc or adjacent arcs. In this paper we assume the dependency tree to be a spanning projective tree. The model scores each subgraph using a linear representation. Then scoring function $score(x, g)$ is,

$$score(x, g) = \mathbf{f}(x, g) \cdot \mathbf{w} \quad (2)$$

where $\mathbf{f}(x, g)$ is a high-dimensional feature vector based on features defined over g and x and $\mathbf{w}$ refers to the weights for the features.

The maximum spanning tree is the highest scoring tree in $Y(G_x)$. The task of decoding algorithms in the parsing model for an input sentence x is to find y^*, where

$$\begin{aligned} y^* &= \arg\max_{y \in Y(G_x)} score(x, y) \\ &= \arg\max_{y \in Y(G_x)} \sum_{g \in y} score(x, g) \\ &= \arg\max_{y \in Y(G_x)} \sum_{g \in y} \mathbf{f}(x, g) \cdot \mathbf{w} \end{aligned} \quad (3)$$

In our system, we use the decoding algorithm proposed by Carreras (2007), which is a second-order CKY-style algorithm (Eisner, 1996) and feature weights **w** are learned during training using the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; McDonald et al., 2005).

2.2 Base features

Previous studies have defined different sets of features for the graph-based parsing models, such as the first-order features defined in McDonald et al. (2005), the second-order parent-siblings features defined in McDonald and Pereira (2006), and the second-order parent-child-grandchild features defined in Carreras (2007). Bohnet (2010) explorers a richer set of features than the above sets. We further extend the features defined by Bohnet (2010) by introducing more lexical features as the base features. The base feature templates are listed in Table 1, where h, d refer to the head, the dependent respectively, c refers to d's sibling or child, b refers to the word between h and d, $+1$ (-1) refers to the next (previous) word, w and p refer to the surface word and part-of-speech tag respectively, $[wp]$ refers to the surface word or part-of-speech tag, $d(h, d)$ is the direction of the dependency relation between h and d, and $d(h, d, c)$ is the directions of the relation among h, d, and c. We generate the base features based on the above templates.

2.3 Baseline parser

We train a parser with the base features as the Baseline parser. We define $\mathbf{f}_b(x, g)$ as the base features and $\mathbf{w}_b$ as the corresponding weights. The scoring function becomes,

$$score(x, g) = \mathbf{f}_b(x, g) \cdot \mathbf{w}_b \quad (4)$$

3 Meta features

In this section, we propose a semi-supervised approach to transform the features in the base feature space (F_B) to features in a higher-level space (F_M) with the following properties:

- The features in F_M are able to build connections between known and unknown features in F_B and therefore should be highly informative.
- The transformation should be learnable based on a labeled training set and an automatically parsed data set, and automatically computable for the test sentences.

The features in F_M are referred to as meta features. In order to perform the feature transformation, we choose to define a simple yet effective mapping function. Based on the mapped values, we define feature templates for generating the meta features. Finally, we build a new parser with the base and meta features.

3.1 Template-based mapping function

We define a template-based function for mapping the base features to predefined discrete values. We first put the base features into several groups and then perform mapping.

We have a set of base feature templates T_B. For each template $T_i \in T_B$, we can generate a set of base features F_i from dependency trees in the parsed data, which is automatically parsed by the Baseline parser. We collect the features and count their frequencies. The collected features are sorted in decreasing order of frequencies. The mapping function for a base feature f_b of F_i is defined as follows,

$$\Phi(f_b) = \begin{cases} H_i & \text{if } R(f_b) \leq \text{TOP10} \\ M_i & \text{if TOP10} < R(f_b) \leq \text{TOP30} \\ L_i & \text{if TOP30} < R(f_b) \\ O_i & \text{Others} \end{cases}$$

where $R(f_b)$ is the position number of f_b in the sorted list, "Others" is defined for the base features that are not included in the list, and TOP10 and TOP 30 refer to the position numbers of top 10% and top 30% respectively. The numbers, 10% and 30%, are tuned on the development sets in the experiments. For a base feature generated from template T_i, we have four possible values: H_i, M_i, L_i, and O_i. In

(a) First-order standard

$h_{[wp]}$, $d_{[wp]}$, d(h,d)
$h_{[wp]}$, d(h,d)
d_w, d_p, d(h,d)
$d_{[wp]}$, d(h,d)
h_w, h_p, d_w, d_p, d(h,d)
h_p, h_w, d_p, d(h,d)
h_w, d_w, d_p, d(h,d)
h_w, h_p, $d_{[wp]}$, d(h,d)

(b) First-order Linear

h_p, b_p, d_p, d(h,d)
h_p, h_{+1p}, d_{-1p}, d_p, d(h,d)
h_{-1p}, h_p, d_{-1p}, d_p, d(h,d)
h_p, h_{+1p}, d_p, d_{+1p}, d(h,d)
h_{-1p}, h_p, d_p, d_{+1p}, d(h,d)

(c) Second-order standard

h_p, d_p, c_p, d(h,d,c)
h_w, d_w, c_w, d(h,d,c)
h_p, $c_{[wp]}$, d(h,d,c)
d_p, $c_{[wp]}$, d(h,d,c)
h_w, $c_{[wp]}$, d(h,d,c)
d_w, $c_{[wp]}$, d(h,d,c)

(d) Second-order Linear

$h_{[wp]}$, $h_{+1[wp]}$, $c_{[wp]}$, d(h,d,c)
$h_{-1[wp]}$, $h_{[wp]}$, $c_{[wp]}$, d(h,d,c)
$h_{[wp]}$, $c_{-1[wp]}$, $c_{[wp]}$, d(h,d,c)
$h_{[wp]}$, $c_{[wp]}$, $c_{+1[wp]}$, d(h,d,c)
$h_{-1[wp]}$, $h_{[wp]}$, $c_{-1[wp]}$, $c_{[wp]}$, d(h,d,c)
$h_{[wp]}$, $h_{+1[wp]}$, $c_{-1[wp]}$, $c_{[wp]}$, d(h,d,c)
$h_{-1[wp]}$, $h_{[wp]}$, $c_{[wp]}$, $c_{+1[wp]}$, d(h,d,c)
$h_{[wp]}$, $h_{+1[wp]}$, $c_{[wp]}$, $c_{+1[wp]}$, d(h,d,c)
$d_{[wp]}$, $d_{+1[wp]}$, $c_{[wp]}$, d(h,d,c)
$d_{-1[wp]}$, $d_{[wp]}$, $c_{[wp]}$, d(h,d,c)
$d_{[wp]}$, $c_{-1[wp]}$, $c_{[wp]}$, d(h,d,c)
$d_{[wp]}$, $c_{[wp]}$, $c_{+1[wp]}$, d(h,d,c)
$d_{[wp]}$, $d_{+1[wp]}$, $c_{-1[wp]}$, $c_{[wp]}$, d(h,d,c)
$d_{[wp]}$, $d_{+1[wp]}$, $c_{[wp]}$, $c_{+1[wp]}$, d(h,d,c)
$d_{-1[wp]}$, $d_{[wp]}$, $c_{-1[wp]}$, $c_{[wp]}$, d(h,d,c)
$d_{-1[wp]}$, $d_{[wp]}$, $c_{[wp]}$, $c_{+1[wp]}$, d(h,d,c)

Table 1: Base feature templates

total, we have $4 \times N(T_B)$ possible values for all the base features, where $N(T_B)$ refers to the number of the base feature templates, which is usually small. We can obtain the mapped values of all the collected features via the mapping function.

3.2 Meta feature templates

Based on the mapped values, we define meta feature templates in F_M for dependency parsing. The meta feature templates are listed in Table 2, where f_b is a base feature of F_B, h_p refers to the part-of-speech tag of the head and h_w refers to the surface word of the head. Of the table, the first template uses the mapped value only, the second and third templates combine the value with the head information. The number of the meta features is relatively small. It has $4 \times N(T_B)$ for the first type, $4 \times N(T_B) \times N(POS)$ for the second type, and $4 \times N(T_B) \times N(WORD)$ for the third one, where $N(POS)$ refers to the number of part-of-speech tags, $N(WORD)$ refers to the number of words. We remove any feature related to the surface form if the word is not one of the Top-N most frequent words in the training data. We used N=1000 for the experiments for this paper. This method can reduce the size of the feature sets. The empirical statistics of the feature sizes at Section 4.2.2 shows that the size of meta features is only 1.2% of base features.

$[\Phi(f_b)]$
$[\Phi(f_b)], h_p$
$[\Phi(f_b)], h_w$

Table 2: Meta feature templates

3.3 Generating meta features

We use an example to demonstrate how to generate the meta features based on the meta feature templates in practice. Suppose that we have sentence "I ate the meat with a fork." and want to generate the meta features for the relation among "ate", "meat", and "with", where "ate" is the head, "meat" is the dependent, and "with" is the closest left sibling of "meat". Figure 1 shows the example.

We demonstrate the generating procedure using template T_k = "$h_w, d_w, c_w, d(h, d, c)$" (the second template of Table 1-(c)), which contains the surface forms of the head, the dependent, its sibling, and the directions of the dependencies among h, d, and c. We can have a base feature "ate, meat, with, RIGHTSIB", where "RIGHTSIB" refers to the parent-siblings structure with the right direction. In the auto-parsed data, this feature occurs 200 times and ranks between TOP10 and TOP30. Accord-

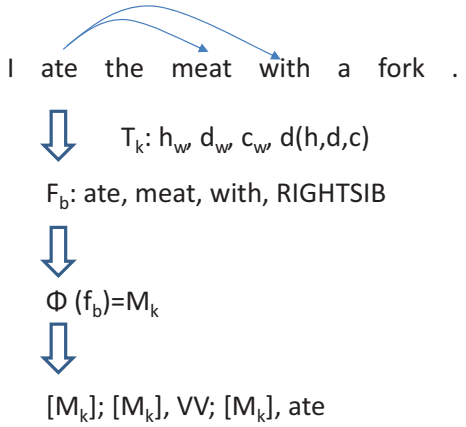


Figure 1: An example of generating meta features

ing to the mapping function, we obtain the mapped value M_k. Finally, we have the three meta features "$[M_k]$", "$[M_k], VV$", and "$[M_k], ate$", where VV is the part-of-speech tag of word "ate". In this way, we can generate all the meta features for the graph-based model.

3.4 Meta parser

We combine the base features with the meta features by a new scoring function,

$$score(x, g) = \mathbf{f}_b(x, g) \cdot \mathbf{w}_b + \mathbf{f}_m(x, g) \cdot \mathbf{w}_m \quad (5)$$

where $\mathbf{f}_b(x, g)$ refers to the base features, $\mathbf{f}_m(x, g)$ refers to the meta features, and $\mathbf{w}_b$ and $\mathbf{w}_m$ are their corresponding weights respectively. The feature weights are learned during training using MIRA (Crammer and Singer, 2003; McDonald et al., 2005). Note that $\mathbf{w}_b$ is also retrained here.

We use the same decoding algorithm in the new parser as in the Baseline parser. The new parser is referred to as the meta parser.

4 Experiments

We evaluated the effect of the meta features for the graph-based parsers on English and Chinese data.

4.1 Experimental settings

In our experiments, we used the Penn Treebank (PTB) (Marcus et al., 1993) for English and the Chinese Treebank version 5.1 (CTB5) (Xue et al., 2005) for Chinese. The tool "Penn2Malt"[1] was used to convert the data into dependency structures with the English head rules of Yamada and Matsumoto (2003) and the Chinese head rules of Zhang and Clark (2008). We followed the standard data splits as shown in Table 3. Following the work of Koo et al. (2008), we used a tagger trained on training data to provide part-of-speech (POS) tags for the development and test sets, and used 10-way jackknifing to generate part-of-speech tags for the training set. We used the MXPOST (Ratnaparkhi, 1996) tagger for English and the CRF-based tagger for Chinese. We used gold standard segmentation in the CTB5. The data partition of Chinese were chosen to match previous work (Duan et al., 2007; Li et al., 2011; Hatori et al., 2011).

	train	dev	test
PTB (sections)	2-21	22	23
CTB5 (files)	001-815 1001-1136	886-931 1148-1151	816-885 1137-1147

Table 3: Standard data splits

For the unannotated data in English, we used the BLLIP WSJ corpus (Charniak et al., 2000) containing about 43 million words.[2] We used the MXPOST tagger trained on the training data to assign part-of-speech tags and used the Baseline parser to process the sentences of the Brown corpus. For the unannotated data in Chinese, we used the Xinhua portion of Chinese Gigaword[3] Version 2.0 (LDC2009T14) (Huang, 2009), which has approximately 311 million words. We used the MMA system (Kruengkrai et al., 2009) trained on the training data to perform word segmentation and POS tagging and used the Baseline parser to parse the sentences in the Gigaword data.

In collecting the base features, we removed the features which occur only once in the English data and less than four times in the Chinese data. The feature occurrences of one time and four times are based on the development data performance.

We measured the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of to-

[1] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

[2] We ensured that the text used for building the meta features did not include the sentences of the Penn Treebank.

[3] We excluded the sentences of the CTB data from the Gigaword data.

kens (excluding all punctuation tokens) with the correct HEAD. We also reported the scores on complete dependency trees evaluation (COMP).

4.2 Feature selection on development sets

We evaluated the parsers with different settings on the development sets to select the meta features.

4.2.1 Different models vs meta features

In this section, we investigated the effect of different types of meta features for the models trained on different sizes of training data on English.

There are too many base feature templates to test one by one. We divided the templates into several categories. Of Table 1, some templates are only related to part-of-speech tags (P), some are only related to surface words (W), and the others contain both part-of-speech tags and surfaces (M). Table 4 shows the categories, where numbers $[1-4]$ refer to the numbers of words involved in templates. For example, the templates of N3WM are related to three words and contain the templates of W and M. Based on different categories of base templates, we have different sets of meta features.[4]

Category	Example
N1P	$h_p, d(h,d)$
N1WM	$h_w, d(h,d)$; $h_w, h_p, d(h,d)$
N2P	$h_p, d_p, d(h,d)$
N2WM	$h_w, d_w, d(h,d)$; $h_w, d_p, d(h,d)$
N3P	$h_p, d_p, c_p, d(h,d,c)$
N3WM	$h_w, d_w, c_w, d(h,d,c)$; $d_w, d_{+1p}, c_p, d(h,d,c)$
N4P	$h_p, h_{+1p}, c_p, c_{+1p}, d(h,d,c)$
N4WM	$h_w, h_{+1w}, c_w, c_{+1w}, d(h,d,c)$; $h_w, h_{+1p}, c_p, c_{+1p}, d(h,d,c)$

Table 4: Categories of base feature templates

We randomly selected 1% and 10% of the sentences respectively from the training data. We trained the POS taggers and Baseline parsers on these small training data and used them to process the unannotated data. Then, we generated the meta features based on the newly auto-parsed data. The meta parsers were trained on the different subsets of the training data with different sets of meta features. Finally, we have three meta parsers: MP1, MP10, MPFULL, which were trained on 1%, 10% and 100% of the training data.

	MP1	MP10	MPFULL
Baseline	82.22	89.50	93.01
+N1P	82.42	89.48	93.08
+N1WM	82.80	89.42	93.19
+N2P	81.29	89.01	93.02
+N2WM	82.69	90.10	93.23
+N3P	83.32	89.73	93.05
+N3WM	84.47	90.75	93.80
+N4P	82.73	89.48	93.01
+N4WM	84.07	90.42	93.67
OURS	85.11	91.14	93.91

Table 5: Effect of different categories of meta features

Table 5 shows the results, where we add each category of Table 4 individually. From the table, we found that the meta features that are only related to part-of-speech tags did not always help, while the ones related to the surface words were very helpful. We also found that MP1 provided the largest relative improvement among the three settings. These suggested that the more sparse the base features were, the more effective the corresponding meta features were. Thus, we built the final parsers by adding the meta features of N1WM, N2WM, N3WM, and N4WM. The results showed that OURS achieved better performance than the systems with individual sets of meta features.

4.2.2 Different meta feature types

In Table 2, there are three types of meta feature templates. Here, the results of the parsers with different settings are shown in Table 6, where CORE refers to the first type, WithPOS refers to the second one, and WithWORD refers to the third one. The results showed that with all the types the parser (OURS) achieved the best. We also counted the numbers of the meta features. Only 327,864 (or 1.2%) features were added into OURS. Thus, we used all the three types of meta features in our final meta parsers.

[4]We also tested the settings of dividing WM into two subtypes: W and M. The results showed that both two sub-types provided positive results. To simplify, we merged W and M into one category WM.

System	NumOfFeat	UAS
Baseline	27,119,354	93.01
+CORE	+498	93.84
+WithPOS	+14,993	93.82
+WithWORD	+312,373	93.27
OURS	+327,864	93.91

Table 6: Numbers of meta features

4.3 Main results on test sets

We then evaluated the meta parsers on the English and Chinese test sets.

4.3.1 English

The results are shown in Table 7, where MetaParser refers to the meta parser. We found that the meta parser outperformed the baseline with an absolute improvement of 1.01 points (UAS). The improvement was significant in McNemar's Test ($p < 10^{-7}$).

	UAS	COMP
Baseline	92.76	48.05
MetaParser	93.77	51.36

Table 7: Main results on English

4.3.2 Chinese

	UAS	COMP
Baseline	81.01	29.71
MetaParser	83.08	32.21

Table 8: Main results on Chinese

The results are shown in Table 8. As in the experiment on English, the meta parser outperformed the baseline. We obtained an absolute improvement of 2.07 points (UAS). The improvement was significant in McNemar's Test ($p < 10^{-8}$).

In summary, Tables 7 and 8 convincingly show the effectiveness of our proposed approach.

4.4 Different sizes of unannotated data

Here, we considered the improvement relative to the sizes of the unannotated data used to generate the meta features. We randomly selected the 0.1%, 1%, and 10% of the sentences from the full data. Table 9 shows the results, where P0.1, P1, and P10 correspond to 0.1%, 1%, and 10% respectively. From the table, we found that the parsers obtained more benefits as we used more raw sentences. We also tried generating the meta features from the training data only, shown as TrainData in Table 9. However, the results shows that the parsers performed worse than the baselines. This is not surprising because only the known base features are included in the training data.

	English	Chinese
Baseline	92.76	81.01
TrainData	91.93	80.40
P0.1	92.82	81.58
P1	93.14	82.23
P10	93.48	82.81
FULL	93.77	83.08

Table 9: Effect of different sizes of auto-parsed data

4.5 Comparison with previous work

4.5.1 English

Table 10 shows the performance of the previous systems that were compared, where McDonald06 refers to the second-order parser of McDonald and Pereira (2006), Koo10 refers to the third-order parser with model1 of Koo and Collins (2010), Zhang11 refers to the parser of Zhang and Nivre (2011), Li12 refers to the unlabeled parser of Li et al. (2012), Koo08 refers to the parser of Koo et al. (2008), Suzuki09 refers to the parser of Suzuki et al. (2009), Chen09 refers to the parser of Chen et al. (2009), Zhou11 refers to the parser of Zhou et al. (2011), Suzuki11 refers to the parser of Suzuki et al. (2011), and Chen12 refers to the parser of Chen et al. (2012).

The results showed that our meta parser outperformed most of the previous systems and obtained the comparable accuracy with the best result of Suzuki11 (Suzuki et al., 2011) which combined the clustering-based word representations of Koo et al. (2008) and a condensed feature representation. However, our approach is much simpler than theirs and we believe that our meta parser can be further improved by combining their methods.

Type	System	UAS	COMP
Sup	McDonald06	91.5	
	Koo10	93.04	-
	Zhang11	92.9	48.0
	Li12	93.12	-
	Our Baseline	92.76	48.05
Semi	Koo08	93.16	
	Suzuki09	93.79	
	Chen09	93.16	47.15
	Zhou11	92.64	46.61
	Suzuki11	94.22	-
	Chen12	92.76	-
	MetaParser	93.77	51.36

Table 10: Relevant results for English. Sup denotes the supervised parsers, Semi denotes the parsers with semi-supervised methods.

4.5.2 Chinese

Table 11 shows the comparative results, where Li11 refers to the parser of Li et al. (2011), Hatori11 refers to the parser of Hatori et al. (2011), and Li12 refers to the unlabeled parser of Li et al. (2012). The reported scores on this data were produced by the supervised learning methods and our Baseline (supervised) parser provided the comparable accuracy. We found that the score of our meta parser for this data was the best reported so far and significantly higher than the previous scores. Note that we used the auto-assigned POS tags in the test set to match the above previous studies.

System	UAS	COMP
Li11	80.79	29.11
Hatori11	81.33	29.90
Li12	81.21	-
Our Baseline	81.01	29.71
MetaParser	83.08	32.21

Table 11: Relevant results for Chinese

4.6 Analysis

Here, we analyzed the effect of the meta features on the data sparseness problem.

We first checked the effect of unknown features on the parsing accuracy. We calculated the number of unknown features in each sentence and computed the average number per word. The average numbers were used to eliminate the influence of varied sentence sizes. We sorted the test sentences in increasing orders of these average numbers, and divided equally into five bins. BIN 1 is assigned the sentences with the smallest numbers and BIN 5 is with the largest ones. Figure 2 shows the average accuracy scores of the Baseline parsers against to the bins. From the figure, we found that for both two languages the Baseline parsers performed worse while the sentences contained more unknown features.

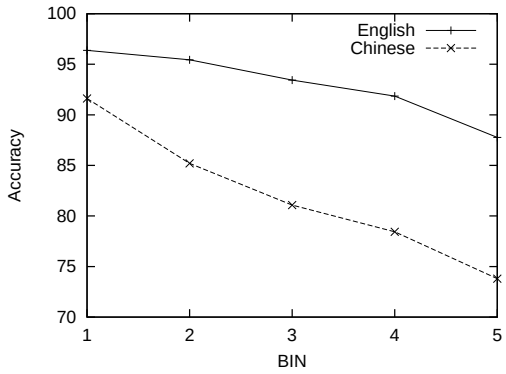


Figure 2: Accuracies relative to numbers of unknown features (average per word) by Baseline parsers

Then, we investigated the effect of the meta features. We calculated the average number of **active meta features** per word that were transformed from the unknown features for each sentence. We sorted the sentences in increasing order of the average numbers of active meta features and divided them into five bins. BIN 1 is assigned the sentences with the smallest numbers and BIN 5 is with the largest ones. Figures 3 and 4 show the results, where "Better" is for the sentences where the meta parsers provided better results than the baselines and "Worse" is for those where the meta parsers provided worse results. We found that the gap between "Better" and "Worse" became larger while the sentences contain more active meta features for the unknown features. The gap means performance improvement. This indicates that the meta features are very effective in processing the unknown features.

5 Related work

Our approach is to use unannotated data to generate the meta features to improve dependency parsing.

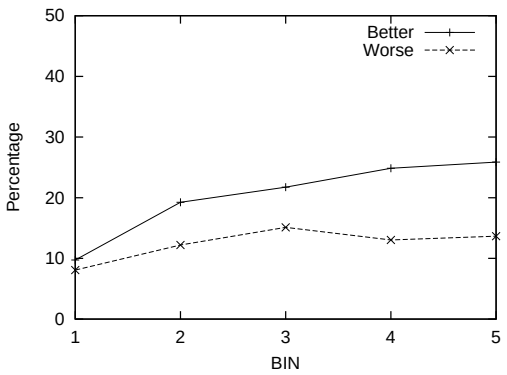


Figure 3: Improvement relative to numbers of active meta features on English (average per word)

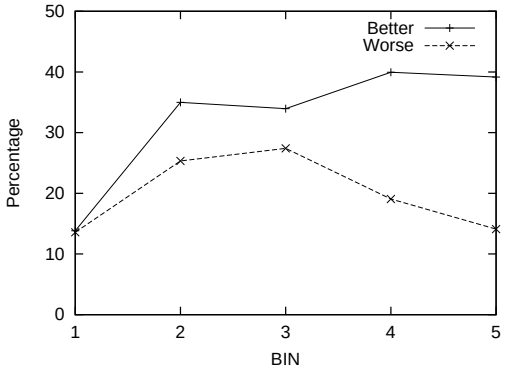


Figure 4: Improvement relative to numbers of active meta features on Chinese (average per word)

Several previous studies relevant to our approach have been conducted.

Koo et al. (2008) used a word clusters trained on a large amount of unannotated data and designed a set of new features based on the clusters for dependency parsing models. Chen et al. (2009) extracted subtree structures from a large amount of data and represented them as the additional features to improve dependency parsing. Suzuki et al. (2009) extended a Semi-supervised Structured Conditional Model (SS-SCM) of Suzuki and Isozaki (2008) to the dependency parsing problem and combined their method with the word clustering feature representation of Koo et al. (2008). Chen et al. (2012) proposed an approach to representing high-order features for graph-based dependency parsing models using a dependency language model and beam search. In future work, we may consider to combine their methods with ours to improve performance.

Several previous studies used co-training/self-training methods. McClosky et al. (2006) presented a self-training method combined with a reranking algorithm for constituency parsing. Sagae and Tsujii (2007) applied the standard co-training method for dependency parsing. In their approaches, some automatically parsed sentences were selected as new training data, which was used together with the original labeled data to retrain a new parser. We are able to use their approaches on top of the output of our parsers.

With regard to feature transformation, the work of Ando and Zhang (2005) is similar in spirit to our work. They studied semi-supervised text chunking by using a large projection matrix to map sparse base features into a small number of high level features. Their project matrix was trained by transforming the original problem into a large number of auxiliary problems, obtaining training data for the auxiliary problems by automatically labeling raw data and using alternating structure optimization to estimate the matrix across all auxiliary tasks. In comparison with their approach, our method is simpler in the sense that we do not request any intermediate step of splitting the prediction problem, and obtain meta features directly from self-annotated data. The training of our meta feature values is highly efficient, requiring the collection of simple statistics over base features from huge amount of data. Hence our method can potentially be useful to other tasks also.

6 Conclusion

In this paper, we have presented a simple but effective semi-supervised approach to learning the meta features from the auto-parsed data for dependency parsing. We build a meta parser by combining the meta features with the base features in a graph-based model. The experimental results show that the proposed approach significantly improves the accuracy. Our meta parser achieves comparable accuracy with the best known parsers on the English data (Penn English Treebank) and the best accuracy on the Chinese data (Chinese Treebank Version 5.1) so far. Further analysis indicate that the meta features are very effective in processing the unknown features. The idea described in this paper is general and can be applied to other NLP applications, such as part-

of-speech tagging and Chinese word segmentation, in future work.

Acknowledgments

This study was started when Wenliang Chen and Min Zhang were members of the Department of Human Language Technology, Institute for Infocomm Research, Singapore. Wenliang Chen was funded partially by the National Science Foundation of China (61203314) and Yue Zhang was supported by MOE grant 2012-T2-2-163. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

R.K. Ando and T. Zhang. 2005. A high-performance semi-supervised learning method for text chunking. *ACL*.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL-X*. SIGNLL.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June. Association for Computational Linguistics.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium*.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of EMNLP 2009*, pages 570–579, Singapore, August.

Wenliang Chen, Min Zhang, and Haizhou Li. 2012. Utilizing dependency language models for graph-based dependency parsing models. In *Proceedings of ACL 2012*, Korea, July.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, Warsaw, Poland.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING1996*, pages 340–345.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Chu-Ren Huang. 2009. Tagged Chinese Gigaword Version 2.0, LDC2009T14. *Linguistic Data Consortium*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL 2010*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of ACL-IJCNLP2009*, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of EMNLP 2011*, UK, July.

Zhenghua Li, Min Zhang, Wanxiang Che, and Ting Liu. 2012. A separately passive-aggressive training algorithm for joint pos tagging and dependency parsing. In *Proceedings of the 24rd International Conference on Computational Linguistics (Coling 2012)*, Mumbai, India. Coling 2012 Organizing Committee.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguisticss*, 19(2):313–330.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of Coling-ACL*, pages 337–344.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, pages 122–131.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98. Association for Computational Linguistics.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING)*, pages 64–70.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*, pages 133–142.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using Giga-word scale unlabeled data. In *Proceedings of ACL-08: HLT*, pages 665–673, Columbus, Ohio, June. Association for Computational Linguistics.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of EMNLP2009*, pages 551–560, Singapore, August. Association for Computational Linguistics.

Jun Suzuki, Hideki Isozaki, and Masaaki Nagata. 2011. Learning condensed feature representations from large unsupervised data sets for supervised learning. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 636–641, Portland, Oregon, USA, June. Association for Computational Linguistics.

Nianwen Xue, Fei Xia, Fu dong Chiou, and Martha Palmer. 2005. Building a Large Annotated Chinese Corpus: the Penn Chinese Treebank. *Journal of Natural Language Engineering*, 11(2):207–238.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT 2003*, pages 195–206.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP 2008*, pages 562–571, Honolulu, Hawaii, October.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL-HLT2011*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.

Guangyou Zhou, Jun Zhao, Kang Liu, and Li Cai. 2011. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of ACL-HLT2011*, pages 1556–1565, Portland, Oregon, USA, June. Association for Computational Linguistics.

Leveraging lexical cohesion and disruption for topic segmentation

Anca Şimon
Université de Rennes 1
IRISA & INRIA Rennes

Guillaume Gravier
CNRS
IRISA & INRIA Rennes

Pascale Sébillot
INSA de Rennes
IRISA & INRIA Rennes

anca-roxana.simon@irisa.fr
guillaume.gravier@irisa.fr
pascale.sebillot@irisa.fr

Abstract

Topic segmentation classically relies on one of two criteria, either finding areas with coherent vocabulary use or detecting discontinuities. In this paper, we propose a segmentation criterion combining both lexical cohesion and disruption, enabling a trade-off between the two. We provide the mathematical formulation of the criterion and an efficient graph based decoding algorithm for topic segmentation. Experimental results on standard textual data sets and on a more challenging corpus of automatically transcribed broadcast news shows demonstrate the benefit of such a combination. Gains were observed in all conditions, with segments of either regular or varying length and abrupt or smooth topic shifts. Long segments benefit more than short segments. However the algorithm has proven robust on automatic transcripts with short segments and limited vocabulary reoccurrences.

1 Introduction

Topic segmentation consists in evidentiating the semantic structure of a document: Algorithms developed for this task aim at automatically detecting frontiers which define topically coherent segments in a text.

Various methods for topic segmentation of textual data are described in the literature, e.g., (Reynar, 1994; Hearst, 1997; Ferret et al., 1998; Choi, 2000; Moens and Busser, 2001; Utiyama and Isahara, 2001), most of them relying on the notion of lexical cohesion, i.e., identifying segments with a consistent use of vocabulary, either based on words or on semantic relations between words. Reoccurrences of words or related words and lexical chains are two popular methods to evidence lexical cohesion. This general principle of lexical cohesion is further exploited for topic segmentation with two radically different strategies. On the one hand, a measure of the *lexical cohesion* can be used to determine coherent segments (Reynar, 1994; Moens and Busser, 2001; Utiyama and Isahara, 2001). On the other hand, shifts in the use of vocabulary can be searched for to directly identify the segment frontiers by measuring the *lexical disruption* (Hearst, 1997).

Techniques based on the first strategy yield more accurate segmentation results, but face a problem of over-segmentation which can, up to now, only be solved by providing prior information regarding the distribution of segment length or the expected number of segments. In this paper, we propose a segmentation criterion combining both cohesion and disruption along with the corresponding algorithm for topic segmentation. Such a criterion ensures a coherent use of vocabulary within each resulting segment, as well as a significant difference of vocabulary between neighboring segments. Moreover, the combination of these two strategies enables regularizing the number of segments found without resorting to prior knowledge.

This piece of work uses the algorithm of Utiyama and Isahara (2001) as a starting point, a versatile and performing topic segmentation algorithm cast in a statistical framework. Among the benefits of this algorithm are its independency to any particular domain and its ability to cope with thematic segments

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1314–1324,
Seattle, Washington, USA, 18-21 October 2013.

of highly varying lengths, two interesting features to obtain a generic solution to the problem of topic segmentation. Moreover, the algorithm has proven to be up to the state of the art in several studies, with no need of a priori information about the number of segments (contrary to algorithms in (Malioutov and Barzilay, 2006; Eisenstein and Barzilay, 2008) that can attain a higher segmentation accuracy). It also provides an efficient graph based implementation of which we take advantage.

To account both for cohesion and disruption, we extend the formalism of Isahara and Utiyama using a Markovian assumption between segments in place of the independence assumption of the original algorithm. Keeping unchanged their probabilistic measure of lexical cohesion, the Markovian assumption enables to introduce the disruption between two consecutive segments. We propose an extended graph based decoding strategy, which is both optimal and efficient, exploiting the notion of generalized segment model or semi hidden Markov models. Tests are performed on standard textual data sets and on a more challenging corpus of automatically transcribed broadcast news shows.

The seminal idea of this paper was partially published in (Simon et al., 2013) in the French language. The current paper significantly elaborates on the latter, with a more detailed description of the algorithm and additional contrastive experiments including more data sets. In particular, new experiments clearly demonstrate the benefit of the method in a realistic setting with statistically significant gains.

The organization of the article is as follows. Existing work on topic segmentation is presented in Section 2, emphasizing the motivations of the model we propose. Section 3 details the baseline method of Utiyama and Isahara before introducing our algorithm. Experimental protocol and results are given in Section 4. Section 5 summarizes the finding and concludes with a discussion of future work.

2 Related work

Defining the concept of theme precisely is not trivial and a large number of definitions have been given by linguists. Brown and Yule (1983) discuss at length the difficulty of defining a topic and note: *"The notion of 'topic' is clearly an intuitively satisfactory way of describing the unifying principle which makes one stretch of discourse 'about' something and the next stretch 'about' something else, for it is appealed to very frequently in the discourse analysis literature. Yet the basis for the identification of 'topic' is rarely made explicit"*. To skirt the issue of defining a topic, they suggest to focus on topic-shift markers and to identify topic changes, what most current topic segmentation methods do.

Various characteristics can be exploited to identify thematic changes in text data. The most popular ones rely either on the lexical distribution information to measure lexical cohesion (i.e., word reoccurrences, lexical chains) or on linguistic markers such as discourse markers which indicate continuity or discontinuity (Grosz and Sidner, 1986; Litman and Passonneau, 1995). Linguistic markers are however often specific to a type of text and cannot be considered in a versatile approach as the one we are targeting, where versatility is achieved relying on the sole lexical cohesion.

The key point with lexical cohesion is that a significant change in the use of vocabulary is considered to be a sign of topic shift. This general idea translates into two families of methods, local ones targeting a local detection of lexical disruptions and global ones relying on a measure of the lexical cohesion to globally find segments exhibiting coherence in their lexical distribution.

Local methods (Hearst, 1997; Ferret et al., 1998; Hernandez and Grau, 2002; Claveau and Lefèvre, 2011) locally compare adjacent fixed size regions, claiming a boundary when the similarity between the adjacent regions is small enough, thus identifying points of high lexical disruption. In the seminal work of Hearst (1997), a fixed size window divided into two adjacent blocks is used, consecutively centered at each potential boundary. Similarity between the adjacent blocks is computed at each point, the resulting similarity profile being analyzed to find significant valleys which are considered as topic boundaries.

On the contrary, global methods (Reynar, 1994; Choi, 2000; Utiyama and Isahara, 2001; Ji and Zha, 2003; Malioutov and Barzilay, 2006; Misra et al., 2009) seek to maximize the value of the lexical cohesion on each segment resulting from the segmentation globally on the text. Several approaches have

been taken relying on self-similarity matrices, such as dot plots, or on graphs. A typical and state-of-the-art algorithm is that of Utiyama and Isahara (2001) whose principle is to search globally for the best path in a graph representing all possible segmentations and where edges are valued according to the lexical cohesion measured in a probabilistic way.

When the lengths of the respective topic segments in a text (or between two texts) are very different from one another, local methods are challenged. Finding out an appropriate window size and extracting boundaries become critical with segments of varying length, in particular when short segments are present. Short windows will render comparison of adjacent blocks difficult and unreliable while long windows cannot handle short segments. The lack of a global vision also makes it difficult to normalize properly the similarities between blocks and to deal with statistics on segment length. While global methods override these drawbacks, they face the problem of over-segmentation due to the fact that they mainly rely on the sole lexical cohesion. Short segments are therefore very likely to be coherent which calls for regularization introduced as priors on the segments length.

These considerations naturally lead to the idea of methods combining lexical cohesion and disruption to make the best of both worlds. While the two criteria rely on the same underlying principle of lexical coherence (Grosz et al., 1995) and might appear as redundant, the resulting algorithms are quite different in their philosophy. A first (and, to the best of our knowledge, unique) attempt at capturing a global view of the local dissimilarities is described in Malioutov and Barzilay (2006). However, this method assumes that the number of segments to find is known beforehand which makes it difficult for real-world usage.

3 Combining lexical cohesion and disruption

We extend the graph-based formalism of Utiyama and Isahara to jointly account for lexical cohesion and disruption in a global approach. Clearly, other formalisms than the graph-based one could have been considered. However, graph-based probabilistic topic segmentation has proven very accurate and versatile, relying on very minimal prior knowledge on the texts to segment. Good results at the state-of-the-art have also been reported in difficult conditions with this approach (Misra et al., 2009; Claveau and Lefèvre, 2011; Guinaudeau et al., 2012).

We briefly recall the principle of probabilistic graph-based segmentation before detailing a Markovian extension to account for disruption.

3.1 Probabilistic graph-based segmentation

The idea of the probabilistic graph-based segmentation algorithm is to find the segmentation into the most coherent segments constrained by a prior distribution on segments length. This problem is cast into finding the most probable segmentation of a sequence of t basic units (i.e., sentences or utterances composed of words) $W = u_1^t$ among all possible segmentations, i.e.,

$$\hat{S} = \arg\max_S P[W|S]P[S] \ . \qquad (1)$$

Assuming that segments are mutually independent and assuming that basic units within a segment are also independent, the probability of a text W for a segmentation $S = S_1^m$ is given by

$$P[W|S_1^m] = \prod_{i=1}^{m} \prod_{j=1}^{n_i} P[w_j^i|S_i] \ , \qquad (2)$$

where n_i is the number of words in the segment S_i, w_j^i is the j^{th} word in S_i and m the number of segments. The probability $P[w_j^i|S_i]$ is given by a Laplace law where the parameters are estimated on S_i, i.e.,

$$P[w_j^i|S_i] = \frac{f_i(w_j^i) + 1}{n_i + k} \ , \qquad (3)$$

where $f_i(w_j^i)$ is the number of occurrences of w_j^i in S_i and k is the total number of distinct words in W, i.e., the size of the vocabulary $\mathcal{V}$. This probability favors segments that are homogeneous, increasing when words are repeated and decreasing consistently when they are different. The prior distribution on segment length is given by a simple model, $P[S_1^m] = n^{-m}$, where n is the total number of words, exhibiting a large value for a small number of segments and conversely.

The optimization of Eq. 1 can be efficiently implemented as the search for the best path in a

weighted graph which represents all the possible segmentations. Each node in the graph corresponds to a possible frontier placed between two utterances (i.e., we have a node between each pair of utterances), the arc between nodes i and j representing a segment containing utterances u_{i+1} to u_j. The corresponding arc weight is the generalized probability of the words within segment $S_{i \to j}$ according to

$$v(i,j) = \sum_{k=i+1}^{j} ln(P[u_k|S_{i \to j}]) - \alpha ln(n)$$

where the probability is given as in Eq. 3. The factor α is introduced to control the trade-off between the segments length and the lexical cohesion.

3.2 Introduction of the lexical disruption

Eq. 2 derives from the assumption that each segment S_i is independent from the others, which makes it impossible to consider disruption between two consecutive segments. To do so, the weight of an arc corresponding to a segment S_i should take into account how different this segment is from S_{i-1}. This is typically handled using a Markovian assumption of order 1. Under this assumption, Eq. 2 is reformulated as

$$P[W|S_1^m] = P[W|S_1] \prod_{i=2}^{m} P[W|S_i, S_{i-1}] \ ,$$

where the notion of disruption can be embedded in the term $P[W|S_i, S_{i-1}]$ which explicitly mentions both segments. Formally, $P[W|S_i, S_{i-1}]$ is defined as a probability. However, arbitrary scores which do not correspond to probabilities can be used instead as the search for the best path in the graph of possible segmentations makes no use of probability theory. In this study, we define the score of a segment S_i given S_{i-1} as

$$\ln P[W|S_i, S_{i-1}] = \ln P[W_i|S_i] - \lambda \Delta(W_i, W_{i-1}) \quad (4)$$

where W_i designates the set of utterances in S_i and the rightmost part reflects the disruption between the content of S_i and of S_{i-1}. Eq. 4 clearly combines the measure of lexical cohesion with a measure of the disruption between consecutive segments: $\Delta(W_i, W_{i-1}) > 0$ measures the coherence between S_i and S_{i-1}, the substraction thus accounting for disruption by penalizing consecutive coherent segments. The underlying assumption is that the bigger $\Delta(W_i, W_{i-1})$, the weaker the disruption between the two segments. Parameter λ controls the respective contributions of cohesion and disruption.

We initially adopted a probabilistic measure of disruption based on cross probabilities, i.e., $P[W_i|S_{i-1}]$ and $P[W_{i-1}|S_i]$, which proved to have limited impact on the segmentation. We therefore prefer to rely on a cosine similarity measure between the word vectors representing two adjacent segments, building upon a classical strategy of local methods such as TextTiling (Hearst, 1997). The cosine similarity measure is calculated between vectors representing the content of resp. S_i and S_{i-1}, denoted $\mathbf{v}_i$ and $\mathbf{v}_{i-1}$, where $\mathbf{v}_i$ is a vector containing the (tf-idf) weight of each term of $\mathcal{V}$ in S_i. The cosine similarity is classically defined as

$$\cos(\mathbf{v}_{i-1}, \mathbf{v}_i) = \frac{\sum\limits_{v \in \mathcal{V}} \mathbf{v}_{i-1}(v) \ \mathbf{v}_i(v)}{\sqrt{\sum\limits_{v \in \mathcal{V}} \mathbf{v}_{i-1}^2(v) \ \sum\limits_{v \in \mathcal{V}} \mathbf{v}_i^2(v)}} \ . \quad (5)$$

$\Delta(W_i, W_{i-1})$ is calculated from the cosine similarity measure as

$$\Delta(W_i, W_{i-1}) = (1 - \cos(\mathbf{v}_{i-1}, \mathbf{v}_i))^{-1} \ , \quad (6)$$

thus yielding a small penalty in Eq. 4 for highly disrupting boundaries, i.e., corresponding to low similarity measure.

Given the quantities defined above, the algorithm boils down to finding the best scoring segmentation as given by

$$\hat{S} = \arg\max_S \sum_{i=1}^{m} ln(P[W_i|S_i]) - \lambda \sum_{i=2}^{m} \Delta(W_i, W_{i-1}) - \alpha m ln(n) \ . \quad (7)$$

3.3 Segmentation algorithm

Translating Eq. 7 into an efficient algorithm is not straightforward since all possible combinations of adjacent segments need be considered. To do so in a graph based approach, one needs to keep separated the paths of different lengths ending in a given node. In other words, only paths of the same length ending

at a given point, with different predecessors, should be recombined so that disruption can be considered properly in subsequent steps of the algorithm. Note that, in standard decoding as in Utiyama and Isahara's algorithm, only one of such paths, the best scoring one, would be retained. We employ a strategy inspired from the decoding strategy of segment models or semi-hidden Markov model with explicit duration model (Ostendorf et al., 1996; Delakis et al., 2008).

Search is performed through a lattice $L = \{V, E\}$, with V the set of nodes representing potential boundaries and E the set of edges representing segments, i.e., a set of consecutive utterances. The set V is defined as

$$V = \{n_{ij} | 0 \leq i, j \leq N\} \quad ,$$

where n_{ij} represents a boundary after utterance u_i reached by a segment of length j utterances and $N = t+1$. In the lattice example of Fig. 1, it is trivial to see that for a given node, all incoming edges cover the same segment. For example, the node n_{42} is positioned after $u4$ and all incoming segments contain the two utterances $u3$ and $u4$. Edges are defined as

$$E = \{e_{ip,jl} | 0 \leq i, p, j, l \leq N; \\ i < j; i = j - l; L_{\text{min}} \leq l \leq L_{\text{max}}\} \quad ,$$

where $e_{ip,jl}$ connects n_{ip} and n_{jl} with the constraint that $l = j - i$ and $L_{\text{min}} \leq l \leq L_{\text{max}}$. Thus, an edge $e_{ip,jl}$ represents a segment of length l containing utterances from u_{i+1} to u_j, denoted $S_{i \to j}$. In Fig. 1, $e_{01,33}$ represents a segment of length 3 from n_{01} to n_{33}, covering utterances u_1 to u_3. To avoid explosion of the lattice, a maximum segment length L_{max} is defined. Symmetrically, a minimum segment size can be used.

The property of this lattice, where, by construction, all edges out of a node have the same segment as a predecessor, makes it possible to weight each edge in the lattice according to Eq. 4. Consider a node n_{ij} for which all incoming edges encompass utterances u_{i-j} to u_i. For each edge out of n_{ij}, whatever the target node (i.e., the edge length), one can therefore easily determine the lexical cohesion as defined by the generalized probability of Eq. 3 and the disruption with respect to the previous segment as defined by Eq. 6.

Algorithm 1 Maximum probability segmentation

Step 0. Initialization
$q[0][j] = 0 \quad \forall j \in [L_{\text{min}}, L_{\text{max}}]$
$q[i][j] = -\infty \quad \forall i \in [1, N], j \in [L_{\text{min}}, L_{\text{max}}]$

Step 1. Assign best score to each node
for $i = 0 \to t$ **do**
  for $j = L_{\text{min}} \to L_{\text{max}}$ **do**
    for $k = L_{\text{min}} \to L_{\text{max}}$ **do**
      /* extend path ending after u_i with a segment of length j with an arc of length k */

$$q[i+k][k] = \max \begin{cases} q[i+k][k], \\ q[i][j] + \\ \text{Cohesion}(u_{i+1} \to u_{i+k}) - \\ \lambda \Delta(u_{i-j} \to u_i; u_{i+1} \to u_{i+k}) \end{cases}$$

    end for
  end for
end for

Step 2. Backtrack from n_{Nj} with best score $q[N][j]$

Given the weighted decoding graph, the solution to Eq. 7 is obtained by finding out the best path in the decoding lattice, which can be done straightforwardly by scanning nodes in topological order. The decoding algorithm is summarized in Algorithm 1 with an efficient implementation in $o(NL_{\text{max}}^2)$ which does not require explicit construction of the lattice.

4 Experiments

Experiments are performed on three distinct corpora which exhibit different characteristics, two containing textual data and one spoken data. We first present the corpora before presenting and discussing results on each.

4.1 Corpora

The artificial data set of Choi (2000) is widely used in the literature and enables comparison of a new segmentation method with existing ones. Choi's data set consist of 700 documents, each created by concatenating the first z sentences of 10 articles randomly chosen from the Brown corpus, assuming each article is on a different topic. Table 1 provides

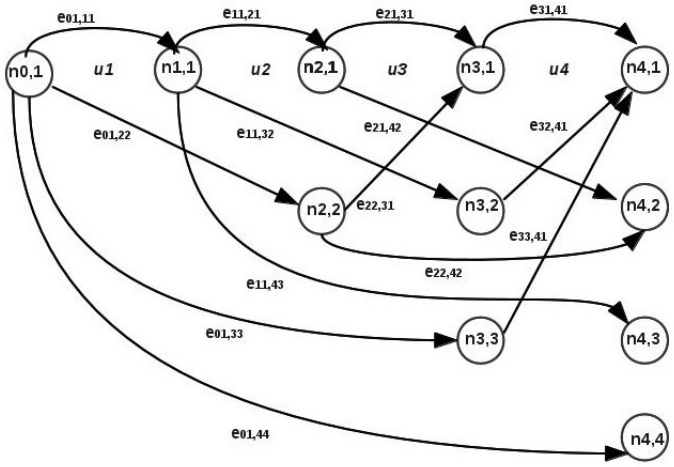


Figure 1: An example of a lattice L.

$z =$	3–11	3–5	6–8	9–11
# samples	400	100	100	100

Table 1: Number of documents in Choi's corpus (Choi, 2000).

the corpus statistics, where z=3–11 means z is randomly chosen in the range $[3, 11]$. Hence, Choi's corpus is adapted to test the ability of our model to deal with variable segments length, z=3–11 being the most difficult condition. Moreover, Choi's corpus provides a direct comparison with results reported in the literature.

One of the main criticism of Choi's data set is the presence of abrupt topic changes due to the artificial construction of the corpus. We therefore report results on a textual corpus with more natural topic changes, also used in (Eisenstein and Barzilay, 2008). The data set consists of 277 chapters selected from (Walker et al., 1990), a medical textbook, where each chapter—considered here as a document—was divided by its author into thematically coherent sections. The data set has a total of 1,136 segments with an average of 5 segments per document and an average of 28 sentences per segment. This data set is used to study the impact of smooth, natural, topic changes.

Finally, results are reported on a corpus of automatic transcripts of TV news spoken data. The data set consists of 56 news programs (≈1/2 hour each), broadcasted in February and March 2007 on the French television channel France 2, and transcribed by two different automatic speech recognition (ASR) systems, namely IRENE (Huet et al., 2010) and LIMSI (Gauvain et al., 2002), with respective word error rates (WER) around 36 % and 30 %. Each news program consists of successive reports of short duration (2-3 min), possibly with consecutive reports on different facets of the same news. The reference segmentation was established by associating a topic with each report, i.e., placing a boundary at the beginning of a report's introduction (and hence at the end of the closing remarks). The TV transcript data set, which corresponds to some real-world use cases in the multimedia field, is very challenging for several reasons. On the one hand, segments are short, with a reduced number of repetitions, synonyms being frequently employed. Moreover, smooth topic shifts can be found, in particular at the beginning of each program with different reports dedicated to the headline. On the other hand, transcripts significantly differ from written texts: no punctuation signs or capital letters; no sentence structure but rather utterances which are only loosely syntactically motivated; presence of transcription errors which may imply an accentuated lack of word repetitions.

All data were preprocessed in the same way: Words were tagged and lemmatized with TreeTag-

ger[1] and only the nouns, non modal verbs and adjectives were retained for segmentation. Inverse document frequencies used to measure similarity in Eq. 5 are obtained on a per document basis, referring to the number of sentences in textual data and of utterances in spoken data.

4.2 Results

Performance is measured by comparison of hypothesized frontiers with reference ones. Alignment assumes a tolerance of 1 sentence on texts and of 10 seconds on transcripts, which corresponds to standard values in the literature. Results are reported using recall, precision and F1-measure. Recall refers to the proportion of reference frontiers correctly detected; Precision corresponds to the ratio of hypothesized frontiers that belong to the reference segmentation; F1-measure combines recall and precision in a single value. These evaluation measures were selected because recall and precision are not sensitive to variations of segment length contrary to the *Pk* measure (Beeferman et al., 1997) and do not favor segmentations with a few number of frontiers as *WindowDiff* (Pevzner and Hearst, 2002) (see (Niekrasz and Moore, 2010) for a rigorous analytical explanation of the biases of *Pk* and *WindowDiff*).

Several configurations were considered in the experiments; due to space constraints, only the most salient experiments are presented here. In Eq. 7, the parameter α, which controls the contribution of the prior model with respect to the lexical cohesion and disruption, allows for different trade-offs between precision and recall. For any given value of λ, α is thus varied, providing the range of recall/precision values attainable. Results are compared to a baseline system corresponding to the application of the original algorithm of Utiyama and Isahara (i.e., setting $\lambda = 0$). This baseline has been shown to be a high-performance algorithm, in particular with respect to local methods that exploit lexical disruption. Differences in F1-measure between this baseline and our system presented below are all statistically significant at the level of $p < 0.01$ (paired t-test).

Choi's corpus. Figure 2 reports results obtained on Choi's data set, each graphic corresponding to a specific variation in the size of the thematic segments forming the documents (e.g., 9 to 11 sentences for the top left graphic). Results are provided for different values of λ in terms of F1-measure boxplots, i.e., variations of the F1-measure when α varies (same range of variation for α considered for each plot), where the leftmost boxplot, denoted by UI, corresponds to the baseline. Box and whisker plots graphically depicts the distribution of the F1-measures that can be attained by varying α, plotting the median value, the first and third quartile and the extrema.

z	τ	F1 gain	Confidence interval 95 % UI	Confidence interval 95 % Combined
3-5	0	-0.2	[66.6,74.26]	[75.23,78.08]
3-5	1	0.7	[72.25,83.4]	[87.88,92.13]
3-11	1	0.23	[68.5,79.3]	[86.6,87.43]
6-8	1	0.4	[68.48,80.99]	[76.9,85.17]
9-11	0	1.6	[64.35,75.16]	[81.31,84.86]
9-11	1	1.4	[68.39,80.39]	[84.37,88.9]

Table 2: Gain in F1-measure for Choi's corpus when using lexical cohesion and disruption, and the corresponding 95 % confidence intervals for the F1-measure. Results are reported for different tolerance τ. UI denotes the baseline and Combined the proposed model.

Figure 2 shows that, whatever the segments length, results globally improve according to the importance given to the disruption (λ variable). Moreover, the variation in F1-measure diminishes when disruption is considered, thus indicating the influence of the prior model diminishes. When the segments size decreases (see Figs. 2(b), 2(c), 2(d)), the difference in the maximum F1-measure between our results and that of the baseline lowers, however still in favor of our model. This can be explained by the fact that our approach is based on the distribution of words, thus more words better help discriminate between potential thematic frontiers. Finally, using too large values for λ can lead to under-segmentation, as can be seen in Fig. 2(d) where, for $\lambda = 3$, the variation of F1-measure increases and the distribution becomes negatively skewed (i.e., the median is closer to the third quartile than to the first).

These results are confirmed by Table 2 which presents the gain in F1-measure (i.e., the difference between the highest F1-measure obtained when

[1] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

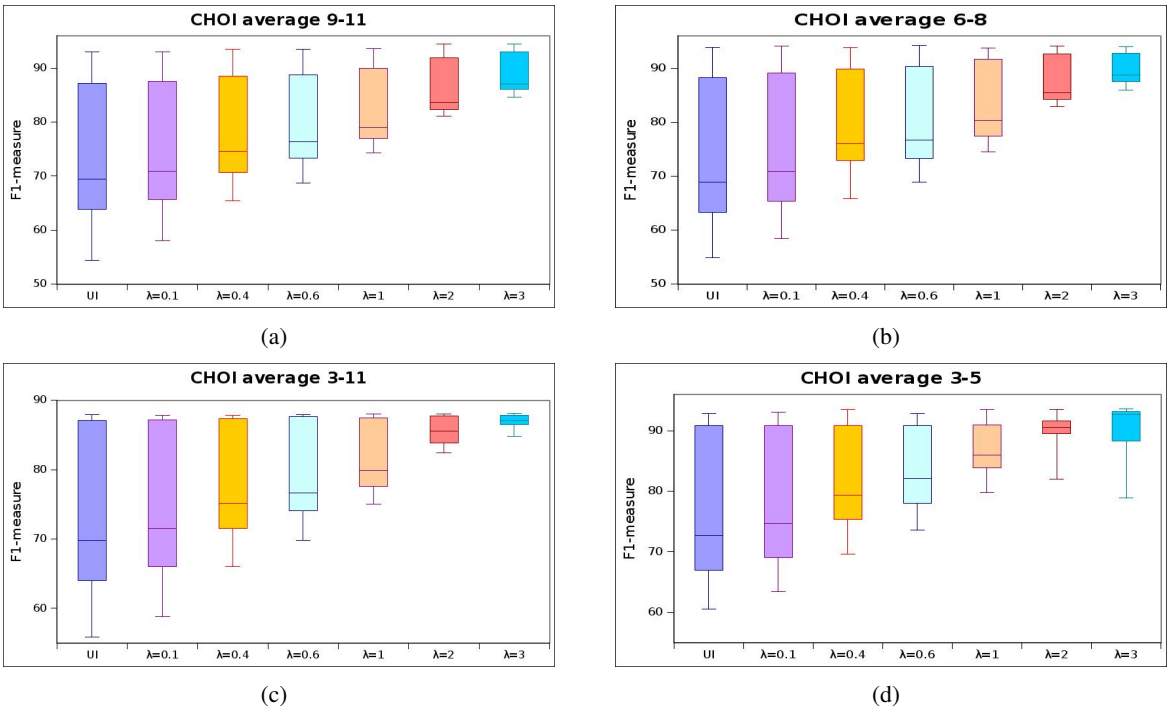


Figure 2: F1-measure variation obtained on Choi's corpus. In each graphic, the leftmost boxplot UI corresponds to results obtained by using the sole lexical cohesion (baseline), while the λ value is the importance given to the lexical disruption in our approach. Results are provided for the same range of variation of factor α, allowing a tolerance of 1 sentence between the hypothesized and reference frontiers.

combining lexical cohesion and disruption and the highest value for the baseline) for each of the four sets of documents in Choi's corpus, together with the 95 % confidence intervals: The effect of using the disruption is higher when segment size is longer, whether evaluation allows or not for a tolerance τ between the hypothesized frontiers and the reference ones. A qualitative analysis of the segmentations obtained confirmed that employing disruption helps eliminate wrong hypothesis and shift hypothesized frontiers closer to the reference ones (explaining the higher gain at tolerance 0 for 9-11 data set). When smaller segments—thus few word repetitions—and no tolerance are considered (e.g., 3–5), disruption cannot improve segmentation. Our model is globally stable with respect to segment length, with relatively similar gain for 3–11 and 6–8 data sets in which the average number of words (distinct or repeated) is close.

Results discussed up to now are optimistic as they correspond to the best F1 value attainable computed a posteriori. Stability of the results was confirmed

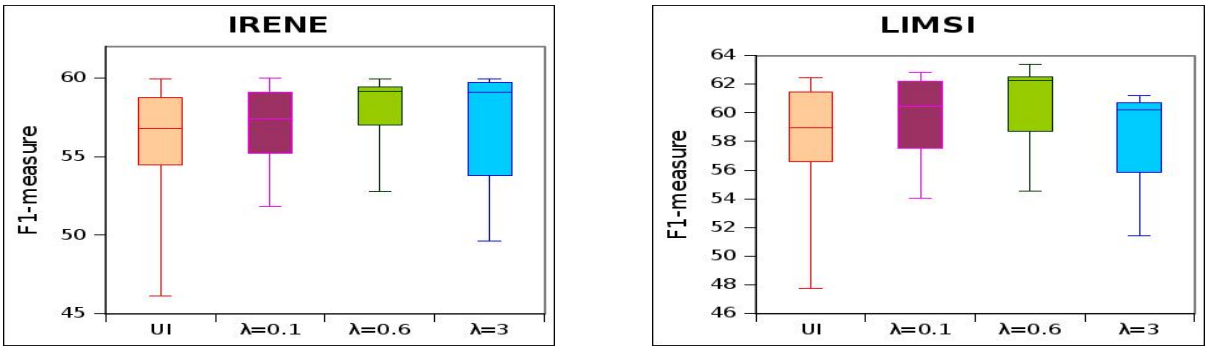
$z =$	3–5	3–11	6–8	9–11
UI	91.9	87.0	93.1	92.8
Combined	92.9	87.5	93.5	94.0

Table 3: F1 results using cross-validation on Choi's data set.

using cross-validation with 5 folds (10 folds for z=3–11): Parameters λ and α maximizing the F1-measure are determined on all but one fold, this last fold being used for evaluation. Results, averaged over all folds, are reported in Tab. 3 for the baseline and the method combining cohesion and disruption.

Medical textbook corpus. The medical textbook corpus was previously used for topic segmentation by Eisenstein and Barzilay (2008) with their algorithm BayesSeg[2]. We thus compare our results with those obtained by BayesSeg and by the baseline. When considering the best F1-measure (i.e., the best F1-measure which can be achieved by varying α and

[2]The code and the data set are available at http://groups.csail.mit.edu/rbg/code/bayesseg/



(a) (b)

Figure 3: Boxplots showing F1-measure variation on transcripts obtained using IRENE and LIMSI automatic speech recognition systems.

λ), we achieved an improvement of 2.2 with respect to BayesSeg when no tolerance is allowed, and of 0.5 when the tolerance is of 1 sentence. The corresponding figures with respect to the baseline are 0.6 and 0.4. When considering the F1-measure value for which the number of hypothesized frontiers is the closest to the number of reference boundaries, improvement is of resp. 1.5 and 0.5 with respect to BayesSeg, -0.1 and 0.4 with respect to the baseline. These results show that our model combining lexical cohesion and disruption is also able to deal with topic segmentation of corpora from a homogeneous domain, with smooth topic changes and segments of regular size.

One can argue that the higher number of free parameters in our method explains most of the gain with respect to BayesSeg. While BayesSeg has only one free parameter (as opposed to two in our case), the number of segments is assumed to be provided as prior knowledge. This assumption can be seen as an additional free parameter, i.e., the number of segments, and is a much stronger constraint than we are using. Moreover, cross-validation experiments on the Choi data set show that improvement is not due to over-fitting of the development data thanks to an additional parameter. Gains on development set with parameters tuned on the development set itself and with parameters tuned on a held-out set in cross-validation experiments are in the same range.

TV news transcripts corpus Figure 3 provides results, in terms of F1-measure variation, for TV news transcripts obtained with the two ASR systems. On this highly challenging corpus, with short segments, wrongly transcribed spoken words, and thus few word repetitions, the capabilities of our model to overcome the baseline system are reduced. Yet, an improvement of the quality of the segmentation of these noisy data is still observed, and general conclusions are quite similar—though a bit weaker—to those already made for Choi's corpus. Results are confirmed in Table 4 which presents the gain in F1-measure of our model together with the 95 % confidence interval, where F1-measure values correspond to that of segmentations with a number of hypothesized frontiers the closest to the reference. The two first lines show that the gain is smaller for IRENE transcripts which have a higher WER, thus fewer words available to discriminate between segments belonging to different topics. The impact of transcription errors is illustrated in the last three lines, when segmenting six TV news for which manual reference transcripts were available (line 3), where the higher the WER, the smaller the F1-measure gain.

5 Conclusions

We have proposed a method to combine lexical cohesion and disruption for topic segmentation. Experimental results on various data sets with various characteristics demonstrate the impact of taking into account disruption in addition to lexical cohesion. We observed gains both on data sets with segments of regular length and on data sets exhibiting segments of highly varying length within a document. Unsurprisingly, bigger gains were observed on doc-

Corpus	F1 gain	Confidence interval 95 % UI	Combined
IRENE	0.3	[54.4,57.6]	[56.92,59]
LIMSI	0.86	[56.7,60.2]	[59.44,61.95]
MANUAL (6)	0.77	[70.39,72.29]	[71.7,73.29]
IRENE (6)	0.2	[56.81,60.94]	[59.51,63.43]
LIMSI (6)	0.5	[64.27,68.64]	[67.7,71.56]

Table 4: Gain in F1-measure for TV news corpus automatic and manual transcripts when using lexical cohesion and disruption, and the corresponding 95 % confidence intervals. Last three rows report results on only 6 shows for which manual reference transcripts are available.

uments containing relatively long segments. However the segmentation algorithm has proven to be robust on automatic transcripts with short segments and limited vocabulary reoccurrences. Finally, we tested both abrupt topic changes and smooth ones with good results on both. Further work can be considered to improve segmentation of documents characterized by small segments and few words repetitions, such as using semantic relations or vectorization techniques to better exploit implicit relations not considered by lexical reoccurrence.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1997. Text segmentation using exponential models. In *2nd Conference on Empirical Methods in Natural Language Processing*, pages 35–46.

Gillian Brown and George Yule. 1983. Discourse analysis. *Cambridge University Press*.

Freddy Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *1st International Conference of the North American Chapter of the Association for Computational Linguistics*, pages 26–33.

Vincent Claveau and Sébastien Lefèvre. 2011. Topic segmentation of TV-streams by mathematical morphology and vectorization. In *12th International Conference of the International Speech Communication Association*, pages 1105–1108.

Manolis Delakis, Guillaume Gravier, and Patrick Gros. 2008. Audiovisual integration with segment models for tennis video parsing. *Computer Vision and Image Understanding*, 111(2):142–154.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Conference on Empirical Methods in Natural Language Processing*, pages 334–343.

Olivier Ferret, Brigitte Grau, and Nicolas Masson. 1998. Thematic segmentation of texts: Two methods for two kinds of texts. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 392–396.

Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. 2002. The LIMSI broadcast news transcription system. *Speech Communication*, 37(1–2):89–108.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, June.

Camille Guinaudeau, Guillaume Gravier, and Pascale Sébillot. 2012. Enhancing lexical cohesion measure with confidence measures, semantic relations and language model interpolation for multimedia spoken content topic segmentation. *Computer Speech and Language*, 26(2):90–104.

Marti A. Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Nicolas Hernandez and Brigitte Grau. 2002. Analyse thématique du discours : segmentation, structuration, description et représentation. In *5e colloque international sur le document électronique*, pages 277–285.

Stéphane Huet, Guillaume Gravier, and Pascale Sébillot. 2010. Morpho-syntactic post-processing of n-best lists for improved French automatic speech recognition. *Computer Speech and Language*, 24(4):663–684.

Xiang Ji and Hongyuan Zha. 2003. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 322–329.

Diane J. Litman and Rebecca J. Passonneau. 1995. Combining multiple knowledge sources for discourse segmentation. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 108–115.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Hemant Misra, François Yvon, Joemon M. Jose, and Olivier Cappe. 2009. Text segmentation via topic modeling: an analytical study. In *Proc. ACM conference on Information and knowledge management*, pages 1553–1556.

Marie-Francine Moens and Rik De Busser. 2001. Generic topic segmentation of document texts. In *24th*

International Conference on Research and Developement in Information Retrieval, pages 418–419.

John Niekrasz and Johanna D. Moore. 2010. Unbiased discourse segmentation evaluation. In *Spoken Language Technology*, pages 43–48.

Mari Ostendorf, Vassilios V. Digalakis, and Owen A. Kimball. 1996. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28:19–36.

Jeffrey C. Reynar. 1994. An automatic method of finding topic boundaries. In *32nd Annual Meeting on Association for Computational Linguistics*, pages 331–333.

Anca Simon, Guillaume Gravier, and Pascale Sébillot. 2013. Un modèle segmental probabiliste combinant cohésion lexicale et rupture lexicale pour la segmentation thématique. In *20e conférence Traitement Automatique des Langues Naturelles*, pages 202–214.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *39th Annual Meeting on the Association for Computational Linguistics*, pages 499–506.

Kenneth H. Walker, Dallas W. Hall, and Willis J. Hurst. 1990. *Clinical Methods: The History, Physical, and Laboratory Examinations*. Butterworths.

This Text Has the Scent of Starbucks: A Laplacian Structured Sparsity Model for Computational Branding Analytics

William Yang Wang
School of Computer Science
Carnegie Mellon University
ww@cmu.edu

Edward Lin and John Kominek
Voci Technologies, Inc.
Pittsburgh, PA 15217
{ed.lin,john.kominek}@vocitec.com

Abstract

We propose a Laplacian structured sparsity model to study computational branding analytics. To do this, we collected customer reviews from Starbucks, Dunkin' Donuts, and other coffee shops across 38 major cities in the Midwest and Northeastern regions of USA. We study the brand related language use through these reviews, with focuses on the brand satisfaction and gender factors. In particular, we perform three tasks: automatic brand identification from raw text, joint brand-satisfaction prediction, and joint brand-gender-satisfaction prediction. This work extends previous studies in text classification by incorporating the dependency and interaction among local features in the form of structured sparsity in a log-linear model. Our quantitative evaluation shows that our approach which combines the advantages of *graphical modeling* and *sparsity modeling* techniques significantly outperforms various standard and state-of-the-art text classification algorithms. In addition, qualitative analysis of our model reveals important features of the language uses associated with the specific brands.

1 Introduction

In marketing science, branding is a modern marketing strategy of creating a unique image for a product in the customers' mind. Establishing the brand in the broad social context is just as important as building a good product (Makens, 1965; Lederer and Hill, 2001; Kim et al., 2013). In fact, blind taste test experiments have frequently shown how branding directly leads to the success of products and companies. Most notably is a continued study sponsored by Pepsi, known as the Pepsi Challenge[1], where Pepsi demonstrates how even though people preferred the taste of Pepsi, Coca-Cola's branding has made it more popular. Even now, Microsoft uses similar blind taste tests[2] to compare search engines, Bing and Google, showing that although participants prefer Bing's results, Google's brand might have strengthened over the years. These studies all suggest that brand and its associations play important roles in the customers' perceptions and decisions.

To accommodate the market change, companies frequently adjust branding strategies by analyzing how their customers receive and respond to branding messages. So far, such analysis is often done by using surveys and focus groups (Moon and Quelch, 2006), which is expensive and not time-efficient. Recently, with the advance of machine learning techniques, researchers from the chemistry and vision communities started to pay attention to the problem of automatic brand identification from smell (Luo et al., 2004) and images (Pelisson et al., 2003). In contrast, even though textual data that contains hidden branding information is abundantly available in many forms over the Web, automatic discovery and computational analysis on such data are not well studied in the past.

Computational branding analytics (CBA) seeks to extract information, trends, and demographics about a brand on the basis of free-form text, e.g. from blogs, Twitter comments, reviews, or forum posts. As described in Section 3, in this study we use a sub-

[1] http://en.wikipedia.org/wiki/Pepsi_Challenge
[2] http://www.bingiton.com/

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1325–1336,
Seattle, Washington, USA, 18-21 October 2013.

set of online Yelp reviews that discuss coffee shops. The main reason is that this source has the advantage of providing ground truth of multi-labeled data: each review has meta-information defining a 5-star rating, the object of the review, and the reviewer's name (from which we infer gender). For the purpose of this paper we decompose CBA into three sub-problems.

- How well can the brand being discussed be identified by the raw text?
- How well can the joint value of brand and rating be predicted?
- How well can the joint value of brand, rating, and gender be predicted?

There are two reasons why one may want to construct text-based classifiers of brand, rating, and gender, when such information is present in the review header. The first is that trained classifiers can then be applied to other data sources, such as blogs, where what is available is only the review itself. The second is that by "opening the hood" to the classifier one can examine which words exhibit high affiliation with the predicted variables. This can be done, for example, to contrast the preferences of males and females with respect to evaluating the qualities of a coffee shop. Examples of such insights are provided in Section 5.5.

In this paper, we propose a Laplacian structured sparsity model for computational branding analytics. Our main contributions are two-fold: first, in the novel task of automatic brand identification from text, we show that by incorporating the dependency structure and graphical interactions among local features, our model significantly outperforms various text classification algorithms such as the standard logistic regression, principle component analysis (PCA), linear kernel support vector machine (SVM), sparse, non-sparse, and mixed-penalty log-linear models. These improvements could also be seen from a joint brand-satisfaction prediction task and a gender-specific joint brand-satisfaction prediction task. In addition, our Laplacian augmented L_1-ball projection experiment shows that the advantage of Laplacian structured sparsity is robust across different parameter settings in a L_1-constrained problem. Secondly, the qualitative analysis of our machine learning model shows the interesting features and language use that relate to brand and its associated pragmatics.

In the next section, we outline related work in CBA, sparsity, and spectral graph learning. In Section 3, we describe the corpus in this study. The Laplacian structured sparsity model is introduced in Section 4. The experimental setup and results are presented in Section 5. A short discussion is followed in Section 6 and we conclude in Section 7.

2 Related Work

Early work on statistical brand analysis in the marketing community dates back to the work of Kuehn (1962), where he first hypothesizes that brand choice could be described as a learning process. Guadagni and Little (1983) further empirically tested the hypothesis by building a calibrated multinomial logistic regression model to predict the purchase of ground coffee, using the data from the optical scanning of product code in supermarkets. Outside the marketing community, statistical brand analysis is rarely seen. More recently, a study (Luo et al., 2004) applies neural networks to identify cigarette brands, with the hope of detecting illegal cigarettes from smell features. In image processing, researchers have studied the problem of brand identification from image using histogram comparison (Pelisson et al., 2003). However, to the best of our knowledge, even though textual data is vastly available, the problems of automatic brand identification from raw text and computational branding analytics, are new.

Although the domain of our data is on branding, our work also aligns with previous work in text and language classification. Over the years, logistic regression and linear kernel SVM have shown to be very successful in various regression and classification tasks in NLP (Chahuneau et al., 2012; Biadsy et al., 2011). Recently, sparse discriminative methods that model the sparse nature of text become attractive, because unlike dense models, they are less likely to overfit to the training data, easier to interpret, and often lead to state-of-the-art results. For example, Eisenstein et al. (2011b) use the $L_{1,\infty}$ sparsity model to discover sociolinguistic patterns. Wang et al. (2012a) compare lasso, ridge, and elastic net models to predict impoliteness behaviors in teenager conversations. Martins et al. (2011) investigate the tree-structured overlapping group lasso for

structured prediction problems. Chen et al. (2013) study the use of element-wise, group-wise, and hierarchical sparsity models for dialogue act classification. Sparse inducing priors are also investigated and shown to be effective in generative models for topic modeling (Eisenstein et al., 2011a; Wang et al., 2012b; Paul and Dredze, 2012).

Besides lacking sparsity, since the traditional discriminative methods in NLP often use interdependent features such as n-grams tokens, and part-of-speech tags, they also suffer from the problem of not explicitly modeling the complex dependency structure and interaction of local features from a global perspective. To solve this problem, graph methods seem to be a good solution, because they are simple, generalizable, and are often used to model such complex dependency structures (Cohen, 2012). However, combining the sparse modeling and spectral graphical modeling approaches in a principled way is challenging. Belkin et al. (2006) and Weinberger et al. (2007) are among the first to investigate graph Laplacians as a manifold regularization method for statistical learning. Recently, Gao et al. (2012) propose a histogram intersection based kNN method to construct a Laplacian matrix for a least-square sparse coding problem in image processing. Unfortunately, this method might be too specific to the SIFT-based image coding tasks, thus might not be applicable to the text classification problem that utilizes n-gram lexical features.

3 Datasets

We collected Yelp reviews from 1,860 Starbucks, Dunkin' Donuts[3], and other coffee shops all over the Midwest and Northeast regions in the period of 2009. A detail statistics of our data can be found in Table 1. The Midwest region includes 12 states[4] and 19 major cities, and the Northeast region includes 9 states[5] and 19 major cities. For each region, we divide the coffee shops into 60% training, 20% development, and 20% test, and there are no overlaps of coffee shops among these subsets. There are three values for the brand label: Starbucks, Dunkin' Donuts, and all other coffee shop brands. The majority class is "all other coffee shop brands", and the majority baseline is shown in Table 2. In the task of joint brand-satisfaction prediction, we utilize the review scores to approximate user satisfaction: scores 1-2 as the unsatisfactory label, 3 as moderate, and 4-5 as satisfactory. Since the Yelp reviews do not reveal the reviewer's gender, we use a similar method that U.S. Census Bureau used (OConnell and Gooding, 2006): we first automatically match the first name of the reviewer with the prior name-gender distributions in the census records, then manually examine the no-match cases and a subsample of the matched cases. For those who we cannot determine the gender, the review will be dropped from the gender-specific brand-satisfaction prediction task. After filtering, there are 8,528 documents for training, 2,928 for development, and, 3,046 for testing. Since the focus of this paper is not on feature engineering, we use unigram features to represent each review. Below is an example of positive review from a male Starbucks customer from Midwest.

	Coffee Shops			Reviews		
	Train	Dev.	Test	Train	Dev.	Test
1	451	150	150	3,513	1,087	1,424
2	665	222	222	6,982	2,530	2,358
T.	1,116	372	372	10,495	3,617	3,782

Table 1: Dataset statistics. *1: midwest region. 2: northeast region. T.: total.*

> *My favorite place for my iced vanilla lattes. They have screwed up my order before: instead of a grande, I got a venti. Not a fan of their pastries though. I got a donut once, and ended up feeding it to a pigeon in city garden. Friendly and fast service. Not open Sundays.*

The coffee shop dataset is freely available[6] for research purposes.

4 Our Approach

4.1 Problem Formulation and Predictive Tasks

The automatic brand identification problem could be considered as a traditional multiclass classifica-

[3]We chose these two brands because they are reported as the leading coffee shops by WSJ (Ovide, 2011) and Forbes (DiCarlo, 2004).

[4]IL, WI, SD, ND, MN, MO, OH, NE, KS, IA, IN, and MI.

[5]CT, ME, MA, NH, RI, VT, NJ, NY and PA.

[6]http://www.cs.cmu.edu/~yww/data/emnlp2013.zip

tion problem where the estimated label $\hat{Y}$ could be drawn from $Mult(\Gamma)$, where Γ is the parameter for the multinomial distribution. To solve this, a simple but accurate solution is to decompose the multiclass problem into multiple binary classification problems (Rifkin and Klautau, 2004) by training k one-vs-all binary classifiers, and then use the argmax criteria to select the best hypothesis from the k posteriors. As for a binary classifier, we need to infer the posterior from a Bernoulli distribution that is parametrized by $\hat{\theta}_y$. Similarly, we can derive k binary classifiers:

$$\hat{\theta}_y^{(1)}, \hat{\theta}_y^{(2)}, ..., \hat{\theta}_y^{(k)}. \quad (1)$$

So, instead of drawing $\hat{Y}$ from a multinomial distribution $Mult(\Gamma)$, we can draw the final label $\hat{Y}$ that has the largest posterior across all k classifiers:

$$\hat{Y} = \underset{Y,i=1,2,...,k}{\operatorname{argmax}} \Pr(Y|\hat{\theta}_y^{(i)}, \vec{X}_t) \quad (2)$$

where $\vec{X}_t$ is the testing vector, and $\Pr(Y|\hat{\theta}_y^{(i)}, \vec{X}_t)$ is the posterior probability given the learned classifiers and the testing vector.

In this paper, we investigate three multiclass classification tasks: first, we perform a 3-way classification task for automatic brand identification. In the task of brand-satisfaction prediction, we model the brand and the satisfaction label at the same time (Chahuneau et al., 2012): we perform the task of jointly predicting aggregate brand-satisfaction score for a review using 9-way classification. Similarly, we perform 18-way classification for the gender-specific joint brand-satisfaction prediction task.

4.2 The Log-Linear Framework and Its Regularized Variants

If we consider the standard logistic regression model as the binary classifier in this log-linear framework[7], then each classifier can be written as:

$$\hat{\theta}_y = \frac{\exp\left(\vec{W}^\top \vec{X}_j\right)}{1 + \exp\left(\vec{W}^\top \vec{X}_j\right)} \quad (3)$$

here, $\vec{X}_j$ is the j-th observed feature vector, label $y \in \{0, 1\}$, and $\vec{W}$ is a vector of the coefficients. To estimate the model parameters in equation (3), we only need to set the weights $\vec{W}$. We can obtain the following log likelihood, and its gradient function by taking the first-order partial derivative of $\vec{W}$:

$$\ell = \sum_j y_j \log \hat{\theta}_{y_j} + (1 - y_j) \log(1 - \hat{\theta}_{y_j}) \quad (4)$$

$$\frac{\partial \ell}{\partial \vec{W}} = \sum_j \left(\frac{\partial \hat{\theta}_{y_j}}{\partial \vec{W}}\right)\left(\frac{y_j}{\hat{\theta}_{y_j}} - \frac{1 - y_j}{1 - \hat{\theta}_{y_j}}\right) \quad (5)$$

$$\frac{\partial \hat{\theta}_{y_j}}{\partial \vec{W}} = \left(\hat{\theta}_{y_j} - (\hat{\theta}_{y_j})^2\right)\vec{X}, \quad (6)$$

since the log likelihood objective function (4) is concave, using standard gradient ascent with maximum likelihood estimation can solve the problem. However, this model does not penalize the noisy features and unreliable features that might overfit to the training data. To address this issue, we introduce the L_1 norm from lasso technique (Tibshirani, 1996) to regularize the above likelihood function. Thus, instead of maximizing the likelihood, we can minimize the loss function of the negative log-likelihood with a linear penalty:

$$\min\Big(-\ell + \lambda_1 ||\vec{W}||\Big) \quad (7)$$

where λ_1 is the regularization coefficient. The benefit of L_1 penalty in a discriminative model is similar to the double exponential distribution of the sparse priors in generative models (Eisenstein et al., 2011a): they both push the weights of many noisy features into zeros, revealing only the important features. However, since the L_1 penalty can introduce discontinuities to the original convex function, we can also consider an alternative non-sparse ridge estimator (Le Cessie and Van Houwelingen, 1992) with log loss and L_2 norm, and has the convex property:

$$\min\Big(-\ell + \lambda_2 ||\vec{W}||^2\Big) \quad (8)$$

Another option that balances the sparsity and smoothness would be the elastic net model (Zou and Hastie, 2005) that uses the composite penalty:

$$\min\Big(-\ell + \lambda_1 ||\vec{W}|| + \lambda_2 ||\vec{W}||^2\Big) \quad (9)$$

4.3 The Laplacian Structured Sparsity Model

So far, none of the above element-wise penalty models in the previous subsection takes into account the

[7] We thank Jacob Eisenstein for the initial derivation of the logistic regression model.

dependency structure of the local features. Inspired by Gao et al.(2012), we group the local features that have similar distributions together. The intuition is that, for features that have very similar empirical distributions in the training set, their weights should not be drastically different after the learning process in the same task. In our new objective function, it is desirable to introduce a new component that structurally penalize these cases where features that are very similar to each other, but have learned completely different weights, probably due to the noise or the data sparsity issue in the training data.

The Objective Function: To do this, we first define an inter-feature affinity matrix A, where $A_{(p,q)}$ measures the similarity between a pair of features p and q. In the spectral graph theory, this affinity matrix can be viewed as a weighted undirected graph $G = (V, E)$, where each node V_p denotes a feature p, and each edge $E_{(p,q)}$ indicates the closeness among the features p and q. We also introduce a weighted diagonal degree matrix D, of which each element in the diagonal $D_{(p,p)}$ is the sum of all weighted connections of node V_p: $D_{(p,p)} = \sum_{q=1}^{Q} A_{(p,q)}$. We propose the following objective function:

$$\min \Big(-\ell + \lambda_1 ||\vec{W}|| + \lambda_2 ||\vec{W}||^2 \tag{10}$$

$$+ \alpha \sum_{(p,q)} ||\vec{W}_p - \vec{W}_q||^2 A_{(p,q)} \Big) \tag{11}$$

We then denote a graph Laplacian matrix $L = D - A$ (Belkin and Niyogi, 2001), and rewrite the objective function as:

$$\min \Big(-\ell + \lambda_1 ||\vec{W}|| + \lambda_2 ||\vec{W}||^2 \tag{12}$$

$$+ \alpha (\vec{W}^\top L \vec{W}) \Big) \tag{13}$$

where α is the regularization parameter for the Laplacian structured sparsity term. Intuitively, the objective function can be interpreted as the sum of a negative log loss function, the sparsity-inducing penalty, the quadratic penalty, and the Laplacian structured penalty. Or, another view of this new model could be seen as a Laplacian augmented elastic net model where structured sparsity and feature interaction are considered.

The Laplacian Matrix: In this model, a key aspect is to derive the Laplacian matrix L. We propose the following three steps to learn the Laplacian matrix:

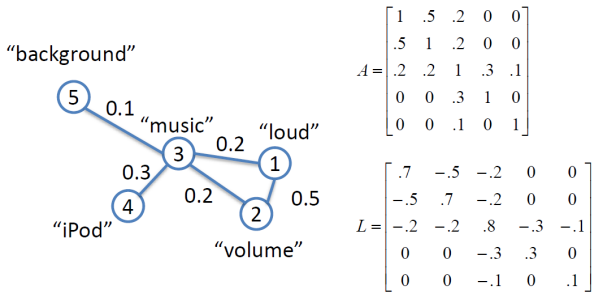


Figure 1: An example of the graph G, the corresponding affinity matrix A, and the corresponding Laplacian matrix L.

1. **Construct the distance matrix** $Dist$**.** To construct the distance matrix between each feature, we first transpose the instance-feature matrix, $I = \sum_j \vec{X}_j$, and assume that each feature (e.g. unigram in our task) is a random variable that has a multinomial distribution over the instances in the training set. Then, we compare each pair of features, and calculate the inter-feature distance matrix $Dist$ with Euclidean distance as a measure, and use the k-nearest neighbors (kNN) method (Beyer et al., 1999) to select the top neighbors of each feature.

2. **Derive the affinity matrix** A**.** To assign the weight on the edge $E_{(p,q)}$ for each connected nodes (the kNN of V in $Dist$), we use the cosine similarity $cosine(V_p, V_q)$ metric (Wang and Hirschberg, 2011).

3. **Generate the degree matrix** D **and Laplacian matrix** L**.** As discussed earlier, we sum up the symmetric affinity matrix by row, and obtain a diagonal degree matrix D, and we further define a Laplacian matrix $L = D - A$.

To calculate the above matrices in an efficient manner, we partition the covariate into blocks, and process each block in parallel (Chen et al., 2011). An intuitive example of the graph G, its associated affinity matrix A, and Laplacian matrix L, is shown in Figure 1.

Parameter Estimation: Regarding the optimization of objective function in (12-13), a notable problem is that the sparsity inducing L_1 term is non-differentiable, whereas this is not the case for the L_2 norm and the Laplacian structured sparsity term. If we first take the derivative of the latter two terms,

and we can derive the following gradient components:

$$\frac{\partial(\lambda_2||\vec{W}||^2 + \alpha(\vec{W}^\top L\vec{W}))}{\partial\vec{W}} \tag{14}$$

$$= 2\lambda_2\vec{W} + \alpha(\vec{W}^\top L^\top + \vec{W}^\top L) \tag{15}$$

$$= 2\lambda_2\vec{W} + \alpha(L^\top + L)\vec{W} \tag{16}$$

since our Laplacian matrix is symmetric, we can rewrite (16) as

$$2(\lambda_2\vec{W} + \alpha L\vec{W}) \tag{17}$$

Then, we combine the gradient of the log loss function in (5) with (17), and apply a bound-constrained re-formulation (Schmidt et al., 2007) and the limited memory BFGS (L-BFGS) method (Liu and Nocedal, 1989) to solve the L_1 regularized problem. The L-BFGS method has relatively low space complexity, and does not require the calculation of full Hessian matrix, thus it is often used for L_1 optimization problems.

Augmented Laplacian for an L_1-Constrained Problem: Instead of formulating the L_1-regularized problem by adding the L_1 norm, an alternative solution is to formulate a L_1-constrained problem by fixing the sum of all weights τ in the weight vector $\vec{W}$. The reason is because adding the L_1 norm will make the objective function not continuously differentiable, where as the L_1 constraint could be just a simple linear constraint (Lee et al., 2006). Thus, the alternative L_1-constrained problem could be defined as:

$$\min(-\ell), s.t. \sum_p \vec{W}_p \leq \tau \tag{18}$$

To test the robustness of Laplacian structured sparsity term in the setup of a L_1-constrained problem, we can incorporate the Laplacian penalty term into the above formula, and derive:

$$\min\Big(-\ell + \alpha(\vec{W}^\top L\vec{W})\Big), s.t. \sum_p \vec{W}_p \leq \tau \tag{19}$$

Note that the Laplacian matrix is positive-semidefinite,

$$\vec{W}^\top L\vec{W} = \vec{W}^\top \sum_{(p,q)} L_{(p,q)}\vec{W} \tag{20}$$

$$= \sum_{(p,q)} \vec{W}^\top L_{(p,q)}\vec{W} \tag{21}$$

$$= \sum_{(p,q)} ||\vec{W}_p - \vec{W}_q||^2 A_{(p,q)} \tag{22}$$

because this graph Laplacian penalty can be viewed as a quadratic term, and the objective function in equation 19 is now convex differentiable and will produce sparse estimates, so that we are able to use a limited-memory projected quasi-Newton method (Schmidt et al., 2009) to solve the dual form of this problem. The Lagrangian dual form of the problem in equation 19 can be written as:

$$\mathcal{L}(\vec{W}, \xi) = -\ell + \alpha(\vec{W}^\top L\vec{W}) \tag{23}$$

$$+ \beta\left(\sum_p \vec{W}_p - \tau\right) - \xi\vec{W} \tag{24}$$

where $\beta \in \mathbb{R}$ is a Lagrange multiplier, and $\xi \in \mathbb{R}^p_+$ is a p-dimensional vector of non-negative Lagrange multipliers. And then we can take first-order partial derivative with regard to $\vec{W}$, and set it to zero to derive the optimality:

$$\frac{\partial\mathcal{L}}{\partial\vec{W}} = -\sum_j \left(\hat{\theta}_{y_j} - (\hat{\theta}_{y_j})^2\right)\vec{X}\left(\frac{y_j}{\hat{\theta}_{y_j}} - \frac{1-y_j}{1-\hat{\theta}_{y_j}}\right) \tag{25}$$

$$+ 2\alpha L\vec{W} + \beta - \xi = 0 \tag{26}$$

To speed up the training, we use the linear-time L_1-ball projection method from Duchi et al. (2008) in our implementation.

5 Experiments

We first compare our model to various baselines in the 3-way automatic brand identification task. Besides the logistic regression, lasso, ridge and elastic net model that we introduced in Section 4.2, we also compare with a PCA-based logistic regression model where the dimensions of the feature space is reduced in half before the classification. A state-of-the-art linear kernel SVM model (Chang and Lin, 2011) is also taken into the comparison. In the second part, we perform 9-way joint classification of the brand-satisfaction labels. Similarly, we also perform a 18-way joint classification of the brand-gender-satisfaction labels. To test the robustness of our model, we vary the levels of sparsity of our Laplacian augmented method in a L_1-constrained problem. Finally, we analyze the identified features for CBA. Throughout this section, we use classification accuracy to report the results. We tune the regularization parameters of log-linear models and

Method	Dev.	Test
Majority class	75.67	78.08
Logistic regression	91.98	91.06
Linear SVM	92.45	91.75
PCA	91.67	91.20
Lasso	92.81	91.96
Ridge	92.56	91.67
Elastic net	92.81	91.83
Laplacian structured sparsity	**93.17***	**92.44***

Table 2: The automatic brand identification (3-way) performances. The best result is highlighted in **bold**. * indicates $p < .001$ comparing to the second best result.

Method	Dev.	Test
Majority class	55.43	55.18
Logistic regression	65.80	65.80
Linear SVM	67.67	65.44
PCA	63.92	62.53
Lasso	68.37	66.84
Ridge	67.79	65.55
Elastic net	68.79	66.82
Laplacian structured sparsity	**69.56***	**67.32***

Table 3: The joint brand-satisfaction prediction (9-way) performances. The best result is highlighted in **bold**. * indicates $p < .001$ comparing to the second best result.

the cost parameter of the SVM on the development set, and report results on both the development set and the held-out test set. The parameter for kNN was set to 5 according to previous literature (Gao et al., 2012). A paired two-tailed t-test is used to test the statistical differences among various models.

5.1 Automatic Brand Identification from Text

Given any piece of raw text from the Web (e.g. blogs, tweets, news, or forum posts), the first task for CBA is to identify which brand this text is related to. Our customer review data set is useful for this task, because the ground truth of the brand label is attached to each review. Table 2 shows the result of our model in this automatic brand identification task. In this 3-way classification task, the overall results indicate that it is relatively easy to identify the related brand from customer reviews. When evaluating our Laplacian structured sparsity model, our proposed model obtains the best performances of 93.17% and 92.44%, which are statistically better than the second best results ($p < .001$) in both datasets.

5.2 Joint Brand-Satisfaction Prediction

In our training data set, we observe a subtle correlation between the brand and satisfaction labels ($r = 0.09$, $p < .001$), which suggests us that it might be interesting to perform a joint prediction task for the brand-satisfaction labels. This task is also attractive from the business perspective, because it would be very useful for the companies to directly identify user's level of satisfaction about their brands. Table 3 shows that we achieve 69.56% accuracy on the development set, and 67.32% accuracy on the test set using our proposed Laplacian structured model ($p < .001$ comparing to the second best results).

Method	Dev.	Test
Majority class	28.24	27.68
Logistic regression	36.03	35.16
Linear SVM	41.05	39.49
PCA	35.35	34.44
Lasso	40.74	39.53
Ridge	40.98	38.94
Elastic net	41.15	38.96
Laplacian structured sparsity	**41.22***	**40.22***

Table 4: The joint brand-gender-satisfaction prediction (18-way) performances. The best result is highlighted in **bold**. * indicates $p < .001$ comparing to the second best result.

5.3 Joint Brand-Gender-Satisfaction Prediction

Another big interest in the marketing community is to predict subgroup preferences of specific brands. In this direction, we perform a 18-way joint brand-gender-satisfaction prediction using the gender labels that we described in Section 3. Table 4 shows that our proposed Laplacian structured sparsity model obtains a test accuracy of 40.22%, significantly better than the second best result ($p < .001$).

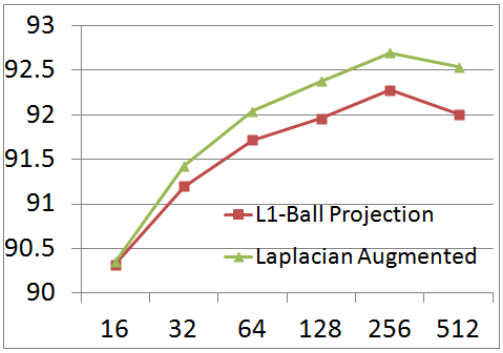


Figure 2: Automatic brand identification test performance varying the level of sparsity τ in a L_1 constrained problem.

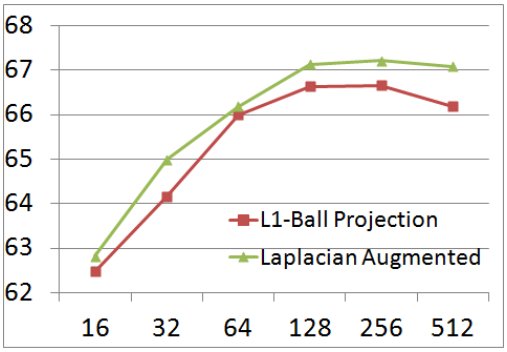


Figure 3: Joint brand-satisfaction prediction test performance varying the level of sparsity τ in a L_1 constrained problem.

5.4 Varying the Level of Sparsity in a L_1-Constrained Problem

To test the robustness of the Laplacian structured sparsity component, we exponentially increase the sum of weights τ to vary the level of sparsity in a L_1-constrained setup. When τ increases, the non-zero weights in the model also increases. Figures 2 and 3 show that the Laplacian augmented L_1-ball projection statistically outperform the L_1-ball projection baseline in all levels of sparsity ($p < .001$). In Figure 4, Laplacian augmented L_1-ball projection is also statistically better than the L_1-ball projection ($p < .001$), except when $\tau = 32$ and $\tau = 64$.

5.5 Exploratory Data Analysis

We outline the top 15 keywords from the Laplacian structured sparsity model that are associated with the Starbucks and Dunkin' Donuts brands in the automatic brand identification task in the Table 5. First of all, it is observed that our model has discovered synonyms for both brands: "sbux", "dd", "dds".

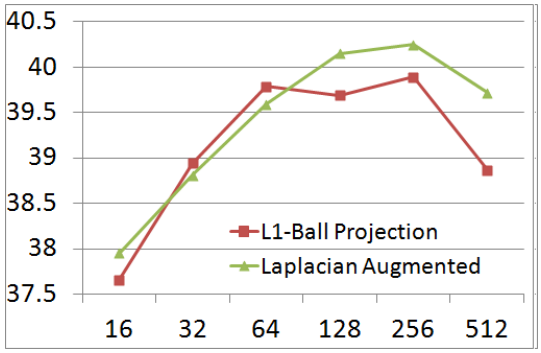


Figure 4: Joint brand-gender-satisfaction prediction test performance varying the level of sparsity τ in a L_1 constrained problem.

Also, the results imply that Starbucks' unique cup size branding strategy, "venti", "grande", "tall", has resonated with their customers as the words prominently show up as top features in reviews. Aligned with previous study in marketing science (Moon and Quelch, 2006), an informative set of features related to Starbucks store decorations showed up in our model: "store", "restroom", "public", "bathroom", and "spacious". In contrast, these features stopped to show up on the list of Dunkin' Donuts. Instead, TV and game (sports), which are indeed important features of dining at Dunkin' Donut, appeared. Note that Baskin-Robbins, which is a sub-brand of Dunkin' Brands Group, Inc., also appeared as informative features to predict Dunkin' Donuts.

To understand the preferences of different gender subgroups towards the two brands, we contrast in Table 6 and Table 7 the top features that identify the satisfied **female** and **male** customers in the joint brand-gender-satisfaction prediction task.

Interestingly, it seems that the female customers identify Starbucks as a place for "studying", with "fireplace" as the top preference of the spots in the store, and "winter" is also a high-ranked feature. Also, the adjective "super" was frequently mentioned by the female Starbucks customers (but not the males). As for Dunkin' Donuts, the top-ranked keywords are still mainly associated with its names, but it seems the snack "Munchkins" is highly preferred by the female customers. Not surprisingly, the cue words that the male customers identify the Starbucks brand do not always agree with those of the females. For example, instead of "fireplace", they prefer staying at the "patio", and drink the coffee from the "clover" brewing system. Interestingly,

Starbucks	weight	Dunkin'	weight
starbucks	1.9365	dd	2.4224
sbux	1.0152	dunkin	1.7781
venti	0.8216	donuts	1.6989
corporate	0.7032	dunks	1.6455
store	0.6580	dds	1.4936
particular	0.6512	donut	1.3979
tall	0.5496	dunkins	1.3729
restroom	0.5447	glazed	0.9975
tourists	0.5431	robbins	0.9402
public	0.5260	baskin	0.8578
lines	0.4956	sugar	0.6475
drink	0.4787	d	0.6327
bathroom	0.4721	ice	0.5835
spacious	0.4629	stale	0.5404
location	0.4611	game	0.5049
grande	0.4563	tv	0.5010

Table 5: Top features that identify the Starbucks and Dunkin' Donuts brands from the best model.

.

on the Dunkin's side, "munchkins" also disappeared and replaced by "glazed" (donuts). However, both males and females agreed that "fast" or "quick" service was an important feature of creating satisfaction, which echoes with the result from self-reported customer surveys (Moon and Quelch, 2006).

The word "**name**" is a prominent indicator for the female customers of Starbucks: at first we were puzzled, but after we digged into the database, we found reviews such as:

- *"... and the baristas are one of the nicest they always ask for your **name**, so you never end up with coffee meant for the guy behind you."*
- *"... she asked me my name and i told her and she excidetly proclaimed melissa and wrote my **name** on the cup. This place was probably one of the better starbucks ive been to."*
- *"... all of their employees are really friendly, and embarrassingly enough most know me by **name** and know my typical drink order grande nonfat misto with a flavor shot of white mocha. This is actually very helpful."*

The above examples show how our system effectively serves as a salient keyword spotter. And that as a keyword spotter one can use it to extract surrounding context and feed that through to the next stage of analysis, including examination by humans. This is extremely practical and useful, because it provides actionable items. For example, analysts can advise managers to revise their training manual and tell store employees to *remember the names of your frequent female customers*.

Starbucks	weight	Dunkin'	weight
starbucks	0.5013	dd	0.6931
chain	0.4268	dds	0.5620
winter	0.3382	dunkin	0.5344
fireplace	0.3089	donuts	0.4270
studying	0.2972	donut	0.3732
particular	0.2967	dunks	0.3687
super	0.2786	morning	0.3077
name	0.2543	quick	0.3012
know	0.2443	how	0.2940
because	0.2263	munchkins	0.2758

Table 6: Top features that jointly identify the satisfied **female** customers and the Starbucks and Dunkin' Donuts brands from the best model.

Starbucks	weight	Dunkin'	weight
starbucks	0.6632	dd	0.7491
throw	0.3514	dunkin	0.6075
know	0.2959	dds	0.5333
store	0.2885	donuts	0.5326
fix	0.2498	donut	0.3215
particular	0.2487	dunks	0.3158
sbux	0.2462	morning	0.3095
patio	0.2349	rush	0.3030
prefer	0.2324	fast	0.2979
clover	0.2215	moving	0.2520
corporate	0.2153	glazed	0.2326

Table 7: Top features that jointly identify the satisfied **male** customers and the Starbucks and Dunkin' Donuts brands from the best model.

6 Discussions

In our preliminary experiments, we have also experimented with the setup where the two keywords "starbucks" and "dunkin" were removed from the list of features. This change resulted in a uniformed 2% decrease in performances across all the models in Table 2, which did not affect the comparisons.

However, we kept these two keywords in our final experiments, because the reviewers sometimes mention "Dunkin" in Starbucks reviews, and vice versa. Removing the two keywords could be problematic, since it changes the natural distribution of the data.

Regarding the alternative problem setups, our preliminary experiments showed that instead of using one-vs-all binary classifiers, a direct 9-way multi-class classification of joint brand-satisfaction labels using logistic regression only resulted an accuracy of 62%. We also did not adopt the hierarchical classification pipeline, where instead of performing joint classification, multiple layers of classifiers could be trained to classify brand, gender, and satisfaction labels incrementally. This is because the hierarchical classifiers suffered from the error propagation problem, and the second/third layer classifier could not correct the errors from the previous layers (Bennett and Nguyen, 2009).

Our proposed method to generate inter-feature affinity matrix captures interesting dependency of features in this dataset. For example, although the words "frappuccino" and "slurping", "furniture" and "mismatched' are semantically very different, our method actually group them together due to the subtle interactions of these word pairs in our tasks. The example in Figure 1 is also very specific to our dataset. This is very useful, because the word semantic similarity might be context-dependent, and our method learns and adapts the semantic similarity on the fly, hinges on the particular training set. On the other end of the spectrum, even though our method is desirable in our task, one might need to be cautious when working on very small data sets with only a handful of samples. This is because small samples typically have large variances in feature distributions, and that the generated Laplacian matrix might not be as reliable as in our study. To alleviate this potential problem, one might consider building the Laplacian matrix using external resources such as WordNet or FrameNet, even though this approach could also introduce biases due to the mismatched task domains.

We also observed that the accuracy of the automatic brand identification task was high, indicating the promising future of CBA for hidden brand information from other genres of text over the Web. Although the performances of joint brand-satisfaction and joint brand-gender-satisfaction predictions are relatively lower, there is still much room for improvements: for example, using the syntactic, semantic, and meta-data features could potentially enrich the proposed model. Also, it is possible to consider the higher order n-gram features for better exploratory data analysis. However, since the focus of this paper is a proof of concept for Laplacian structured sparsity models and computational branding analytics, we have not yet explored various multi-view representations to augment our model.

Why does Laplacian structured sparsity model work better in these classication tasks? Similar to the application in image classifcation (Gao et al., 2010), one advantage of Laplacian regularization in text classification is that our model can explicitly model the dependency of local features. Another reason is the expressiveness of our model: our model allows one to express the feature interactions in a structured manner. Thirdly, by embedding the structure in the regularization term, our model is more flexible: one can now control the structured penalty by tuning the regularization parameter on the development set.

7 Conclusions

We introduce a Laplacian structured sparsity model for computational branding analytics (CBA). In the automatic brand identification, our model achieves the best result, dominating many competitive baselines. We also introduce the tasks of joint brand-satisfaction and brand-gender-satisfaction predictions, and show that the Laplacian structured sparsity do well in these tasks. A closer evaluation that varying the levels of sparsity in a L_1 constrained problem also indicates that the Laplacian augmented L_1-ball projection model can provide state-of-the-art results. By examining the weights of the derived Laplacian structured sparsity model, interesting indicators of brands and theirs gender-specific customer satisfaction associations are also discovered. In the future, we would like to investigate other methods for generating robust inter-feature Laplacians that include deep syntactic and semantic features.

Acknowledgement

The authors would like to thank the anonymous reviewers for valuable comments.

References

M. Belkin and P. Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems (NIPS)*.

M. Belkin, P. Niyogi, and V. Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research (JMLR)*.

P. N. Bennett and N. Nguyen. 2009. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. 1999. When is nearest neighbor meaningful? *Proceedings of the International Conference on Database Theory (ICDT)*.

F. Biadsy, W.Y. Wang, A. Rosenberg, and J. Hirschberg. 2011. Intoxication detection using phonetic, phonotactic and prosodic cues. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech 2011)*.

V. Chahuneau, K. Gimpel, B.R. Routledge, L. Scherlis, and N.A. Smith. 2012. Word salad: Relating food prices and descriptions.

C.C. Chang and C.J. Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

W. Y. Chen, Y. Song, H. Bai, C. J. Lin, and E. Y. Chang. 2011. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Y. N. Chen, W. Y. Wang, and A. I. Rudnicky. 2013. An empirical investigation of sparse log-linear models for improved dialogue act classification. In *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*.

W. W. Cohen. 2012. Learning similarity measures based on random walks. In *Procceedings of the 21nd ACM International Conference on Information and Knowledge Management (CIKM)*.

L. DiCarlo. 2004. Dunkin' donuts vs. starbucks. In *Forbes.com - Monday Matchup*.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. 2008. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning (ICML)*.

J. Eisenstein, A. Ahmed, and E. Xing. 2011a. Sparse additive generative models of text. *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*.

J. Eisenstein, N. A. Smith, and E. P. Xing. 2011b. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*.

S. Gao, I. W. Tsang, L. T. Chia, and P. Zhao. 2010. Local features are not lonely–laplacian sparse coding for image classification. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

S. Gao, I. Tsang, and L. Chia. 2012. Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

P. M. Guadagni and J. D. C. Little. 1983. A logit model of brand choice calibrated on scanner data. *Marketing science*.

M. K. Kim, K. Lopetcharat, and M. A. Drake. 2013. Influence of packaging information on consumer liking of chocolate milk. *Journal of dairy science*.

A.A. Kuehn. 1962. Consumer brand choice as a learning process.

S. Le Cessie and JC Van Houwelingen. 1992. Ridge estimators in logistic regression. *Applied statistics*.

Chris Lederer and Sam Hill. 2001. See your brands through your customers eyes. *Harvard Business Review*, 79(6):125–133.

S.I. Lee, H. Lee, P. Abbeel, and A.Y. Ng. 2006. Efficient l1 regularized logistic regression. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

D.C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*.

D. Luo, H.G. Hosseini, and J.R. Stewart. 2004. Application of ann with extracted parameters from an electronic nose in cigarette brand identification. *Sensors and Actuators B: Chemical*.

J. C. Makens. 1965. Effect of brand preference upon consumers perceived taste of turkey meat. *Journal of Applied Psychology*.

A. F. T. Martins, N. A. Smith, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2011. Structured sparsity in structured prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Moon and J. Quelch. 2006. *Starbucks: delivering customer service*. Harvard Business School.

M. OConnell and G. Gooding. 2006. The use of first names to evaluate reports of gender and its effect on the distribution of married and unmarried couple households. In *Proceedings of the Annual Meetings of the Population Association of America*.

S. Ovide. 2011. Face off! dunkin' donuts vs. starbucks. In *Deal Journal - Wall Street Journal Blogs*.

M. Paul and M. Dredze. 2012. Factorial lda: Sparse multi-dimensional text models. In *Advances in Neural Information Processing Systems (NIPS)*.

F. Pelisson, D. Hall, O. Riff, and J. Crowley. 2003. Brand identification using gaussian derivative histograms. *Computer Vision Systems*.

R. Rifkin and A. Klautau. 2004. In defense of one-vs-all classification. *The Journal of Machine Learning Research (JMLR)*.

M. Schmidt, G. Fung, and R. Rosales. 2007. Fast optimization methods for l1 regularization: A comparative study and two new approaches. *Machine Learning*.

M. Schmidt, E. Van Den Berg, M. Friedlander, and K. Murphy. 2009. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Proceedings of Conference on Artificial Intelligence and Statistics (AIStats)*.

R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*.

W. Y. Wang and J. Hirschberg. 2011. Detecting levels of interest from spoken dialog with multistream prediction feedback and similarity based hierarchical fusion learning. In *Proceedings of the 12th annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2011)*.

W. Y. Wang, S. Finkelstein, A. Ogan, A. W. Black, and J. Cassell. 2012a. "love ya, jerkface": using sparse log-linear models to build positive (and impolite) relationships with teens. In *Proceedings of the 13th annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2012)*.

W. Y. Wang, E. Mayfield, S. Naidu, and J. Dittmar. 2012b. Historical analysis of legal opinions with a sparse mixed-effects latent variable model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*.

K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul. 2007. Graph laplacian regularization for large-scale semidefinite programming. *Advances in neural information processing systems (NIPS)*.

H. Zou and T. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

Mining New Business Opportunities: Identifying Trend related Products by Leveraging Commercial Intents from Microblogs

Jinpeng Wang[1], **Wayne Xin Zhao**[1], **Haitian Wei**[2], **Hongfei Yan**[1] **and Xiaoming Li**[1]
[1]School of Electronic Engineering and Computer Science, Peking University, China
[2]Daton Securities Co., Ltd., No.93 Jianguo Rd, Chaoyang District, Beijing, China
JooPoo@pku.edu.cn, {batmanfly,haataa.wei,yhf1029}@gmail.com, lxm@pku.edu.cn

Abstract

Hot trends are likely to bring new business opportunities. For example, "Air Pollution" might lead to a significant increase of the sales of related products, e.g., mouth mask. For e-commerce companies, it is very important to make rapid and correct response to these hot trends in order to improve product sales. In this paper, we take the initiative to study the task of how to identify trend related products. The major novelty of our work is that we automatically learn commercial intents revealed from microblogs. We carefully construct a data collection for this task and present quite a few insightful findings. In order to solve this problem, we further propose a graph based method, which jointly models relevance and associativity. We perform extensive experiments and the results showed that our methods are very effective.

1 Introduction

A trend is a hot topic (e.g., the release of a popular movie) which is being widely discussed by the public. Hot trends usually attract much attention from the public, and they are likely to bring new business opportunities. Consider the following e-shopping scenario. A user in Beijing would like to buy something to reduce the health impacts of Beijing air pollution[1]. Different from traditional e-shopping stories, in this case the user may not have a clear idea of what she should buy, and cannot even formulate the purchase needs into a clear query. Faced with trend-driven business opportunities, e-commerce companies typically ask workers to manually identify related products and make heuristic rules to match user queries (e.g., incorporating trending keywords into related product titles).

To improve trend-driven e-commerce, in this paper, we propose to study the novel task of *automatically identifying trend related products*. Why is it compelling to understand and study trend-driven product purchase? Because hot trends are closely related to business opportunities directly or indirectly. As a case of direct causal relationship, the worldwide popularity of the movie series "Harry Potter" created the great success of the original novels of "Harry Potter". As a case of indirect causal relationship, the stock rise or salary increase might exert positive effects on product sale. Based on our empirical analysis (See Section 3), a considerable proportion, i.e. 50%, of hot trends discussed on the largest Chinese microblog (i.e. Sina Weibo) indeed have corresponding product entries in the largest Chinese *C2C* e-commerce website (i.e. Taobao), which indicates a strong correlation between hot trends and product sale.

Although the task is important and emergent, it has at least two major challenges. First of all, how to infer users' trend-driven purchase intents promptly. A trend usually happens unexpectedly. Without prior knowledge and experiences, it is particularly difficult to make rapid response to relate the trend to candidate products. Our solution is to *leverage trend-related commercial intents from microblogs* by mining users' real-time response to a trending topic. We adopt the solution based on two key considerations: (1) Microblogs are fast. As previous studies showed,

[1]See http://www.nytimes.com/2013/04/04/world/asia/two-major-air-pollutants-increase-in-china.html to find more details about the trending topic "Beijing Air Pollution".

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1337–1347,
Seattle, Washington, USA, 18-21 October 2013.

the first story of a trending topic indeed was usually reported in microblogs rather than traditional news media (Sakaki et al., 2010; Kwak et al., 2010; Leskovec et al., 2009). (2) Microblogs contain users' commercial intents. The microblogging service has become one of the most popular social network platforms, where users may tweet about their needs and desires (Hollerit et al., 2013). E.g., a microblog user may complain about the air quality and evince the desire to buy a mouth mask in a tweet. The example indicates we can make use of tweet-level relatedness to capture the correlation between *trends* and *products*.

Second, how to achieve a comprehensive coverage of related products but not hurting precision. The above solution will miss the related products which have not been discussed in microblogs. Our idea is to take the associativity between products into consideration. Our definition about associativity is very general and can have different instantiations in specific settings. For example, we can define product associativity to be the similarity between product descriptions, or the ratio of historical purchase records in e-commerce companies. However, one-step associativity may not fully discover the underlying relatedness between products due to the fact that the product associativity is indeed transitive. Thus, a transitable associativity model is needed.

To address these two challenges, we propose a unified graph based ranking algorithm which jointly models the above two aspects, i.e., relevance and associativity. Given a trend, the algorithm runs in an iterative way and seeks a trade-off between relevance and associativity by propagating the scores on the product graph. Our contribution can be summarized as follows: (1) we introduce the novel task of identifying trend related products, most of all, we propose to leverage trend-related commercial intents from microblogs; (2) we present insightful empirical analysis to illustrate the correlation between hot trends and product sale (See Section 3); (3) we propose a novel graph based ranking algorithm which jointly considers relevance and associativity; (4) we carefully construct the test collection based on real data of the largest microblog and the largest *C2C* e-commerce website in China. (5) we perform extensive experiments and present some important implications for practice.

To the best of our knowledge, our work was the first to consider identifying trend related products by leveraging commercial intents from microblogs. We believe the current work will have important impact on industry and inspire more follow-up research studies. The rest of this paper is organized as follows. We present the data collection and empirical analysis of the impact of hot trends on product sale in Section 3. We present a novel graph-based method in Section 4. Experimental setup and results are discussed in Section 5 and Section 6. Finally, the related work is discussed in Section 7. And the conclusions and future work are given in Section 8.

2 Problem Definition

A *trend* is a hot topic widely discussed by the public, e.g., the release of a hot movie. Usually, a trend e can be described by a small set of keywords denoted by $\mathcal{K}_e$ and a corresponding time span $\mathcal{T}_e$.

Trend-related Products Identification: Given a trend e, we assume that the following inputs are available: 1) tweets that contain trend keywords $\mathcal{K}_e$ and 2) a product database which provides a set of candidate products $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ with necessary detailed information, e.g., titles and descriptions. The objective of trend-related products identification is to identify products in $\mathcal{P}$ that are related to trend e within the time span $\mathcal{T}_e$, denoted by $\mathcal{P}_R$. For convenience, we will not explicitly mention the time span unless needed.

To better understand the problem, we first present an illustrative example in Table 1, which will be discussed as the running case throughout the paper. In this example, we can see that a few users tweet their product needs related to the trend "Air Pollution". We take Taobao as the product database and present a few related products in it.

Table 1: An illustrative example for the studied task.

Trend keywords: Air Pollution
Tweets:
What bad air! We need to buy masks ASAP!!!
I am planing to buy an air purifier. Hoping it can defend air pollution.
#air_pollution I will recommend to keep some houseplants at home.
Product database: Taobao[2]
Related products: Mouth Mask, Air Purifier, Houseplant

[2]The biggest C2C e-commerce site in China, similar to eBay.

3 Data and Observations

As discussed earlier, hot trends may exert positive effects on the sale of related products. In this section, we will construct a deep analysis on this point by presenting quantitative answers to the following two problems:

- Q_1: What is the proportion of hot trends that potentially lead to business opportunities, and how is their impact on related products?
- Q_2: How is the associativity between related products?

These findings are key and fundamental to develop our models.

3.1 Data Collection

To perform the above analysis, the key is how to construct an experimental data collection which relates hot trends to corresponding related products. We jointly consider microblogs and e-commerce platforms: we obtain hot trends in microblogs and manually identify trend-related products in e-commerce websites. In this paper, we adopt Sina Weibo[3] as the microbloging platform and Taobao[4] as the e-commerce platform, which are the biggest microblogging service and the largest C2C company in China respectively. The analysis method is general and can equally apply to other platforms. For both two data signals, we consider a two-month time span, i.e. from May 2013 to June 2013.

Trend detection. Since trend detection is not our focus, we directly obtained trends from "trending topics" provided by the microblog platform. Our work can be easily extended to incorporate a trend detection component. Similar to "trending topics" in Twitter, Sina Weibo provides a public list of top searched keywords which can be obtained by the Weibo search API[5]. In the list, top 50 keywords are presented and ordered by the number of being searched. Weibo classifies these keywords into five categories: *China*, *movie*, *business*, *person* and *sports*. We consider these keywords to be trend keywords. These keywords are dynamically updated and we monitor the trend lists in the considered time span. We define the start and end time of a trend to be the first day and the last day on the trend list respectively, which spans the active interval of a trend. We only keep the trend which has an active interval with more than one day. For each trend, we use the trend keywords to retrieve all related tweets in the active interval, and use the pattern based method in (Hollerit et al., 2013) to extract all mentioned product keywords. We present a few example patterns used for extracting product keywords in Table 2. After that we can obtain a set of product keywords for each trend.

Table 2: Example patterns for extracting product keywords.

Patterns	Example segments of tweets
买*(buy)*	买了飞利浦剃须刀送父亲 *bought father a Philips PT720 (Electric Razor).*
使用*(use)*	使用N95口罩降低污染 *use N95 (mouth mask) to reduce the impact of bad air*
推荐 *(recommend)*	推荐Galaxy S4 *recommend Galaxy S4 (cell phone)*

Related product identification and annotation. For each trend, we have the product keyword set together with the trend keywords as described above. We use these keywords to retrieve candidate products in the product search engine of Taobao within the active interval of the trend. For each candidate product, we further crawl its product page and obtain corresponding related products suggested by Taobao, which are treated as candidate, too. We invite two senior post-graduate students major in economics as human judges. The judge is required to make a binary decision whether a product is related to a trend by following a detailed guideline compiled by a senior officer of an e-commerce company in Beijing. For each trend, we provide the trend keywords, product keywords in tweets, related tweets, related news articles from China Daily[6]. Web access is available during the annotation process. Due to space limit, we do not present the annotation guideline here. We use Cohen's kappa to measure the agreement of these two judges, which has a high value of 0.75. To speed up the work, we further group all products which have the same lowest categorial label (e.g., leaf label) [7], and we

[3]http://www.weibo.com/

[4]http://www.taobao.com/

[5]http://s.weibo.com/top/summary

[6]http://www.chinadaily.com.cn

[7]Taobao has provided a category tree for products: http://list.taobao.com/browse/cat-0.htm).

will treat a group as a product in later experiments. We only keep the products with the same judgments and the trends with at least one related product. We present the statistics of data set in Table 3 [8]. Since current e-commerce search engines mainly adopt keyword matching based retrieval method, we further examine the performance of simply using trend keywords as queries. We compute the percentage of related/unrelated products with at least one trend keyword in their description. We can see that only 29.7% related products can be found on average. These statistics indicate that more effective methods are needed for the current task.

Table 3: Statitics of the data set.

# business-related trends	113
average candidate products per trend	55.1
average related products per trend	7.3
average perc. of rel. prod. with trend keywords	29.7%
average perc. of unrel. prod. with trend keywords	6.3%

3.2 Observations

Now we analyze the data collection and present our observations.

A_1: First of all, it is important to find out the proportion of hot trends that potentially leads to business opportunities. Recall that each trend has a category label and possibly a set of related products identified by the judges. We refer to a trend with related products as a business-related trend. We present the statistics in Fig. 1. We can see that about 36% of all trends have corresponding related products in Taobao, which indicates that these trends highly relate to business. *Movies* and *Sports* have higher proportions of business-related trends, i.e. 81% and 52% respectively, while the other categories have lower proportions but still with a substantial number of business related trends. It is noteworthy that *Business* has the lowest proportion, the major reason is that trends in *Business* are usually general events, i.e., the release of new economic policy, which do not directly correspond to related products. As we discussed earlier, these trends may have indirect impact on product sales. Currently, we only focus on direct impact, and indirect impacts will be considered in future work.

[8]The data set can be downloaded at `http://sewm.pku.edu.cn/~wjp`.

Next we continue to examine the impact of hot trends on the sale of related products. We obtain product sales from Taobao product pages. As we can see in Fig. 2, the average sale of related products in all categories gradually increased with trends going on. Interestingly, we can see that categories *Movies* and *China* achieved very significant increase. Products related to *Movies* trends are usually related to the movie itself, e.g., movie tickets; while products related to *China* tend to be commodities (e.g., the mouth masks for the trend of "Air Pollution") or trending products (e.g., Shenzhou-10 Spacecraft Model for the trend of "the launch of Shenzhou-10").

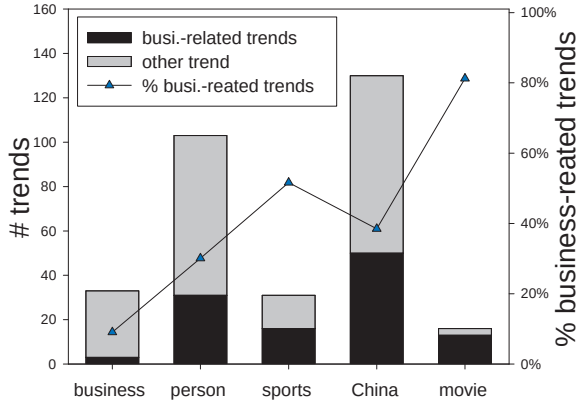


Figure 1: The proportion and volume of business-related trends in five categories.

A_2: Recall we have discussed that product associativity is useful for improving the coverage of related products. Here we would like to quantitatively examine the associativity between related products given a trend. For a trend, we first compute the average pairwise similarity between *related* products in terms of their descriptive texts (e.g. title and description). Since there are more unrelated products, we randomly sample an equal number of unrelated products from the candidate products we previously generated. Then we compute the average similarity between a related product and an unrelated product. We further average these values over all the trends of each category. The average similarity of related-related product pairs is 0.112, while the average similarities of unrelated-unrelated and related-unrelated

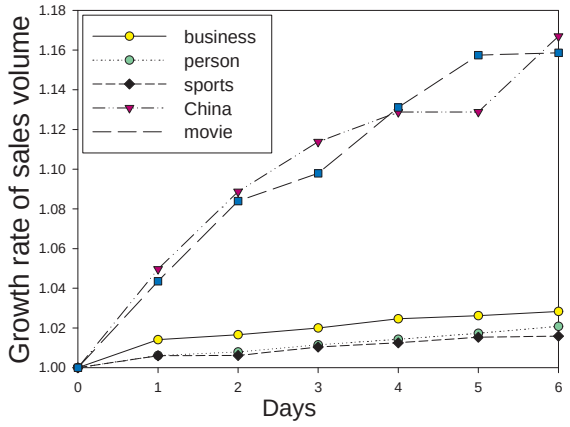


Figure 2: An illustrative analysis of the impact on related products in five categories. We measure the impact by computing the average growth ratio of sale in Taobao.

product pairs are 0.039 and 0.058 respectively.[9]

In summary, A_1 indicates that a large proportion of hot trends are potentially related to products and will exert positive effects on product sale; A_2 indicates that there is a strong associativity between related products, which can be utilized to improve both precision and recall of the algorithm.

4 The Proposed Method

In this section, we present a graph based ranking algorithm jointly models the relevance of a product and the associativity between products. Recall that we have collected a set of product keywords and a set of candidate products for each trend. Our aim is to re-rank these candidate products to obtain a better ranking of related products. We adopt a biased random walk algorithm: 1) relevance is modeled as biased restart probability and 2) associativity is modeled through random walk on the product graph.

4.1 Modeling the Product Relevance

Recall that in Section 2 we use the pattern based method to extract product keywords from tweets related to a trend. However, to stimulate the real scenario that we want to identify the related products at the beginning of a trend, we only keep the keywords which were contained in tweets published in the first three days when a trend began. These extracted product keywords directly reveal users' commercial intents on the trends. Instead of modeling personalized intents, we consider learning a unified trend-driven intent by representing the intent as a weighted vector over these product keywords. And the key is how to set the keyword weight.

Keyword weighting. A good weighting method should be able to leverage commercial interests/intents of users well and emphasize the keywords users really focus on. Thus we consider making use of the *retweeting* (a.k.a. *forwarding*) mechanism in microblogs. Retweet links are shown to be better in revealing relevance and interests (Welch et al., 2011). Formally, we use the following weighting formula for a keyword k:

$$\text{Weight}(k) = \sum_{t \in \mathcal{C}_k} \log_{10}(\#rt_t + 1), \quad (1)$$

where $\mathcal{C}_k$ is the set of all originally-written tweets (i.e., not a retweet) that contain the keyword k in the considered time span, and $\#rt_t$ is the retweet number of a tweet t. We further normalize and build the weight vector over all the considered keywords, called as *intent vector*. We denote the intent vector of a trend e by $\vec{e}$.

Product relevance. Having the intent vector, now we discuss about how to define the product relevance. Given a product p, we extract all the words in the title and description parts of a product. We represent it as a vector using the widely *tf-idf* weighting method. We denote the weight vector of product p by $\vec{p}$. We measure the product relevance between e and p as $\text{rel}(e,p) = \frac{\vec{e}\cdot\vec{p}}{|\vec{e}||\vec{p}|}$.

4.2 Modeling the Associativity between Products

To start this part, we first present an illustrative example in Fig. 3. We can see there are four related products for the trend "Air Pollution". We assume that only "mouth mask" was mentioned in microblogs. Now we expect to mine more related products with "mouth mask" as a known related product. We can compute the similarity between a pair of products. Intuitively, if the similarity between a candidate product and "mouth mask" is higher than a predefined threshold, we can consider it to be related, too. In this example, "air detector" and "air purifier" are similar to "mouth mask" in terms of prod-

[9]The difference was tested to be statistically significant.

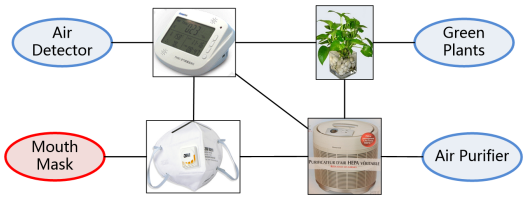


Figure 3: An example to illustrate the importance of associativity. Products which were mentioned in tweets related to "Air Pollution" are marked in red circles while the others are marked in blue circles. The link between products indicate the similarity between two products. Links with weights lower than a predefined threshold are not considered. Although "green plants" is related to this trend, it was not mentioned in tweets and did not have a direct link to "mouth mask".

uct descriptions and considered to be related, while "Green plants" is determined to be unrelated since it has very little overlap words with "mouth mask" in the description. It indicates one-step similarity method is not able to fully capture the real associativity between products.

Thus, we propose to use the random walk method to propagate the relatedness score on the product graph. Let P denote the number of all the candidate products, and $\boldsymbol{r}_{P\times 1}$ denote the relatedness score vector where r_i denote the relatedness score of product p_i.

We first construct the product graph. We represent each candidate product as a vertex in the graph and built the link with the cosine similarity between the descriptive texts of two products as the link weight.[10] We denote the similarity matrix by $\mathbf{M}_{P\times P}$ and $M_{i,j}$ denotes the similarity between products p_i and p_j. Formally, we formulate the problem in a standard PageRank form

$$\boldsymbol{r}^{(n+1)} = \mu \cdot \boldsymbol{r}^{(n)} \cdot \mathbf{M} + (1-\mu)\cdot \boldsymbol{y}, \qquad (2)$$

where $\boldsymbol{y}$ is the restart probability vector usually set to be uniform. With this method, it is easy to see that relatedness score can be propagated on the product graph, which better captures underlying associativity between products.

[10]Other similarity methods can be used, e.g., co-purse history record.

4.3 Jointly Modeling Relevance and Associativity

Having discussed about how to model both relevance and associativity, now we are ready to present a joint model to capture these two factors. By following (Zhao et al., 2013), the main idea is that instead of using a uniform restart distribution $\boldsymbol{y}$, we use an relevance biased restart distribution in Eq. 2. We set the restart probability of a product to its corresponding relevance. Formally, we set $y_i = \text{rel}(e, p_i)$. Let us further explain the idea. At the beginning of each iteration, each product is first assigned to its relevance score: the more relevant it is, the larger score it has. During the iteration, each product begins to collect relevance evidence from its neighbors on the product graph: the more relevant neighbors it has, the larger score it obtains. And the final score is indeed a trade-off between its own relevance score and neighboring relevance scores it receives. In order to obtain an ergodic walk, we add a small value, i.e. $1e-4$, to each entry of $\boldsymbol{y}$ and then normalize this vector. We denote our algorithm as JMRA (Jointly Modeling Relevance and Associativity).

To have an intuitive understanding of our algorithm, let us turn to the example in Fig. 3 again. At the beginning, only "mouth mask" has a large relevance score, with the iteration going on, the relatedness score will be propagated between products on the graph. Although "green plants" has not a direct link with "mouth mask", it can obtain relatedness score from its neighbors, i.e. "air detector" and "air purifier". JMRA is able to discover such latent associativity between products.

5 Experimental Setup

We use the test collection which have been described in Section 3. The statistics of the data set is shown in Table 3.

5.1 Evaluation Metrics

For a real product search engine, top results are particularly important, thus we adopt *precision@5* and *precision@10* as the evaluation metrics. Similar to Information Retrieval, we also consider using Mean Average Precision (MAP) as metrics to measure the overall quality of retrieved products.

5.2 Methods to Compare

We compare the following methods for inferring relating products:

SALES: we rank the candidate products by their historical sales volume descendingly.

TREND: we use trend keywords as queries and rank the products by their relevance.

TREND+fb: based on *TREND*, we further incorporate pseudo-relevance feedback (Salton, 1971; Salton and Buckley, 1997). After some tuning (See Section 6.5), top 3 search results were used to update the query.

JMRA$_r$: it is our method which only considers product relevance in Section 4.1.

JMRA$_r + fb$: we further apply pseudo-relevance feedback to *JMRA*$_r$.

JMRA$_{r+a}$: it is our method which considers both relevance and associativity in Section 4.3.

JMRA$_{r+a} + fb$: we further apply pseudo-relevance feedback to *JMRA*$_{r+a}$.

6 Experimental Results and Analysis

In this section, we first evaluate the performance of the proposed approach and the comparison methods. Next, we analyze a problem in real e-commerce search engines, i.e., the cold start. Then, we give a qualitative case study to further demonstrate the effectiveness of the proposed approach. Finally, we examine the parameter sensitivity to the performance.

6.1 Comparison of Performance

We present the results of various methods in Table 4. We first examine the performance of baselines *SALES*, *TREND* and *TREND+fb*. First, *SALES* has the worst performance due to the fact that a trend usually happen unexpectedly and historical records may not predict it well. The second observation is that the improvement of *TREND+fb* over *TREND* is little. This is mainly because that very few related products can be identified only based on trend keywords so feedback method does not work very well on it.

Then we compare our relevance based methods with the above three baselines. Note that the major difference between *JMRA*$_r$ and *TREND* is that *JMRA*$_r$ makes uses of both trend keywords and product keywords extracted from microblogs. We can see that *JMRA*$_r$ performs better than all three baselines. It proves the effectiveness of leveraging commercial intents from microblogs. Another interesting point is that the relative improvement *JMRA*$_r$*+fb* over *JMRA*$_r$ is larger than that *TREND+fb* over *TREND*. The reason is that pseudo relevance feedback relies highly on top results and a system with better search quality will benefit more from it.

Finally, we consider evaluating our full models which jointly consider relevance and associativity. It is easy to see *JMRA*$_{r+a}$ yields a significant improvement over *JMRA*$_r$ and even outperforms *JMRA*$_r$*+fb*. This observation supports our assumption that product associativity is very important in this task. Again, pseudo relevance feedback has also improved *JMRA*$_{r+a}$.

In summary, our results have shown some important implications for trend-related product retrieval on e-commerce search engines: 1) microblogs are very good signals to learn users' commercial intents; 2) product associativity is particularly important; 3) other advanced retrieval methods are potentially useful, e.g., pseudo relevance feedback.

Table 4: The overall performance of all the methods.

Models	P@5	P@10	MAP
SALES	0.345	0.379	0.225
TREND	0.543	0.325	0.327
TREND+fb	0.550	0.325	0.328
JMRA$_r$	0.611	0.527	0.336
JMRA$_r$*+fb*	0.661	0.552	0.348
JMRA$_{r+a}$	0.733	0.609	0.392
JMRA$_{r+a}$*+fb*	**0.734**	**0.624**	**0.404**

6.2 Cold Start

It is noteworthy that we have considered all the candidate products within the entire active interval of a trend when constructing the test collection. This is mainly to obtain a good coverage of related products since some e-commerce companies might release new products as the response to a trend. During the active interval of a trend, the e-commerce companies may make some heuristic rules to enhance the retrieval of related products, e.g., incorporating trend keywords into product titles and descriptions. In the

real application scenario, an effective method is expected to identify related products at the beginning of a trend when the e-commerce workers may not make any response to the trend. How would it be if we do not have the manually generated trend keywords from workers in product titles and descriptions?

To answer the question, in this part, we continue to examine the impact of cold start on different methods. We select three methods as comparisons, i.e., *TREND+fb*, *JMRA*$_r$ + fb and *JMRA*$_{r+a}$ + fb. We first use keyword matching methods to obtain all the products that related to trend keywords. The descriptive text (i.e., title and description) of these products has been refined to match trend queries by sellers in e-commerce websites. We further removed all the trend keywords in the desriptive text of these products, and gradually add the trend keywords back to original products. In such a process, we would like to examine how cold start affects the performance of different methods.

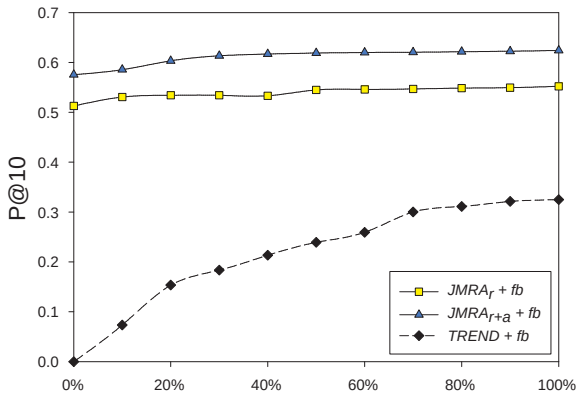


Figure 4: **The impact of cold start for different methods.**

We present the results in Fig. 4. First, performance of all these methods improve with the increase of products with trend keywords. Second, cold start does not affect the relative performance order of different methods, i.e., *TREND+fb* < *JMRA*$_r$ + fb < *JMRA*$_{r+a}$ + fb. Finally, all the methods display similar impact patterns: "significantli increasing" → "stable". An interesting observation is that *JMRA*$_r$ + fb and *JMRA*$_{r+a}$ + fb have much more stable performance compared to *TREND+fb*. It indicates that our methods are very robust to the cold start, and potentially applicable in real e-commerce search engines.

6.3 Case Study

In order to have a intuitive understanding of how different method perform, we present a case study in this part. We select *JMRA*$_r$, *JMRA*$_r$*+fb* and *JMRA*$_{r+a}$*+fb* as comparisons. The results are shown in Table 5. We can see that *JMRA*$_{r+a}$*+fb* have identified the most related products. We further analyze the contribution of different factors. Compared with *JMRA*$_r$, *JMRA*$_r$*+fb* has got one more related product "Houseplant" due to the reason pseudo feedback can make use of top related search results to improve the queries. In this case, "Houseplant" is identified to be related because it is very similar to "Air Detector" (We have presented the corresponding most associative products in brackets in Table 5). Similarly, the comparison between *JMRA*$_r$*+fb* and *JMRA*$_{r+a}$*+fb* shows the effectiveness of product associativity.

6.4 Error Analysis

To further understand the shortcomings of the proposed methods, we use the example in Table 5 for error analysis. Based on our manual inspection, errors may arise from two major sources for our method:

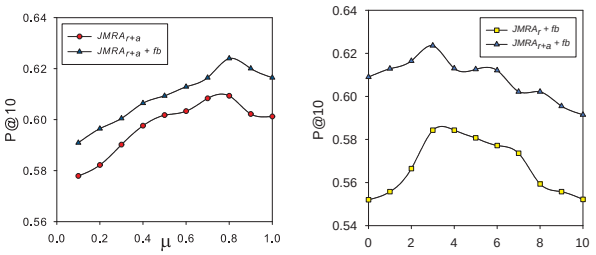
- Product keyword extraction errors: we use a pattern-based product keyword extraction method, and it tends to incorporate some irrelevant words. For example, given the topic "air pollution", users would talk about the impact of "car exhaust" on air quality and advocate to reduce automobile usage and sale. The current keyword extraction method might mistake the word "car" for a product related keyword.

- Search engine retrieval errors: in this paper, we rely on the Taobao product search engine for candidate product generation. It is highly based on surface-form matching to retrieve related products. Therefore, given a query "mouth mask", it might return some irrelevant products, e.g., "party mask". Clearly, pseudo-relevance feedback will also bring additional irrelevant products if top search results contain irrelevant ones.

Table 5: A qualitative comparison of three methods on the topic of "Air Pollution". We mark related products in bold.

Sample keywords learnt from microblogs:
air pollution, mouth mask, air, air purifier, respirator, house, mask, warm, bus, car, purified water

$JMRA_r$	$JMRA_r$+*fb*	$JMRA_{r+a}$+*fb*
Mouth Mask	**Mouth Mask**	**Mouth Mask**
Air Detector	**Air Detector**	**Air Purifier**
Air Purifier	**Air Purifier**	**Air Detector**
Toy House	Humidifier	**Respirator**
Respirator	**Respirator**	**Oxygen Bag (*Mouth Mask*)**
Toy Car	Party Mask	Humidifier
Environment-friendly Bags	Toy Car	**Houseplant (*Air Detector*)**
Humidifier	**Houseplant (*Air Detector*)**	**Anti-pollution Medicine (*Oxygen Bag*)**
Purified Water	Environment-friendly bags	Purified Water
Warmer	Purified Water	Party Mask



(a) (b)

Figure 5: Parameter sensitivity. a) The impact of the damping factor μ and b) the impact of the number of top products used for pseudo feedback.

To solve these problems, one promising way is to leverage more context information about the candidate products and construct deep semantic analysis. We will leave it as future work.

6.5 Parameter Sensitivity

The only parameter for *JMRA* is the damping factor in the random walk model, i.e., μ. Intuitively, a larger value of μ emphasize the associativity more while a smaller value emphasize the relevance more. We tune this parameter at a step of 0.1 and present the results in Fig. 5(a). We can see the performance of $JMRA_{r+a}$+*fb* is consistently better than that of $JMRA_r$+*fb* and peaks at around "0.8". It indicates the robustness of $JMRA_{r+a}$+*fb* and the importance of product associtivity.

We further examine the impact of the number of top products used for pseudo feedback for $JMRA_r$+*fb* and $JMRA_{r+a}$+*fb*. In Fig. 5(b), we can see that both $JMRA_{r+a}$+*fb* and $JMRA_r$+*fb* achieved their best at "3". It indicates that we only need to consider very top results for pseudo feedback.

6.6 Title or Description?

In previous experiments, for each product, we used the descriptive text in both title and description. In this part, we consider examining the individual effect of *title* and *description*. We use $JMRA_{r+a}$+*fb* as the examined method since both relevance and associativity relies on the text information.

Table 6: Evaluating the performance of $JMRA_{r+a}$+*fb* with different text sources.

sources	P@5	P@10	MAP
title	0.690	0.591	0.364
description	0.711	0.602	0.387
title+description	0.734	0.624	0.404

As shown in Table 6, we can see that the performance of only using *description* is better than that of only using *title* and a combination of both parts achieve the best. *title* is usually carefully compiled by e-commerce sellers, thus it reveals the most highlights of the products but very short; while *description* contains more informative text but tends to incorporate noise. In future work, we will consider a more principled way to combine *title* and *description*, e.g., weighted combination.

6.7 Efficiency

Finally, we present a few discussions about the issue of efficiency. All codes were implemented in Python 2.7, and all experiments were performed on a PC with *Intel(R) Core(TM)i5 CPU 760 @ 2.8GHz* and 8GB memory.

Sicne we group products by the categorial label, the number of candidate products is usually very small. Thus, our method *JMRA* runs very efficiently. Even on an extremely large set of candidate products, the iterative random walk algorithm can be easily implemented in a distributed way (Bahmani et al., 2011) and would have very good efficiency.

7 Related Work

Our work is mainly related to the following lines:

Mining the microblogs. Microblogs have been one of the most popular social networking platforms, and they have recently attracted much attention from research communities. The studies on trend (or event) detection (Benson et al., 2011; Weng and Lee, 2011; Sakaki et al., 2010; Zhao et al., 2012) tried to make use of the rapid response of microblogs users as the signal to automatically identify external events. Another important aspect is the content analysis of tweets, including the recommendation of real-time topical news (Phelan et al., 2009), sentiment or opinions analysis (Meng et al., 2012), event summarization using tweets (Chakrabarti and Punera, 2011; Lin et al., 2012; Zhao et al., 2013), etc. Our work do not explicitly incorporate a trend detection component, instead we make use of the trending topics provided by the microblogs platforms. It will be easy to incorporate other trend detection methods as our input.

Identifying online users' commercial intents. The identification of online users' commercial intents has been quite an important research problem in the past. Most researches focus on capturing commercial intention from search queries (Dai et al., 2006; Strohmaier and Kröll, 2012), click-through behaviors (Ashkan and Clarke, 2009), users' mouse movements or scrolling behaviors (Guo and Agichtein, 2010) and search logs (Strohmaier and Kröll, 2012). The most related to our work is the work in (Hollerit et al., 2013), which attempts to detect commercial intent on twitter. But we have very different focus. They aim to identify tweet-level commercial intents while ours aim to identify trend-driven commercial intents. In addition, we also present how to make use of these identified intents and our paper focuses on how to identify trend related products for e-commerce companies to improve service when faced with hot trends.

8 Conclusions

In this paper, we make the first attempt to identify trend related products by leveraging commercial intents from microblogs. We propose a way to construct the evaluation set for this task and present some insightful findings. We propose a graph based method to joint model relevance and associativity. We perform extensive experiments, including quantitative and qualitative analysis.

Currently, our approach is indeed a framework to solve this task, and we may consider improving the individual components in it, e.g. consider non-product keywords in tweets. For future work, we will consider incorporating a trend detection component into our method, which can be more flexible to adapt to various trend signals. We can also refine the method of the product keyword extraction by using more principled solutions.

Acknowledgments

We thank the anonymous reviewers for the constructive comments. The work was partially supported by NSFC Grant 60933004, 61073082 and 61272340. Jinpeng Wang was supported by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority. We thank Taobao for the access to the product data and all Taobao data in this paper will be only used for research purpose.

References

Azin Ashkan and Charles LA Clarke. 2009. Term-based commercial intent analysis. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 800–801. ACM.

Bahman Bahmani, Kaushik Chakrabarti, and Dong Xin. 2011. Fast personalized pagerank on mapreduce. In

Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, SIGMOD '11, pages 973–984, New York, NY, USA. ACM.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

Deepayan Chakrabarti and Kunal Punera. 2011. Event summarization using tweets. In *Proceedings of the 5th Int'l AAAI Conference on Weblogs and Social Media (ICWSM)*, July.

Honghua Kathy Dai, Lingzhi Zhao, Zaiqing Nie, Ji-Rong Wen, Lee Wang, and Ying Li. 2006. Detecting online commercial intention (oci). In *Proceedings of the 15th international conference on World Wide Web*, pages 829–837. ACM.

Qi Guo and Eugene Agichtein. 2010. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137. ACM.

Bernd Hollerit, Mark Kröll, and Markus Strohmaier. 2013. Towards linking buyers and sellers: detecting commercial intent on twitter. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 629–632. International World Wide Web Conferences Steering Committee.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 497–506, New York, NY, USA. ACM.

Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. 2012. Generating event storylines from microblogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 175–184. ACM.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. 2012. Entity-centric topic-oriented opinion summarization in twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 379–387. ACM.

Owen Phelan, Kevin McCarthy, and Barry Smyth. 2009. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388. ACM.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, New York, NY, USA. ACM.

Gerard Salton and Chris Buckley. 1997. Readings in information retrieval. chapter Improving retrieval performance by relevance feedback, pages 355–364. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Gerard Salton, editor. 1971. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, Englewood, Cliffs, New Jersey.

Markus Strohmaier and Mark Kröll. 2012. Acquiring knowledge about human goals from search query logs. *Information Processing & Management*, 48(1):63–82.

Michael J. Welch, Uri Schonfeld, Dan He, and Junghoo Cho. 2011. Topical semantics of twitter links. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 327–336, New York, NY, USA. ACM.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in Twitter. In *Proceedings of the Fifth International Conference on Weblogs and Social Media (ICWSM)*, Menlo Park, CA, USA. AAAI.

Wayne Xin Zhao, Baihan Shu, Jing Jiang, Yang Song, Hongfei Yan, and Xiaoming Li. 2012. Identifying event-related bursts via social media activities. In *EMNLP-CoNLL*, pages 1466–1477.

Wayne Xin Zhao, Yanwei Guo, Rui Yan, Yulan He, and Xiaoming Li. 2013. Timeline generation with social attention. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '13, pages 1061–1064.

Using Topic Modeling to Improve Prediction of Neuroticism and Depression in College Students

Philip Resnik
University of Maryland
College Park, MD 20742
resnik@umd.edu

Anderson Garron
University of Maryland
College Park, MD 20742
agarron@cs.umd.edu

Rebecca Resnik
Mindwell Psychology Bethesda
5602 Shields Drive
Bethesda, MD 20817
drrebeccaresnik@gmail.com

Abstract

We investigate the value-add of topic modeling in text analysis for depression, and for neuroticism as a strongly associated personality measure. Using Pennebaker's Linguistic Inquiry and Word Count (LIWC) lexicon to provide baseline features, we show that straightforward topic modeling using Latent Dirichlet Allocation (LDA) yields interpretable, psychologically relevant "themes" that add value in prediction of clinical assessments.

1 Introduction

In the United States, where 25 million adults per year suffer a major depressive episode (NAMI, 2013), identifying people with mental health problems is a key challenge. For clinical psychologists, language plays a central role in diagnosis: many clinical instruments fundamentally rely on what is, in effect, manual coding of patient language. Automating language assessment in this domain potentially has enormous impact, for two reasons. First, conventional clinical assessments for affective disorders (e.g. The Minnesota Multiphasic Personality Inventory, MMPI) are based on norm-referenced self-report, and therefore depend on patients' willingness and ability to report symptoms. However, some individuals are motivated to underreport symptoms to avoid negative consequences (e.g. active duty soldiers, parents undergoing child custody evaluations), and others lack the self awareness to report accurately.[1] Second, many people – e.g. those without adequate insurance or in rural areas – cannot access a clinician qualified to perform a psychological evaluation (Sibelius, 2013; APA, 2013). There is enormous value in inexpensive screening measures that could be administered in primary care and by social workers and other providers.

We take as a starting point the well known lexicon-driven methods of Pennebaker and colleagues (LIWC, Pennebaker and King (1999)), which relate language use to psychological variables, and improve on them straightforwardly using topic modeling (LDA, Blei et al. (2003)). First, we show that taking automatically derived topics into account improves prediction of neuroticism (emotional instability, John and Srivastava (1999)), as measured by correlation with widely used clinical instruments, when compared with lexically-based prediction alone. Neuroticism is of particular interest as a personality measure because higher scores on neuroticism scales are consistent with increased distress and more difficulty coping; individuals with high levels of neuroticism may also be at higher risk of psychiatric problems categorized as Axis I in the DSM-IV (Association, 2000), including the internalizing disorders (depression, anxiety).[2] Second, we show a similar correlation improvement result for prediction of depression, adding improvement

[1] As evidenced by the fact that assessments such as the MMPI-2-RF include validity scales to detect, e.g., defensiveness, atypical responses, and overly positive self-portrayals (Tellegen et al., 2003).

[2] The Diagnostic and Statistical Manual of Mental Disorders is a widely used organization of mental health conditions; it served as the standard for diagnosis from 1994 until the release of the (quite controversial) DSM-5 in May, 2013. Axis I in the DSM-IV includes all the major diagnostic categories, excluding mental retardation and personality disorders.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1348–1353,
Seattle, Washington, USA, 18-21 October 2013.

on precision (with no decrease in recall), as well as comparison with human performance by clinical psychologists. Third, we show that LDA has identified meaningful, population-specific themes that go beyond the pre-defined LIWC categories and are psychologically relevant.

2 Predicting neuroticism

2.1 Experimental framework

Data. We utilize a collection of 6,459 stream-of-consciousness essays collected from college students by Pennebaker and King (1999) between 1997 and 2008, averaging approximately 780 words each. Students were asked to think about their thoughts, sensations, and feelings in the moment and "write your thoughts as they come to you".

Each essay is accompanied by metadata for its author, which includes scores for the Big-5 personality traits (John and Srivastava, 1999): agreeableness, conscientiousness, extraversion, neuroticism, and openness. Because Big-5 assessment can be done using a variety of different survey instruments (John et al., 2008), and different instruments were used from year to year, we treat the data from each year as an independent dataset.

Any author missing any Big-5 attribute was excluded. Essays were tokenized using NLTK (Bird et al., 2009) and lowercased, eliminating those that failed tokenization because of encoding issues. This resulted in a dataset containing 4,777 essays with associated Big-5 metadata.

LIWC features. For each document, we calculate the number of observed words in each of Pennebaker and King's 64 LIWC categories. These include, among others, syntactic categories (e.g. pronouns, verbs), affect categories (e.g. negative emotion words, anger words), semantic categories (e.g. causation words, cognitive words), and topical categories (e.g. health, religion). For instance, the anger category contains 190 word patterns specifying, for example, words descriptive of contexts involving anger (e.g. *brutal*, *hostile*, *shoot*) and words that would be used by someone when angry (e.g. *bullshit*, *hate*, *moron*). We also explored including essays' average sentence length and total word-count, as an initial proxy for language complexity, which often figures into psychological assessments. However, results adding these features to LIWC did not differ significantly from LIWC alone, and for brevity we do not report them.[3]

LDA features. We use vanilla LDA as implemented in the Mallet toolkit (McCallum, 2002), developing a k-topic model on just training documents, and using the posterior topic distribution for each training and test document as a set of k features. Mallet's stoplist and default parameters were used for burn-in, lag, number of iterations, priors, etc. Details on train/test splits and number k of topics appear below.

LIWC+LDA features. The union of the LIWC features (one feature per category) and LDA features (one feature per topic).

Prediction. We utilize linear regression in the WEKA toolkit (Hall et al., 2009), estimated on training documents, to predict the neuroticism score associated with the author of each test document.[4]

2.2 Results

Table 1 shows the quality of prediction via linear regression, averaged over the eleven datasets, 1997 through 2008, using Pearson correlation (r) as the evaluation metric. For each year, we used 10-fold cross-validation to ensure proper separation of training and test data. We experimented with LDA using 20, 30, 40, and 50 topics.[5]

A first thing to observe is that the multiple regressions using all LIWC categories produce much stronger correlations with neuroticism than the individual category correlations reported by Pennebaker and King.[6] There the strongest individual correlations with neuroticism for any LIWC categories are .16 (negative emotion words) and -.13 (positive emotion words), though it should be noted that their goal was to validate their categories as a meaningful

[3] Using richer measures of complexity, e.g. Pakhomov et al. (2011), is a topic for future work.

[4] In previous work we have found that multiple linear regression is competitive with more complicated techniques such as SVM regression, though we plan to explore the latter in future work.

[5] Full year-by-year data appears in supplemental materials at `http://umiacs.umd.edu/~resnik/papers/emnlp2013-supplemental/`.

[6] The comparison is not perfect, since they used Big-5 data collected between 1993 and 1998, and we also eliminated some files during preprocessing.

Feature set	LIWC	LDA20	LIWC+LDA20	LDA30	LIWC+LDA30	LDA40	LIWC+LDA40	LDA50	LIWC+LDA50
Average r	0.413	0.384	0.430	0.407	0.442*	0.420	0.459**	0.440	0.443*

Table 1: Prediction quality for neuroticism, for alternative feature sets (Pearson's r). *p$<$ .03, **$p < .02$

way to explore personality differences, not prediction.

As noted in Table 1, paired t-tests (df=10, $\alpha = .05$) establish that, in comparing the cross-year averages, augmenting LIWC with topic features improves average correlation significantly over using LIWC features alone for 30, 40, and 50 topics. LDA features alone do not improve significantly over LIWC.

3 Predicting Depression

3.1 Experimental framework

Data. We use essays collected by Rude et al. (2004) similarly to §2.1; in this case, students were asked to "describe your deepest thoughts and feelings about being in college". Each essay is accompanied by the author's Beck Depression Inventory score (BDI). BDI (Beck et al., 1961) is a widely used 21-question instrument that correlates strongly with ratings of depression by psychiatrists. Following Rude et al., we treat BDI ≥ 14 as the threshold for a positive instance of depression.[7] Text preprocessing was done as in §2.1, with 124 documents in total averaging around 390 words each.

Training/test split. Because only 12 of 124 authors met the BDI ≥ 14 threshold, we did not split randomly, lest the test sample include too few positive instances to be useful. Instead we included a random 6 of the 12 above-threshold cases, plus 24 more items sampled at random, to create a 30-item test set. To form the training set from the complementary items, we added two more copies of each positive instance to help address class imbalance (Batista et al., 2004) and, following Rude et al., we excluded items with with BDI of 0 or 1 as potentially invalid.

Human comparison. We created a set of expert human results for comparison by asking three practicing clinical psychologists to review the test documents and rate whether or not the author is suffering depression.[8] They were asked to "decide how at-risk this person might be for depression", assigning 0 (no significant concerns), 1 (mild concerns, but does not require futher evaluation), or 2 (requires attention, refer for further evaluation). Following recommended practice for cases where different labels are not equally distinct from each other (Artstein and Poesio, 2008), we evaluate inter-coder agreement using Krippendorff's α; our α, computed for ordinal data, is 0.722.

Features. We ran 50-topic LDA on the 4,777 essays from §2.1 plus the BDI training items, using the posterior topic distributions as features as in §2.1. As in §2.1, the LIWC features comprised one count per LIWC category, and LIWC+LDA features were the union of the two.

3.2 Results

Regression on LIWC features alone achieved $r = .288$, and adding topic features improved this substantially to $r = .416$. Treating BDI ≥ 14 as the threshold for positive instances (i.e. that an author is depressed), Table 2 shows that adding topic features improves precision without harming recall. Automatic prediction is more conservative than human ratings, trading recall for precision to achieve comparable F-measure on this test set.[9]

[7] Each question contributes a score value from 0 to 3, so BDI scores range from 0 to 63.

[8] These psychologists all have doctoral degrees, are licensed, and spend significant time primarily in assessment and diagnosis of psychological disorders. None were familiar with the specifics of this study.

[9] A reviewer observes, correctly, that in a scenario where a system is providing preliminary screenings to aid psychologists, the precision/recall tradeoff demonstrated here would potentially be undesirable, since a presumed goal would be to not miss any cases, even at the risk of some false positives. We note, however, that the real world is unfortunately replete with situations where there is significant cost or social/professional stigma associated with interventions or follow-up testing; in such situations it might be high precision that is desirable. These are challenging questions, and the ability to trade off precision versus recall more flexibly is a topic we are interested in investigating in future work.

	P	R	F1
LIWC	.43	.50	.46
LIWC+LDA	.50	.50	.50
Rater 1	.38	.83	.52
Rater 2	.33	.83	.47
Rater 3	.33	.66	.44

Table 2: Prediction quality for depression.

4 Qualitative themes

In order to explore the relevance of the themes uncovered by LDA, the third author, a practicing clinical psychologist, reviewed the 50 LDA categories created in §3.1. Each category, represented by its 20 highest-probability words, was given a readable description. Then, for each category, she was asked: "If you were conducting a clinical interview, would observing these themes in a patient's responses make you more (less) likely to feel that the patient merited further evaluation for depression?"

Table 3 shows the seven topics selected as particularly indicative of further evaluation.[10] These capture population-specific properties in ways that LIWC cannot — for example, although LIWC does have a *body* category, it does not have a category that corresponds to somatic complaints, which often co-occur with depression. Similarly, some words related to energy level, e.g. *tired*, would be captured in LIWC's *body*, *bio*, and/or *health* category, but the LDA theme corresponding to low energy or lack of sleep, another potential depression cue, contains words that make sense there only in context (e.g. *tomorrow*, *late*). Other themes, such as the one labeled HOMESICKNESS, are clearly relevant (potentially indicative of an adjustment disorder), but even more specific to the population and context.

5 Related Work

The application of NLP to psychological variables has seen a recent uptick in community activity. One recent shared task brings together research on the Big-5 personality traits (Celli et al., 2013; Kosinski et al., 2013), and another involved research on identification of emotion in suicide notes (Pestian et al., 2012). Other examples include NLP research on autistic spectrum disorders (Van Santen et al., 2010; Prudhommeaux et al., 2011; Lehr et al., 2013) and dementia (Pakhomov et al., 2011; Lehr et al., 2012; Roark et al., 2011).

With regard to depression, Neuman et al. (2012) develop a corpus-based "depression lexicon" and produce promising screening results, and De Choudhury et al. (2013) predict social network behavior changes related to post-partum depression. Neither, however, evaluates using formal instruments for clinical assessment.

Related investigations involving LDA include Zhai et al. (2012), who use LIWC to provide priors for corpus-specific emotion categories; Stark et al. (2012), who combine LIWC and LDA-based features in classification of social relationships; and Schwartz et al. (2013), who use lexical and topic-based features in Twitter to predict life satisfaction.

6 Conclusions

In this paper, we have aimed for a small, focused contribution, investigating the value-add of topic modeling in text analysis for depression, and for neuroticism as a strongly associated personality measure. Our contribution here is not technical: corpus-specific topics/themes are anticipated by Zhai et al. (2012), and Stark et al. (2012) employ topic-based features for prediction in a supervised setting. Rather, our contribution here has been to show that topic models can get us beyond the LIWC categories to relevant, population-specific themes related to neuroticism and depression, and to support that claim using evaluation against formal clinical assessments. More data (e.g. Kosinski et al. (2013)) and more sophisticated models (e.g. supervised LDA, Blei and McAuliffe (2008), and extensions such as Nguyen et al. (2013)) will be the key to further progress.

Acknowledgments

We are grateful to Jamie Pennebaker for the LIWC lexicon and for allowing us to use data from Pennebaker and King (1999) and Rude et al. (2004), to the three psychologists who kindly took the time to provide human ratings, and to our reviewers for helpful comments. This work has been supported in part by NSF grant IIS-1211153.

[10] All 50 can be found in the supplemental materials at `http://umiacs.umd.edu/~resnik/papers/emnlp2013-supplemental/`.

VEGETATIVE/ENERGY LEVEL	sleep tired night bed morning class early tomorrow wake late asleep long hours day sleeping nap today fall stay time
SOMATIC	hurts sick eyes hurt cold head tired back nose itches hate stop starting water neck hand stomach feels kind sore
NEGATIVE/TROUBLE COPING	don('t) hate doesn care didn('t) understand anymore feel isn('t) stupid make won('t) wouldn talk scared wanted wrong mad stop shouldn('t)
ANGER/FRUSTRATION	hate damn stupid sucks hell shit crap man ass god don blah thing bad suck doesn fucking fuck freaking real
HOMESICKNESS	home miss friends back school family weekend austin parents college mom lot boyfriend left houston visit weeks wait high homesick
EMOTIONAL STRESS	feel feeling thinking makes make felt feels things nervous scared lonely feelings afraid moment happy worry comfortable stress excited guilty
ANXIETY	feel happy things lot sad good makes bad make hard mind happen crazy cry day worry times talk great wanted

Table 3: LDA-induced themes related to depression.

References

[APA2013] APA. 2013. The critical need for psychologists in rural america. http://www.apa.org/about/gr/education/rural-need.aspx, Downloaded September 16, 2013.

[Artstein and Poesio2008] Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4):555–596, December.

[Association2000] American Psychiatric Association. 2000. *Diagnostic and Statistical Manual of Mental Disorders, 4th Edition, Text Revision (DSM-IV-TR)*. American Psychiatric Association, 4th edition, July.

[Batista et al.2004] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June.

[Beck et al.1961] Aaron T Beck, Calvin H Ward, Mock Mendelson, Jeremiah Mock, and J1 Erbaugh. 1961. An inventory for measuring depression. *Archives of general psychiatry*, 4(6):561.

[Bird et al.2009] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly.

[Blei and McAuliffe2008] David Blei and Jon McAuliffe. 2008. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 121–128. MIT Press, Cambridge, MA.

[Blei et al.2003] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

[Celli et al.2013] Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. 2013. Computational personality recognition (shared task) workshop. In *International Conference on Weblogs and Social Media*. AAAI.

[De Choudhury et al.2013] Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013. Predicting postpartum changes in emotion and behavior via social media. In *Proceedings of the 2013 ACM annual conference on Human factors in computing systems*, pages 3267–3276. ACM.

[Hall et al.2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[John and Srivastava1999] Oliver P John and Sanjay Srivastava. 1999. The big five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality: Theory and research*, 2:102–138.

[John et al.2008] O. P. John, L. P. Naumann, and C. J. Soto. 2008. Paradigm shift to the integrative big five trait taxonomy: History, measurement, and conceptual issues. In *Handbook of personality: Theory and research*, pages 114–158.

[Kosinski et al.2013] Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

[Lehr et al.2012] Maider Lehr, Emily Tucker Prud'hommeaux, Izhak Shafran, and Brian Roark. 2012. Fully automated neuropsychological assessment for detecting mild cognitive impairment. In *INTERSPEECH*.

[Lehr et al.2013] Maider Lehr, Izhak Shafran, Emily Prudhommeaux, and Brian Roark. 2013. Discriminative joint modeling of lexical variation and acoustic confusion for automated narrative retelling assessment. In *Proceedings of NAACL-HLT*, pages 211–220.

[McCallum2002] Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

[NAMI2013] NAMI. 2013. Major depression fact sheet, April. http://www.nami.org/Template.cfm?Section=depression.

[Neuman et al.2012] Yair Neuman, Yohai Cohen, Dan Assaf, and Gabbi Kedma. 2012. Proactive screening for depression through metaphorical and automatic

text analysis. *Artif. Intell. Med.*, 56(1):19–25, September.

[Nguyen et al.2013] Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Neural Information Processing Systems*.

[Pakhomov et al.2011] Serguei Pakhomov, Dustin Chacon, Mark Wicklund, and Jeanette Gundel. 2011. Computerized assessment of syntactic complexity in alzheimers disease: a case study of iris murdochs writing. *Behavior Research Methods*, 43(1):136–144.

[Pennebaker and King1999] James W Pennebaker and Laura A King. 1999. Linguistic styles: language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296.

[Pestian et al.2012] John P Pestian, Pawel Matykiewicz, Michelle Linn-Gust, Brett South, Ozlem Uzuner, Jan Wiebe, K Bretonnel Cohen, John Hurdle, Christopher Brew, et al. 2012. Sentiment analysis of suicide notes: A shared task. *Biomedical Informatics Insights*, 5(Suppl. 1):3.

[Prudhommeaux et al.2011] Emily T Prudhommeaux, Brian Roark, Lois M Black, and Jan van Santen. 2011. Classification of atypical language in autism. *ACL HLT 2011*, page 88.

[Roark et al.2011] Brian Roark, Margaret Mitchell, J Hosom, Kristy Hollingshead, and Jeffrey Kaye. 2011. Spoken language derived measures for detecting mild cognitive impairment. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(7):2081–2090.

[Rude et al.2004] Stephanie Rude, Eva-Maria Gortner, and James Pennebaker. 2004. Language use of depressed and depression-vulnerable college students. *Cognition & Emotion*, 18(8):1121–1133.

[Schwartz et al.2013] H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Megha Agrawal, Gregory J. Park, Shrinidhi K. Lakshmikanth, Sneha Jha, Martin E. P. Seligman, and Lyle Ungar. 2013. Characterizing geographic variation in well-being using tweets. In *Seventh International AAAI Conference on Weblogs and Social Media (ICWSM 2013)*.

[Sibelius2013] Kathleen Sibelius. 2013. Increasing access to mental health services, April. http://www.whitehouse.gov/blog/2013/04/10/increasing-access-mental-health-services.

[Stark et al.2012] Anthony Stark, Izhak Shafran, and Jeffrey Kaye. 2012. Hello, who is calling?: can words reveal the social nature of conversations? In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 112–119. Association for Computational Linguistics.

[Tellegen et al.2003] A. Tellegen, Y.S. Ben-Porath, J.L. McNulty, P.A. Arbisi, and B. Graham, J.R.and Kaemmer. 2003. *The MMPI-2 Restructured Clinical Scales: Development, validation, and interpretation*. Minneapolis, MN: University of Minnesota Press.

[Van Santen et al.2010] Jan PH Van Santen, Emily T Prud'hommeaux, Lois M Black, and Margaret Mitchell. 2010. Computational prosodic markers for autism. *Autism*, 14(3):215–236.

[Zhai et al.2012] Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad L. Alkhouja. 2012. Mr. lda: a flexible large scale topic modeling package using variational inference in mapreduce. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 879–888, New York, NY, USA. ACM.

Predicting the resolution of referring expressions from user behavior

Nikos Engonopoulos[1] **Martín Villalba**[1] **Ivan Titov**[2] **Alexander Koller**[1]
[1]University of Potsdam, Germany [2]University of Amsterdam, Netherlands
{nikolaos.engonopoulos, martin.villalba}@uni-potsdam.de
titov@uva.nl, koller@ling.uni-potsdam.de

Abstract

We present a statistical model for predicting how the user of an interactive, situated NLP system resolved a referring expression. The model makes an initial prediction based on the meaning of the utterance, and revises it continuously based on the user's behavior. The combined model outperforms its components in predicting reference resolution and when to give feedback.

1 Introduction

Speakers and listeners in natural communication are engaged in a highly interactive process. In order to achieve some communicative goal, the speaker will perform an utterance which they believe has a high chance of achieving that goal. They will then monitor the listener's behavior to see whether this goal is actually being achieved. This process is a core part of what is commonly called *grounding* in the dialogue literature (see e.g. (Clark, 1996; Traum, 1994; Paek and Horvitz, 1999; Hirst et al., 1994)). Interactive computer systems that are to carry out an effective and efficient conversation with a user must model this grounding process, and should ideally respond to the user's observed behavior in real time. For instance, if the user of a pedestrian navigation system takes a wrong turn, the system should interpret this as evidence of misunderstanding and bring the user back on track.

We focus here on the problem of predicting how the user has resolved a *referring expression (RE)* that was generated by the system, i.e. a noun phrase that is intended to identify some object uniquely to the listener. A number of authors have recently offered statistical models for parts of this problem. Golland et al. (2010) and Garoufi and Koller (2011) have presented log-linear models for predicting how the listener will resolve a given RE in a given scene; however, these models do not update the probability model based on observing the user's reactions. Nakano et al. (2007), Buschmeier and Kopp (2012), and Koller et al. (2012) all predict what the listener understood based on their behavior, but do not consider the RE itself in the model. The models of Frank and Goodman (2012) and Vogel et al. (2013) aim at explaining the effect of implicatures on the listener's RE resolution process in terms of hypothesized interactions, but do not actually support a real-time interaction between a system and a user.

In this paper, we show how to predict how the listener has resolved an RE by combining a statistical model of RE resolution based on the RE itself with a statistical model of RE resolution based on the listener's behavior. To our knowledge, this is the first approach to combine two such models explicitly. We consider the RE grounding problem in the context of interactive, situated natural language generation (NLG) for the GIVE Challenge (Koller et al., 2010a), where NLG systems must generate real-time instructions in virtual 3D environments. Our evaluation is based on interaction corpora from the GIVE-2 and GIVE-2.5 Challenges, which contain the systems' utterances along with the behavior of human hearers in response to these utterances. We find that the combined model predicts RE resolution more accurately than each of the two component models alone. We see this as a first step towards implementing an actual interactive system that performs human-like grounding based on our RE resolution model.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1354–1359,
Seattle, Washington, USA, 18-21 October 2013.

Figure 1: An example scene in the GIVE environment.

2 Problem definition

In the GIVE Challenge, an interactive NLG system faces the task of guiding a human instruction follower (IF) through a treasure-hunt game in a virtual 3D environment (see Fig. 1). To complete the task, the IF must press a number of buttons in the correct order; these buttons are the colored boxes in Fig. 1, and are scattered all over the virtual environment. The IF can move around freely in the virtual environment, but has no prior knowledge about the world. The NLG system's task is to guide the IF towards the successful completion of the treasure-hunt task. To this end, it is continuously being informed about the IF's movements and visual field, and can generate written utterances at any time. As a comparative evaluation effort, the GIVE Challenges connected NLG systems to thousands of users over the Internet (see e.g. Koller et al. (2010a) for details).

Many system utterances are *manipulation instructions*, such as "press the blue button", containing an RE in the form of a definite NP. We call a given part of an interaction between the system and the IF an *episode* of that interaction if it starts with a manipulation instruction, ends with the IF performing an *action* (i.e., pressing a button), and contains only IF movements and no further utterances in between. Not all manipulation instructions initiate an episode, because the system may decide to perform further utterances (not containing REs) before the IF performs their action. An NLG system will choose the RE for an instruction at runtime out of potentially many semantically valid alternatives ("the blue button", "the button next to the chair", "the button to the right of the red button", etc.). Ideally, it will predict which of these REs has the highest chance to be understood by the IF, given the current scene, and utter an instruction that uses this RE.

After uttering the manipulation instruction, the system needs to ascertain whether the IF understood the RE correctly, i.e. it must engage in grounding. A naive grounding mechanism might wait until the IF actually presses a button and check whether it was the right one. This is what many NLG systems in the GIVE Challenges actually did. However, this can make the communication ineffective (IF performs many useless actions) and risky (IF may press the wrong button and lose). Thus, it is important that the system updates its prediction of how the IF resolved the RE continuously by observing the IF's behavior, *before* the actual button press. For instance, if the IF walks towards the target, this might reinforce the system's belief in a correct understanding; turning away or exiting the room could be strong evidence of the opposite. The system can then exploit the updated prediction to give the IF feedback ("no, the *blue* button") to prevent costly mistakes.

We address these challenges by estimating the probability distribution over the possible objects to which the IF may resolve the RE. We then update this distribution in real time by observing the IF's movements. More specifically, assume that a system tries to refer to some object a^* among some set A of available objects. Given an RE r generated for a^* at time t_0, the state of the world s at t_0, and the observed behavior $\sigma(t)$ of the user at $t \geq t_0$, we estimate the probability $p(a|r, s, \sigma(t))$ that the user resolved r to an object $a \in A$. When generating the instruction, an optimal NLG system will use the RE r that maximizes $p(a^*|r, s, \sigma(t_0))$. It can then track $p(a|r, s, \sigma(t))$ for time points $t > t_0$ throughout the episode, and generate feedback when $p(a'|r, s, \sigma(t))$ exceeds $p(a^*|r, s, \sigma(t))$ for some $a' \neq a^*$; that is, when the updated probability distribution predicts that the IF resolved r to an incorrect button.

3 A model of RE resolution

In order to model the distribution over possible objects, we assume the following generative story: when receiving an instruction containing an RE r at a given world state s, the IF resolves it to an object a; depending on the object a, the IF then moves towards it, exhibiting behavior σ. These assumptions correspond to the following factorization:

$$p(a, \sigma|r, s) = p(\sigma|a)p(a|r, s)$$

The posterior probability distribution over objects a can be obtained by applying the Bayes rule and using the above assumptions:

$$p(a|r, s, \sigma) \propto p(a|r, s)p(a|\sigma)/p(a)$$

For simplicity, we assume a uniform $p(a)$ over all objects in a world. We can thus represent $p(a|r, s, \sigma)$ as the normalized product of a *semantic* model $p_{sem}(a|r, s)$ and an *observational* model $p_{obs}(a|\sigma)$. We use log-linear models for both, and train them separately. The feature functions we use only consider general properties of objects (such as color and distance), and not the identity of the objects themselves. This means that we can train a model on one virtual environment (containing a certain set of objects), and then apply the model to another virtual environment, containing a different set of objects.

Semantic model The semantic model estimates for each object a in the environment the initial probability $p_{sem}(a|r, s)$ that the IF will understand a given RE r uttered in a scene s as referring to a. It represents the meaning of r, contextualized to s, and is only ever evaluated at the time t_0 of the utterance. The features used by this model are:

- **Semantic features** aim to encode whether r is a good description of a. *IsColorModifying* evaluates to 1 if a's color appears as an adjective modifying the head noun of r, e.g. "the blue button". *IsRelPosModifying* evaluates to 1 if a's relative position to the IF is mentioned as an adjective in r, e.g. "the left button".
- **Confusion features** capture the hypothesis that the IF may be confused by the description of a landmark when resolving the RE; e.g. an RE like "the button next to the red button" might confuse the IF into pressing a red button, rather than the one meant by the system. These are the same features as in the Semantic case, but looking for modifier keywords in the entire RE, including the head.
- **Salience features** account for the fact that an IF is more likely to resolve r to a if a was visually salient in s. *IsVisible* evaluates to 1 if a is visible to the IF in s. *IsInRoom* evaluates to 1 if the IF and a are in the same room. *IsTargetInFront* evaluates to 1 if the *angular distance* towards a, i.e. the absolute angle between the camera direction and the straight line from the IF to a, is less than $\frac{\pi}{4}$. *VisualSalience* approximates the visual salience of Kelleher and van Genabith (2004), a weighted count of the number of pixels on which a is rendered (pixels near the center of the screen have higher weights).

Observational model The observational model estimates for each object a the probability $p_{obs}(a|\sigma)$ that the IF will interact with a, given the IF's recent behavior $\sigma(t) = (\sigma_1, \ldots, \sigma_n)$, where σ_i is the state of the world at time $t - (i - 1) \cdot 500\text{ms}$, and $n \geq 1$ is the length of the observed behavior. p_{obs} is constantly re-evaluated for times $t > t_0$ as the IF moves around. p_{obs} uses the following features:

- **Linear distance features** assume that the closest button is also the one the IF understood. *InRoom* returns the number of frames σ_i in σ in which the IF and a are in the same room. *ButtonDistance* returns the distance between the IF and a at σ_1 divided by a constant such that the result never exceeds 1. If a is neither in the same room nor visible, the feature returns 1.
- **Angular distance features** analyze the direction in which the IF looks. *TargetInFront* returns the angular distance towards a at σ_1. *AngleToTarget* returns *TargetInFront* divided by π, or 1 if a is neither in the same room nor visible. *LinearRegAngleTo* applies linear regression to a list of observed angular distances towards a over all frames σ_i, and returns the slope of the regression as a measure of variation. Negative values indicate that the IF turned towards a, while positive values mean the opposite. If a is neither visible nor in the same room as the IF at σ_i, the angle is set to π.
- **Combined distance feature**: a weighted sum of linear and angular distance towards a, called *overall distance* in Koller et al. (2012).
- **Salience features** capture visual salience and its change over time. Defining VS_i as the result of applying the p_{sem} feature *VisualSalience* to σ_i and a, *LastVisualSalience* returns VS_n. *LinearRegVisualSalience* applies linear regression to all values VS_i and returns the slope as a measure of change in salience. *VisualSalienceSum* returns $(\Sigma_{i=1}^{n}\text{VS}_i) * \text{VS}_1$. This emphasizes the contribution of VS_1, which we assume is the

most reliable predictor of the IF's intentions.

- **Binary features** aim to detect concrete behavior patterns: *LastIsVisible* applies the p_{sem} feature *IsVisible* to σ_1, and *IsClose* evaluates to 1 if the IF is close enough and correctly oriented to manipulate a in the GIVE environment at σ_1.

4 Evaluation

Data We evaluated our model using data from the GIVE-2 (Koller et al., 2010b) and the GIVE-2.5 Challenges (Striegnitz et al., 2011), obtained from GIVE Organizers (2012). These datasets constitute *interaction corpora*, in which the IF's activities in the virtual environment were recorded along with the utterances automatically generated by the participating NLG systems. The data consists of 1833 games for GIVE-2 and 687 games for GIVE-2.5.

To extract training data for our model from the GIVE-2.5 data, we first identified moments in the recorded data where the IF pressed a button. From these, we discarded all instances from the tutorial phase of the GIVE game and those that happened within 200 ms after the previous utterance, as these clearly didn't happen in response to it. This yielded 6478 training instances for p_{obs}, each consisting of σ at 1 second before the action, and the button a which the IF pressed. We chose $n = 4$ for representing σ, except to ensure that the features only considered IF behavior that happened in response to an utterance. We achieved this by reducing n for the first few frames after each utterance, such that the time of σ_n was always after the time of the utterance. Finally, we selected those instances which are episodes in the sense of Section 2, i.e. those in which the last utterance before the action contained an RE r. This gave us 3414 training instances for p_{sem}, each consisting of a, r, the time t_0 of the utterance, and the world state s at time t_0.

We obtained test instances from the GIVE-2 data in the same way. This yielded 5028 instances, each representing an episode. We chose GIVE-2 for testing because the mean episode length is higher (3.3s, vs. 2.0s in GIVE-2.5), thus making the evaluation more challenging. Feature selection was done using the training data and a similar dataset from Koller et al. (2012). Note that the test data and training data are based on distinct sets of three virtual environments each, and were obtained with different NLG systems and users. This demonstrates the ability of our model to generalize to unseen environments.

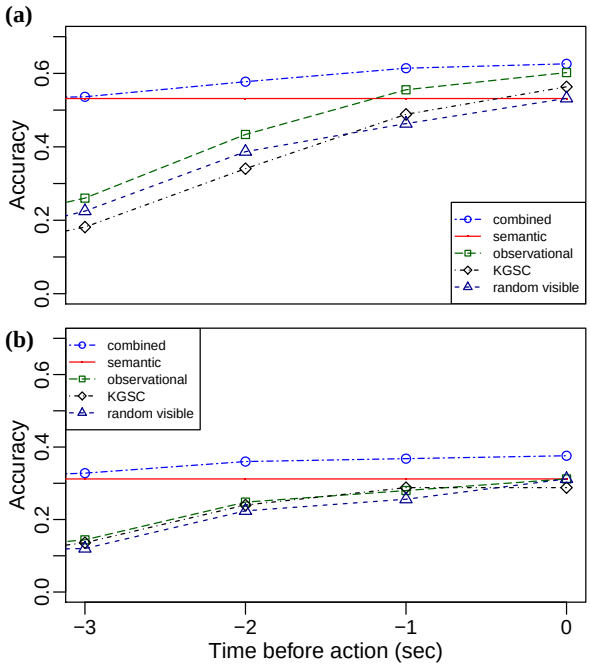


Figure 2: Prediction accuracy for (a) all episodes, (b) unsuccessful episodes as a function of time.

An example video showing our models' predictions on some training episodes can be found at `http://tinyurl.com/re-demo-v`.

Prediction accuracy We first evaluated the ability of our model to predict the button to which the IF resolved each RE. For each test instance $\langle r, s, \sigma, a \rangle$, we compare the object returned by $\arg\max_a p(a|r, s, \sigma(t))$ to the one manipulated by the IF. We call the proportion of correctly classified instances the *prediction accuracy*.

Fig. 2a compares our model's prediction accuracy to that of several baselines. We plot prediction accuracy as a function of the time at which the model is queried for a prediction, by evaluating at 3s, 2s, 1s, and 0s before the button press. The graph is based on the 2094 test instances with an episode length of at least three seconds, to ensure that results for different prediction times are comparable. As expected, prediction accuracy increases as we approach the time of the action. Furthermore, the combined model outperforms both p_{sem} and p_{obs} reliably. This indicates that the component models pro-

vide complementary useful information. Our model also outperforms two more baselines: *KGSC* predicts that the IF will press the button with the minimal *overall distance*, which is the distance metric used by the "movement-based system" of Koller et al. (2012); *random visible* selects a random button from the ones that are currently visible to the IF.

The fact that this last baseline does not approach 1 at action time suggests that multiple buttons tend to be visible when the IF presses one, confirming that the prediction task is not trivial.

Correctly predicting the button that the IF will press is especially useful, and challenging, in those cases where the IF pressed a different button than the one the NLG system intended. Fig. 2b shows a closer look at the 125 unsuccessful episodes of at least three seconds in the test data. These tend to be hard instances, and thus as expected, prediction accuracy drops for all systems. However, by integrating semantic and observational information, the combined model compensates better for this than all other systems, with an accuracy of 37.6% against 31.2% for each individual component.

Feedback appropriateness Second, we evaluated the ability of our model to predict whether the user misunderstood the RE and requires feedback. For all the above models, we assumed a simple feedback mechanism which predicts that the user misunderstood the RE if $p(a') - p(a^*) > \theta$ for some object $a' \neq a^*$, where θ is a confidence threshold; we used $\theta = 0.1$ here. We can thus test on recorded data in which no actual feedback can be given anymore.

We evaluated the models on the 848 test episodes of at least 3s in which the NLG systems logged the button they tried to refer to. The results are shown in Fig. 3 in terms of F1 measure. Here precision is the proportion of instances in which the IF pressed the wrong button (i.e., where feedback should have been given) among the instances where the model actually suggested feedback. Recall is the proportion of instances in which the model suggested feedback among the instances where the IF pressed the wrong button. Again, the combined model outperforms its components and the baselines, primarily due to increased recall. The difference is particularly pronounced early on, which would be useful in giving timely feedback in an actual real-time system.

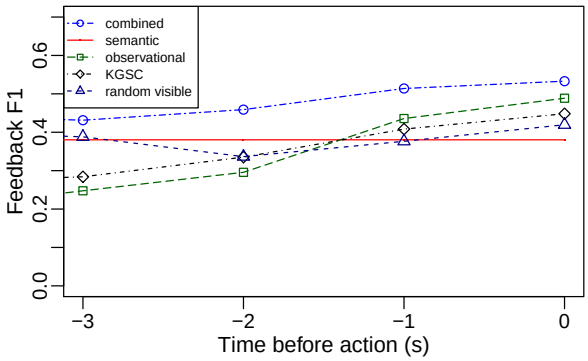


Figure 3: Feedback F1-measure as a function of time.

5 Conclusion and future work

We presented a statistical model for predicting how a user will resolve the REs generated by an interactive, situated NLG system. The model continuously updates an initial estimate based on the meaning of the RE with a model of the user's behavior. It outperforms its components and two baselines on prediction and feedback accuracy.

Our model captures a real-time grounding process on the part of the interactive system. We thus believe that it provides a solid foundation for detecting misunderstandings and generating suitable feedback in an end-to-end dialogue system. We have presented our model in terms of a situated dialogue setting, where clues about what the hearer understood can be observed directly. However, we believe that the fundamental mechanism should apply to other domains as well. This would amount to finding observable linguistic and non-linguistic clues of hearer understanding that can be used as features of p_{obs}.

The immediate next step for future research is to extend our model to an implemented end-to-end situated NLG system for the GIVE Challenge, and evaluate whether this actually improves task performance. This requires, in particular, to compute the RE that is optimal with respect to p_{sem}. We will furthermore improve p_{obs} by switching to a more temporally dynamic probability model.

Acknowledgments. We thank Konstantina Garoufi and the anonymous reviewers for their insightful comments and suggestions. The first two authors were supported by the SFB 632 "Information Structure"; Titov's work was supported by the Cluster of Excellence at Saarland University.

References

Hendrik Buschmeier and Stefan Kopp. 2012. Adapting language production to listener feedback behaviour. In *Proceedings of the Interdisciplinary Workshop on Feedback Behaviors in Dialog*.

Herbert C. Clark. 1996. *Using Language*. Cambridge University Press.

Michael C. Frank and Noah D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998.

Konstantina Garoufi and Alexander Koller. 2011. Combining symbolic and corpus-based approaches for the generation of successful referring expressions. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*.

GIVE Organizers. 2012. Give challenge website: Corpora. `http://give-challenge.org/research/page.php?id=corpora`.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Graeme Hirst, Susan McRoy, Peter Heeman, Philip Edmonds, and Diane Horton. 1994. Repairing conversational misunderstandings and non-understandings. *Speech Communications*, 15:213–229.

J. D. Kelleher and J. van Genabith. 2004. Visual salience and reference resolution in simulated 3-D environments. *Artificial Intelligence Review*, 21(3).

Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010a. The First Challenge on Generating Instructions in Virtual Environments. In E. Krahmer and M. Theune, editors, *Empirical Methods in Natural Language Generation*, number 5790 in LNCS, pages 337–361. Springer.

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010b. Report on the Second NLG Challenge on Generating Instructions in Virtual Environments (GIVE-2). In *Proceedings of the 6th International Natural Language Generation Conference (INLG)*.

Alexander Koller, Konstantina Garoufi, Maria Staudte, and Matthew Crocker. 2012. Enhancing referential success by tracking hearer gaze. In *Proceedings of the 13th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, Seoul.

Yukiko Nakano, Kazuyoshi Murata, Mika Enomoto, Yoshiko Arimoto, Yasuhiro Asa, and Hirohiko Sagawa. 2007. Predicting evidence of understanding by monitoring user's task manipulation in multimodal conversations. In *Proceedings of the ACL 2007 Demo and Poster Sessions*.

Tim Paek and Eric Horvitz. 1999. Uncertainty, utility, and misunderstanding: A decision-theoretic perspective on grounding in conversational systems. In *AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*.

Kristina Striegnitz, Alexandre Denis, Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Mariet Theune. 2011. Report on the Second Second Challenge on Generating Instructions in Virtual Environments (GIVE-2.5). In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*.

David Traum. 1994. *A computational theory of grounding in natural language conversation*. Ph.D. thesis, University of Rochester.

Adam Vogel, Christopher Potts, and Dan Jurafsky. 2013. Implicatures and nested beliefs in approximate Decentralized-POMDPs. In *Proceedings of ACL*.

Chinese Zero Pronoun Resolution: Some Recent Advances

Chen Chen and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{yzcchen,vince}@hlt.utdallas.edu

Abstract

We extend Zhao and Ng's (2007) Chinese anaphoric zero pronoun resolver by (1) using a richer set of features and (2) exploiting the coreference links between zero pronouns during resolution. Results on OntoNotes show that our approach significantly outperforms two state-of-the-art anaphoric zero pronoun resolvers. To our knowledge, this is the first work to report results obtained by an *end-to-end* Chinese zero pronoun resolver.

1 Introduction

A zero pronoun (ZP) is a gap in a sentence that is found when a phonetically null form is used to refer to a real-world entity. An anaphoric zero pronoun (AZP) is a ZP that corefers with one or more preceding noun phrases (NPs) in the associated text. Unlike overt pronouns, ZPs lack grammatical attributes that are useful for overt pronoun resolution such as number and gender. This makes ZP resolution more challenging than overt pronoun resolution.

We aim to improve the state of the art in Chinese AZP resolution by proposing two extensions. First, while previous approaches to this task have primarily focused on employing positional and syntactic features (e.g., Zhao and Ng (2007) [Z&N], Kong and Zhou (2010) [K&Z]), we exploit a richer set of features for capturing the context of an AZP and its candidate antecedents. Second, to alleviate the difficulty of resolving an AZP to an antecedent far away from it, we break down the process into smaller, intermediate steps, where we allow coreference links between AZPs to be established.

We apply our two extensions to a state-of-the-art Chinese AZP resolver proposed by Z&N and evaluate the resulting resolver on the OntoNotes corpus. Experimental results show that this resolver significantly outperforms both Z&N's resolver and another state-of-the-art resolver proposed by K&Z. It is worth noting that while previous work on Chinese ZP resolution has reported results obtained via gold information (e.g., using gold AZPs and extracting candidate antecedents and other features from gold syntactic parse trees), this is the first work to report the results of an *end-to-end* Chinese ZP resolver.

The rest of this paper is organized as follows. Section 2 describes the two baseline AZP resolvers. Sections 3 and 4 discuss our two extensions. We present our evaluation results in Section 5 and our conclusions in Section 6.

2 Baseline AZP Resolution Systems

An AZP resolution algorithm takes as input a set of AZPs produced by an AZP identification system. Below we first describe the AZP identifier we employ, followed by our two baseline AZP resolvers.

2.1 Anaphoric Zero Pronoun Identification

We employ two steps to *identify* AZPs. In the *extraction* step, we heuristically extract candidate ZPs. In the *classification* step, we train a classifier to distinguish AZPs from non-AZPs.

To implement the extraction step, we use Z&N's and K&Z's observation: ZPs can only occur before a VP node in a syntactic parse tree. However, according to K&Z, ZPs do not need to be extracted from every VP: if a VP node occurs in a coordinate structure or is modified by an adverbial node, then only its parent VP node needs to be considered. We extract ZPs from all VPs that satisfy the above constraints.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1360–1365,
Seattle, Washington, USA, 18-21 October 2013.

Syntactic features (13)	whether z is the first gap in an IP clause; whether z is the first gap in a subject-less IP clause, and if so, POS(w_1); whether POS(w_1) is NT; whether t_1 is a verb that appears in a NP or VP; whether P_l is a NP node; whether P_r is a VP node; the phrasal label of the parent of the node containing POS(t_1); whether V has a NP, VP or CP ancestor; whether C is a VP node; whether there is a VP node whose parent is an IP node in the path from t_1 to C.
Lexical features (13)	the words surrounding z and/or their POS tags, including w_1, w_{-1}, POS(w_1), POS(w_{-1})+POS(w_1), POS(w_1)+POS(w_2), POS(w_{-2})+POS(w_{-1}), POS(w_1)+POS(w_2)+POS(w_3), POS(w_{-1})+w_1, and w_{-1}+POS(w_1); whether w_1 is a transitive verb, an intransitive verb or a preposition; whether w_{-1} is a transitive verb without an object.
Other features (6)	whether z is the first gap in a sentence; whether z is in the headline of the text; the type of the clause in which z appears; the grammatical role of z; whether w_{-1} is a punctuation; whether w_{-1} is a comma.

Table 1: Features for AZP identification. z is a zero pronoun. V is the VP node following z. w_i is the ith word to the right of z (if i is positive) or the ith word to the left of z (if i is negative). C is lowest common ancestor of w_{-1} and w_1. P_l and P_r are the child nodes of C that are the ancestors of w_{-1} and w_1 respectively.

Features between a and z (4)	the sentence distance between a and z; the segment distance between a and z, where segments are separated by punctuations; whether a is the closest NP to z; whether a and z are siblings in the associated parse tree.
Features on a (12)	whether a has an ancestor NP, and if so, whether this NP is a descendent of a's lowest ancestor IP; whether a has an ancestor VP, and if so, whether this VP is a descendent of a's lowest ancestor IP; whether a has an ancestor CP; the grammatical role of a; the clause type in which a appears; whether a is an adverbial NP, a temporal NP, a pronoun or a named entity; whether a is in the headline of the text.
Features on z (10)	whether V has an ancestor NP, and if so, whether this NP node is a descendent of V's lowest ancestor IP; whether V has an ancestor VP, and if so, whether this VP is a descendent of V's lowest ancestor IP; whether V has an ancestor CP; the grammatical role of z; the type of the clause in which V appears; whether z is the first or last ZP of the sentence; whether z is in the headline of the text.

Table 2: Features for AZP resolution in the Zhao and Ng (2007) baseline system. z is a zero pronoun. a is a candidate antecedent of z. V is the VP node following z in the parse tree.

To implement the classification step, we train a classifier using SVM^{light} (Joachims, 1999) to distinguish AZPs from non-AZPs. We employ 32 features, 13 of which were proposed by Z&N and 19 of which were proposed by Yang and Xue (2010). A brief description of these features can be found in Table 1.

2.2 Two Baseline AZP Resolvers

The Zhao and Ng (2007) [Z&N] baseline. In our implementation of the Z&N baseline, we use SVM^{light} to train a mention-pair model for determining whether an AZP z and a candidate antecedent of z are coreferent. We consider all NPs preceding z that do not have the same head as its parent NP in the parse tree to be z's candidate antecedents. We use Soon et al.'s (2001) method to create training instances: we create a positive instance between an AZP, z, and its closest overt antecedent, and we create a negative instance between z and each of the intervening candidates. Each instance is represented by the 26 features employed by Z&N. A brief description of these features can be found in Table 2. During testing, we adopt the *closest-first* resolution strategy, resolving an AZP to the closest candidate antecedent that is classified as coreferent with it.[1]

The Kong and Zhou (2010) [K&Z] baseline. K&Z employ a tree kernel-based approach to AZP resolution. Like Z&N, K&Z (1) train a mention-pair model for determining whether an AZP z and a candidate antecedent of z are coreferent, (2) use Soon et al.'s method to create training instances, and (3) resolve an AZP to its closest coreferent candidate antecedent. Unlike Z&N, however, K&Z use the SVM$^{light-TK}$ learning algorithm (Moschitti,

[1] When resolving a *gold* AZP z, if none of the preceding candidate antecedents is classified as coreferent with it, we resolve it to the candidate that has the highest coreference likelihood with it. Here, we employ the signed distance from the SVM hyperplane to measure the coreference likelihood.

2006) to train their model, employing a parse subtree known as a dynamic expansion tree (Zhou et al., 2008) as a structured feature to represent an instance.

3 Extension 1: Novel Features

We propose three kinds of features to better capture the context of an AZP, as described below.

Antecedent compatibility. AZPs are omitted subjects that precede VP nodes in a sentence's parse tree. From the VP node, we can extract its head verb ($Pred_z$) and the head of its object NP (Obj), if any. Note that $Pred_z$ and Obj contain important contextual information for an AZP.

Next, observe that if a NP is coreferent with an AZP, it should be able to fill the AZP's gap and be compatible with the gap's context. Consider the following example:

E1: **他们**在试**那个服务**。那么啊就是 $*pro*$ 希望到九月我们的旅客来的时候。
(**They** are trying **that service**. That means $*pro*$ hope that our visitors can try it when they come in September.)

The head of the VP following $*pro*$ is 希望 (hope). There are two candidate antecedents, 他们 (They) and 那个服务 (that service). If we try using them to fill this AZP's gap, we know based on selectional preferences that 他们希望 (They hope) makes more sense than 那个服务希望 (that service hope). We supply the AZP resolver with the following information to help it make these decisions. First, we find the head word of each candidate antecedent, $Head_c$. Then we form two strings, $Head_c + Pred_z$ and $Head_c + Pred_z + Obj$ (if the object of the VP is present). Finally, we employ them as binary lexical features, setting their feature values to 1 if and only if they can be extracted from the instance under consideration. The training data can be used to determine which of these features are useful.[2]

Narrative event chains. A narrative event chain is a partially ordered set of events related by a common protagonist (Chambers and Jurafsky, 2008). For example, we can infer from the chain "borrow-s invest-s spend-s lend-s" that a person who borrows (presumably money) can invest it, spend it, or lend it to other people.[3] Consider the following example:

E2: **国家**给钱了，$*pro*$ 提供这部分的钱都是自己所里挣的。
(**The country** gives our department money, but all $*pro*$ provides is exactly what we worked for.)

In E2, $*pro*$ is coreferent with 国家 (The country), and the presence of the narrative event chain 给 − 提供 (gives−provides) suggests that the subjects of the two events are likely to be coreferent.

However, given the unavailability of induced or hand-crafted narrative chains in Chinese[4], we make the simplifying assumption that two verbs form a lexical chain if they are lexically identical.[5] We create two features to exploit narrative event chains for a candidate NP, c, if it serves as a subject or object. Specifically, let the verb governing c be $Pred_c$. The first feature, which encodes whether narrative chains are present, has three possible values: 0 if $Pred_c$ and $Pred_z$ are not the same; 1 if $Pred_c$ and $Pred_z$ are the same and c is a subject; and 2 if $Pred_c$ and $Pred_z$ are the same and c is an object. The second feature is a binary lexical feature, $Pred_c$+$Pred_z$+$Subject/Object$; its value is 1 if and only if $Pred_c$, $Pred_z$, and $Subject/Object$ can be found in the associated instance, where $Subject/Object$ denotes the grammatical role of c.

Final punctuation hint. We observe that the punctuation ($Punc$) at the end of a sentence where an AZP occurs also provides contextual information, especially in conversation documents. In conversations, if a sentence containing an AZP ends with a

[2] We tried to apply Kehler et al.'s (2004) and Yang et al.'s (2005) methods to learn Chinese selectional preferences from unlabeled data, but without success.

[3] "-s" denotes the fact that the protagonist serves as the grammatical subject in these events.

[4] We tried to construct narrative chains for Chinese using both learning-based and dictionary-based methods. Specifically, we induced narrative chains using Chambers and Jurafsky's (2008) method, but were not successful owing to the lack of an accurate Chinese coreference resolver. In addition, we constructed narrative chains using both lexically identical verbs and the synonyms obtained from a WordNet-like Chinese resource called Tongyicicilin, but they did not help improve resolution performance.

[5] Experiments on the training data show that if an AZP and a candidate antecedent are subjects of (different occurrences of) the same verb, then the probability that the candidate antecedent is coreferent with the AZP is 0.703. This result suggests that our assumption, though somewhat simplistic, is useful as far as AZP resolution is concerned.

A: **她**现在生活怎么样？
(A: How is **her** life now?)
B: $*pro_1*$ 对生活，就是很朴素，很简单。
(B: $*pro_1*$ attitude toward life is plain and simple.)
A: 嗯。
(A: Yes.)
A: $*pro_2*$ 是在北京还是在美国？
(A: $*pro_2*$ is living in Beijing or the USA?)
B: $*pro_3*$ 在美国。
(B: $*pro_3*$ is living in the USA.)

Figure 1: An illustrative example.

question mark, the mention this AZP refers to is less likely to be the speaker himself[6], as illustrated in the following example:

E3: 冬天 $*pro*$ 冷吗？
(Are $*pro*$ cold in the winter?)

Here, $*pro*$ refers to the person the speaker talks with. To capture this information, we create a binary lexical feature, $Head_c$+$Punc$, whose value is 1 if and only if $Head_c$ and $Punc$ appear in the instance under consideration.

4 Extension 2: Zero Pronoun Links

4.1 Motivation

Like an overt pronoun, a ZP whose closest overt antecedent is far away from it is harder to resolve than one that has a nearby overt antecedent. However, a corpus study of our training data reveals that only 55.2% of the AZPs appear in the same sentence as their closest overt antecedent, and 22.7% of the AZPs appear two or more sentences away from their closest overt antecedent.

Fortunately, we found that some of the difficult-to-resolve AZPs (i.e., AZPs whose closest overt antecedents are far away from them) are coreferential with nearby ZPs. Figure 1, which consists of a set of sentences from a conversation, illustrates this phenomenon. There are three AZPs (denoted by $*pro_i*$, where $1 \leq i \leq 3$), all of which refer to the overt pronoun 她 (She) in the first sentence. In this example, it is fairly easy to resolve $*pro_1*$ correctly,

[6]One may wonder whether we can similarly identify constraints on the antecedents of a ZP from clause conjunctions. Our preliminary analysis suggests that the answer is no.

	Training	Test
Documents	1,391	172
Sentences	36,487	6,083
Words	756,063	110,034
ZPs	23,065	3,658
AZPs	12,111	1,713

Table 3: Statistics on the training and test sets.

since its antecedent is the subject of previous sentence. However, $*pro_3*$ and its closest overt antecedent 她 (She) are four sentences apart. Together with the fact that there are many intervening candidate antecedents, it is not easy for a resolver to correctly resolve $*pro_3*$.

To facilitate the resolution of $*pro_3*$ and difficult-to-resolve AZPs in general, we propose the following idea. We allow an AZP resolver to (1) establish coreferent links between two consecutive ZPs (i.e., $*pro_1*-*pro_2*$ and $*pro_2*-*pro_3*$ in our example), which are presumably easy to establish because the two AZPs involved are close to each other; and then (2) treat them as bridges and infer that $*pro_3*$'s overt antecedent is 她 (She).

4.2 Modified Resolution Algorithm

We implement the aforementioned idea by modifying the AZP resolver as follows. When we resolve an AZP z during testing, we augment the set of candidate antecedents for z with the set of AZPs preceding z. Since we have only specified how to compute features for instances composed of an AZP and an overt candidate antecedent thus far (see Section 2.2), the question, then, is: how can we compute features for instances composed of two AZPs?

To answer this question, we first note that the AZPs in a test text are resolved in a left-to-right manner. Hence, by the time we resolve an AZP z, all the AZPs preceding z have been resolved. Hence, when we create a test instance i between z and one of the preceding AZPs (say y), we create i as if the gap y was filled with the smallest tree embedding the NP to which y was resolved.

By allowing coreference links between (presumably nearby) ZPs to be established, we can reason over the resulting coreference links, treating them as bridges that can help us find an overt antecedent that is far away from an AZP.

System Variation	Gold AZP Gold Parse Tree			System AZP Gold Parse Tree			System AZP System Parse Tree		
	R	P	F	R	P	F	R	P	F
K&Z Baseline System	38.0	38.0	38.0	17.7	22.4	19.8	10.6	13.6	11.9
Z&N Baseline System	41.5	41.5	41.5	22.4	24.4	23.3	12.7	14.2	13.4
Z&N Baseline + Contextual Features	46.2	46.2	46.2	25.2	27.5	26.3	14.4	16.1	15.2
Z&N Baseline + Zero Pronoun Links	42.7	42.7	42.7	22.5	24.6	23.5	13.2	14.8	13.9
Full System	**47.7**	**47.7**	**47.7**	**25.3**	**27.6**	**26.4**	**14.9**	**16.7**	**15.7**

Table 4: Resolution results on the test set.

5 Evaluation

5.1 Experimental Setup

Dataset. For evaluation, we employ the portion of the OntoNotes 4.0 corpus that was used in the official CoNLL-2012 shared task. The shared task dataset is composed of a training set, a development set, and a test set. Since only the training set and the development set are annotated with ZPs, we use the training set for classifier training and reserve the development set for testing purposes. Statistics on the datasets are shown in Table 3. In these datasets, a ZP is marked as $*pro*$. We consider a ZP anaphoric if it is coreferential with a preceding ZP or overt NP.

Evaluation measures. We express the results of both AZP identification and AZP resolution in terms of recall (R), precision (P) and F-score (F).

5.2 Results and Discussion

The three major columns of Table 4 show the results obtained in three settings, which differ in terms of whether gold/system AZPs and manually/automatically constructed parse trees are used to extract candidate antecedents and features.

In the first setting, the resolvers are provided with gold AZPs and gold parse trees. Results are shown in column 1. As we can see, the Z&N baseline significantly outperforms the K&Z baseline by 3.5% in F-score.[7] Adding the contextual features, the ZP links, and both extensions to Z&N increase its F-score significantly by 4.7%, 1.2% and 6.2%, respectively.

In the next two settings, the resolvers operate on the *system* AZPs provided by the AZP identification component. When gold parse trees are employed, the recall, precision and F-score of AZP identification are 50.6%, 55.1% and 52.8% respectively. Column 2 shows the results of the resolvers obtained when these automatically identified AZPs are used. As we can see, Z&N again significantly outperforms K&Z by 3.5% in F-score. Adding the contextual features, the ZP links, and both extensions to Z&N increase its F-score by 3.0%, 0.2% and 3.1%, respectively. The system with contextual features and the full system both yield results that are significantly better than those of the Z&N baseline. A closer examination of the results reveals why the ZP links are not effective in improving performance: when employing system AZPs, many erroneous ZP links were introduced to the system.

Column 3 shows the results of the resolvers when we employ system AZPs and the automatically generated parse trees provided by the CoNLL-2012 shared task organizers to compute candidate antecedents and features. Hence, these are *end-to-end* ZP resolution results. To our knowledge, these are the first reported results on end-to-end Chinese ZP resolution. Using automatic parse trees, the performance on AZP identification drops to 30.8% (R), 34.4% (P) and 32.5% (F). In this setting, Z&N still outperforms K&Z significantly, though by a smaller margin when compared to the previous settings. Incorporating the contextual features, the ZP links, and both extensions increase the F-score by 1.8%, 0.5% and 2.3%, respectively. The system with contextual features and the full system both yield results that are significantly better than those of the Z&N baseline.

6 Conclusions

We proposed two extensions to a state-of-the-art Chinese AZP resolver proposed by Zhao and Ng (2007). Experimental results on the OntoNotes dataset showed that the resulting resolver significantly improved both Zhao and Ng's and Kong and Zhou's (2010) resolvers, regardless of whether gold or system AZPs and syntactic parse trees are used.

[7] All significance tests are paired t-tests, with $p < 0.05$.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of the paper. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 787--797.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44--56. MIT Press.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. Competitive self-trained pronoun interpretation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 33--36.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for Chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882--891.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language processing. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113--120.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521--544.

Yaqin Yang and Nianwen Xue. 2010. Chasing the ghost: recovering empty categories in the Chinese Treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1382--1390.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 165--172.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*, pages 541--550.

GuoDong Zhou, Fang Kong, and Qiaoming Zhu. 2008. Context-sensitive convolution tree kernel for pronoun resolution. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 25--31.

Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction

Jason Weston
Google
111 8th avenue
New York, NY, USA
jweston@google.com

Antoine Bordes
Heudiasyc
UT de Compiègne
& CNRS
Compiègne, France
bordesan@utc.fr

Oksana Yakhnenko
Google
111 8th avenue
New York, NY, USA
oksana@google.com

Nicolas Usunier
Heudiasyc
UT de Compiègne
& CNRS
Compiègne, France
nusunier@utc.fr

Abstract

This paper proposes a novel approach for relation extraction from free text which is trained to jointly use information from the text and from existing knowledge. Our model is based on scoring functions that operate by learning low-dimensional embeddings of words, entities and relationships from a knowledge base. We empirically show on New York Times articles aligned with Freebase relations that our approach is able to efficiently use the extra information provided by a large subset of Freebase data (4M entities, 23k relationships) to improve over methods that rely on text features alone.

1 Introduction

Information extraction (IE) aims at generating structured data from free text in order to populate Knowledge Bases (KBs). Hence, one is given an incomplete KB composed of a set of triples of the form (h, r, t); h is the left-hand side entity (or *head*), t the right-hand side entity (or *tail*) and r the relationship linking them. An example from the Freebase KB[1] is (/m/2d3rf ,<director-of>, /m/3/324), where /m/2d3rf refers to the director "Alfred Hitchcock" and /m/3/324 to the movie "The Birds".

This paper focuses on the problem of learning to perform relation extraction (RE) under weak supervision from a KB. RE is sub-task of IE that considers that entities have already been detected by a different process, such as a named-entity recognizer. RE then aims at assigning to a relation mention m (i.e. a sequence of text which states that some relation is true) the corresponding relationship from the KB, given a pair of extracted entities (h, t) as context. For example, given the triple (/m/2d3rf ,"wrote and directed", /m/3/324), a system should predict <director-of>. The task is said to be weakly supervised because for each pair of entities (h, t) detected in the text, all relation mentions m associated with them are labeled with all the relationships connecting h and t in the KB, whether they are actually expressed by m or not.

Our key contribution is a novel model that employs not only weakly labeled text mention data, as most approaches do, but also leverages triples from the known KB. The model thus learns the plausibility of new (h, r, t) triples by generalizing from the KB, even though this triple is not present. A ranking-based embedding framework is used to train our model. Thereby, relation mentions, entities and relationships are all embedded into a common low-dimensional vector space, where scores are computed. We show that our method can successfully take into account information from a large-scale KB (Freebase: 4M entities, 23k relationships) to improve over systems that are only using text features.

This paper is organized as follows: Section 2 presents related work, Section 3 introduces our model and its main influences, and experimental results are displayed in Section 4.

2 Previous Work

Learning under weak supervision is common in natural language processing, especially for tasks where the annotation costs are significant such as in se-

[1] www.freebase.com

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1366–1371,
Seattle, Washington, USA, 18-21 October 2013.

mantic parsing (Kate and Mooney, 2007; Liang et al., 2009; Bordes et al., 2010; Matuszek et al., 2012). This is also naturally used in IE, since it allows to train large-scale systems without requiring to label numerous texts. The idea was introduced by (Craven et al., 1999), which matched the Yeast Protein Database with PubMed abstracts. It was also used to train open extractors based on Wikipedia infoboxes and corresponding sentences (Wu and Weld, 2007; Wu and Weld, 2010). Large-scale open IE projects (e.g. (Banko et al., 2007)) also rely on weak supervision, since they learn models from a seed KB in order to extend it.

Weak supervision is also a popular option for RE: Mintz et al. (2009) used Freebase to train weakly supervised relational extractors on Wikipedia, an approach generalized by the multi-instance learning frameworks (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). All these works only use textual information to perform extraction.

Lao et al. (2012) proposed the first work aiming to perform RE employing both KB data and text, using a rule-based random walk method. Recently, Riedel et al. (2013) proposed another joint approach based on collaborative filtering for learning entity embeddings. This approach connects text with Freebase by learning shared embeddings of entities through weak supervision, in contrast to our method where no joint learning is performed. We do not compare to these two approaches since they use two different evaluation protocols that greatly differ from those used in all aforementioned previous works. Nevertheless, our method is easier to integrate into existing systems than those, since KB data is used via the addition of a scoring term, which is trained separately beforehand (with no shared embeddings). Besides, we demonstrate in our experimental section that our system can handle a large number of relationships, significantly larger than that presented in (Lao et al., 2012; Riedel et al., 2013).

3 Embedding-based Framework

Our work concerns energy-based methods, which learn low-dimensional vector representations (*embeddings*) of atomic symbols (words, entities, relationships, etc.). In this framework, we learn two models: one for predicting relationships given relation mentions and another one to encode the interactions among entities and relationships from the KB. The joint action of both models in prediction allows us to use the connection between the KB and text to perform relation extraction. One could also share parameters between models (via shared embeddings), but this is not implemented in this work. This approach is inspired by previous work designed to connect words and Wordnet (Bordes et al., 2012).

Both submodels end up learning vector embeddings of symbols, either for entities or relationships in the KB, or for each word/feature of the vocabulary (denoted $\mathcal{V}$). The set of entities and relationships in the KB are denoted by $\mathcal{E}$ and $\mathcal{R}$, and n_v, n_e and n_r denote the size of $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{R}$ respectively. Given a triple (h, r, t) the embeddings of the entities and the relationship (vectors in $\mathbb{R}^k$) are denoted with the same letter, in boldface characters (i.e. $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$).

3.1 Connecting Text and Relationships

The first part of the framework concerns the learning of a function $S_{m2r}(m, r)$, based on embeddings, that is designed to score the similarity of a relation mention m and a relationship r.

Our scoring approach is inspired by previous work for connecting word labels and images (Weston et al., 2010), which we adapted, replacing images by mentions and word labels by relationships. Intuitively, it consists of first projecting words and features into the embedding space and then computing a similarity measure (the dot product in this paper) between this projection and a relationship embedding. The scoring function is then:

$$S_{m2r}(m, r) = \mathbf{f}(m)^\top \mathbf{r}$$

with $\mathbf{f}$ a function mapping words and features into $\mathbb{R}^k$, $\mathbf{f}(m) = \mathbf{W}^\top \Phi(m)$. $\mathbf{W}$ is the matrix of $\mathbb{R}^{n_v \times k}$ containing all word embeddings $\mathbf{w}$, $\Phi(m)$ is the (sparse) binary representation of m ($\in \mathbb{R}^{n_v}$) indicating absence or presence of words/features, and $\mathbf{r} \in \mathbb{R}^k$ is the embedding of the relationship r.

This approach can be easily applied at test time to score (mention, relationship) pairs. Since this type of learning problem is weakly supervised, Bordes et al. (2010) showed that a convenient way to train it is by using a ranking loss. Hence, given a data set $\mathcal{D} = \{(m_i, r_i), i = 1, \ldots, |\mathcal{D}|\}$ consisting of (mention, relationship) training pairs, one could learn the

embeddings using constraints of the form:

$$\forall i,\ \forall r' \neq r_i,\ \ \mathbf{f}(m_i)^\top \mathbf{r}_i > 1 + \mathbf{f}(m_i)^\top \mathbf{r}' \ , \quad (1)$$

where 1 is the margin. That is, we want the relation that (weakly) labels a given mention to be scored higher than other relation by a margin of 1. Then, given any mention m one can predict the corresponding relationship $\hat{r}(m)$ with:

$$\hat{r}(m) = \arg\max_{r' \in \mathcal{R}} S_{m2r}(m, r') = \arg\max_{r' \in \mathcal{R}} \left(\mathbf{f}(m)^\top \mathbf{r}'\right).$$

Learning $S_{m2r}(\cdot)$ under constraints (1) is well suited when one is interested in building a per-mention prediction system. However, performance metrics of relation extraction are sometimes measured using precision recall curves aggregated for all mentions concerning the same pair of entities, as in (Riedel et al., 2010). In that case the scores across predictions for different mentions need to be calibrated so that the most confident ones have the higher scores. This can be better encoded with constraints of the following form:

$$\forall i, j,\ \ \forall r' \neq r_i, r_j,\ \ \mathbf{f}(m_i)^\top \mathbf{r}_i > 1 + \mathbf{f}(m_j)^\top \mathbf{r}' \ .$$

In this setup, scores of pairs observed in the training set should be larger than that of any other prediction across all mentions. In practice, we use "soft" ranking constraints (optimizing the hinge loss), i.e. we minimize:

$$\forall i, j,\ \forall r' \neq r_i, r_j,\ \ \max(0, 1 - \mathbf{f}(m_i)^\top \mathbf{r}_i + \mathbf{f}(m_j)^\top \mathbf{r}').$$

Finally, we also enforce a (hard) constraint on the norms of the columns of $\mathbf{W}$ and $\mathbf{r}$, i.e. $\forall_i, ||\mathbf{W}_i||_2 \leq 1$ and $\forall_j, ||\mathbf{r}_j||_2 \leq 1$. Training is carried out by Stochastic Gradient Descent (SGD), updating $\mathbf{W}$ and $\mathbf{r}$ at each step, following (Weston et al., 2010; Bordes et al., 2013). That is, at the start of training the parameters to be learnt (the $n_v \times k$ word/feature embeddings in W and the $n_r \times k$ relation embeddings r) are initialized to random weights. We initialize each k-dimensional embedding vector randomly with mean 0, standard deviation $\frac{1}{k}$. Then, we iterate the following steps to train them:

1. Select at random a positive training pair (m_i, r_i).
2. Select at random a secondary training pair (m_j, r_j), used to calibrate the scores.
3. Select at random a negative relation r' such that $r' \neq r_i$ and $r' \neq r_j$.
4. Make a stochastic gradient step to minimize $\max(0, 1 - \mathbf{f}(m_i)^\top \mathbf{r}_i + \mathbf{f}(m_j)^\top \mathbf{r}')$.
5. Enforce the constraint that each embedding vector is normalized, i.e. if $||\mathbf{W}_i||_2 > 1$ then $\mathbf{W}_i \leftarrow \mathbf{W}_i / ||\mathbf{W}_i||_2$.

3.2 Encoding Structured Data of KBs

Using only weakly labeled text mentions for training ignores much of the prior knowledge we can leverage from a large KB such as Freebase. In order to connect this relational data with our model, we propose to encode its information into entity and relationship embeddings. This allows us to build a model which can score the plausibility of new entity relationship triples which are missing from Freebase. Several models have been recently developed for that purpose (e.g. in (Nickel et al., 2011; Bordes et al., 2011; Bordes et al., 2012)): we chose in this work to follow the approach of (Bordes et al., 2013), which is simple, flexible and has shown very promising results on Freebase data.

Given a training set $\mathcal{S} = \{(h_i, r_i, t_i), i = 1, \ldots, |\mathcal{S}|\}$ of relations extracted from the KB, this model learns vector embeddings of the entities and of the relationships using the idea that the functional relation induced by the r-labeled arcs of the KB should correspond to a translation of the embeddings. That is, given a k-dimensional embedding of the left-hand side (head) entity, adding the k-dimensional embedding of a given relation should yield the point (or close to the point) of the k-dimensional embedding of the right-hand side (tail) entity. Hence, this method enforces that $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when (h, r, t) holds, while $\mathbf{h} + \mathbf{r}$ should be far away from $\mathbf{t}$ otherwise. The model thus gives the following score for the plausibility of a relation:

$$S_{kb}(h, r, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||_2^2 .$$

A ranking loss is also used for training S_{kb}. The ranking objective is designed to assign higher scores

to existing relations versus any other possibility:

$$\forall i, \forall h' \neq h_i, \quad S_{kb}(h_i, r_i, t_i) \geq 1 + S_{kb}(h', r_i, t_i),$$
$$\forall i, \forall r' \neq r_i, \quad S_{kb}(h_i, r_i, t_i) \geq 1 + S_{kb}(h_i, r', t_i),$$
$$\forall i, \forall t' \neq t_i, \quad S_{kb}(h_i, r_i, t_i) \geq 1 + S_{kb}(h_i, r_i, t').$$

That is, for each known triple (h, r, t), if we replaced the (i) head, (ii) relation or (iii) tail with some other possibility, the modified triple should have a lower score (i.e. be less plausible) than the original triple. The three sets of constraints defined above encode the three types of modification. As in Section 3.1 we use soft constraints via the hinge loss, enforce constraints on the norm of embeddings, i.e. $\forall_{h,r,t}, ||h||_2 \leq 1, ||r||_2 \leq 1, ||t||_2 \leq 1$, and training is performed using SGD, as in (Bordes et al., 2013).

At test time, one may again need to calibrate the scores S_{kb} across entity pairs. We propose a simple approach: we convert the scores by ranking all relationships $\mathcal{R}$ by S_{kb} and instead output:

$$\tilde{S}_{kb}(h, r, t) = \Phi\Big(\sum_{r' \neq r} \delta(S_{kb}(h, r, t) > S_{kb}(h, r', t))\Big),$$

i.e. a function of the rank of r. We chose the simplified model $\Phi(x) = 1$ if $x < \tau$ and 0 otherwise; $\delta(\cdot)$ is the Kronecker function.

3.3 Implementation for Relation Extraction

Our framework can be used for relation extraction in the following way. First, for each pair of entities (h, t) that appear in the test set, all the corresponding mentions $\mathcal{M}_{h,t}$ in the test set are collected and a prediction is performed with:

$$\hat{r}_{h,t} = \operatorname*{argmax}_{r \in \mathcal{R}} \sum_{m \in \mathcal{M}_{h,t}} S_{m2r}(m, r)\,.$$

The predicted relationship can either be a valid relationship or NA – a marker that means that there is no relation between h and t (NA is added to $\mathcal{R}$ during training and is treated like other relationships). If $\hat{r}_{h,t}$ is a relationship, a composite score is defined:

$$S_{m2r+kb}(h, \hat{r}_{h,t}, t) = \sum_{m \in \mathcal{M}_{h,t}} S_{m2r}(m, \hat{r}_{h,t}) + \tilde{S}_{kb}(h, \hat{r}_{h,t}, t)$$

That is, only the top scoring non-NA predictions are re-scored. Hence, our final composite model favors predictions that agree with both the mentions and the KB. If $\hat{r}_{h,t}$ is NA, the score is unchanged.

4 Experiments

We use the training and test data, evaluation framework and baselines from (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012).

NYT+FB This dataset, developed by (Riedel et al., 2010), aligns Freebase relations with the New York Times corpus. Entities were found using the Stanford named entity tagger (Finkel et al., 2005), and were matched to their name in Freebase. For each mention, sentence level features are extracted which include part of speech, named entity and dependency tree path properties. Unlike some of the previous methods, we do not use features that aggregate properties across multiple mentions. We kept the 100,000 most frequent features.There are 52 possible relationships and 121,034 training mentions of which most are labeled as no relation (labeled "NA") – there are 4700 Freebase relations mentioned in the training set, and 1950 in the test set.

Freebase Freebase is a large-scale KB that has around 80M entities, 23k relationships and 1.2B relations. We used a subset restricted to the top 4M entities for scalability reasons – where top is defined as the ones with the largest number of relations to other entities. We used all the 23k possible relationships in Freebase. To make a realistic setting, we did not choose the entity set using the NYT+FB data set, so it may not overlap completely. For that reason, we needed to keep the set rather large. Keeping the top 4M entities gives an overlap of 80% with the entities in the NYT+FB test set. Most importantly, we then removed all the entity pairs present in the NYT+FB test set from Freebase, i.e. all relations they are involved in independent of the relationship. This ensures that we cannot just memorize the true relations for an entity pair – we have to learn to generalize from other entities and relations.

As the NYT+FB dataset was built on an earlier version of Freebase we also had to translate the deprecated relationships into their new variants (e.g. "/p/business/company/place_founded " → "/organization/organization/place_founded") to make the two datasets link (then, the 52 relationships in NYT+FB are now a subset of the 23k from Freebase). We then trained the S_{kb} model on the remaining triples.

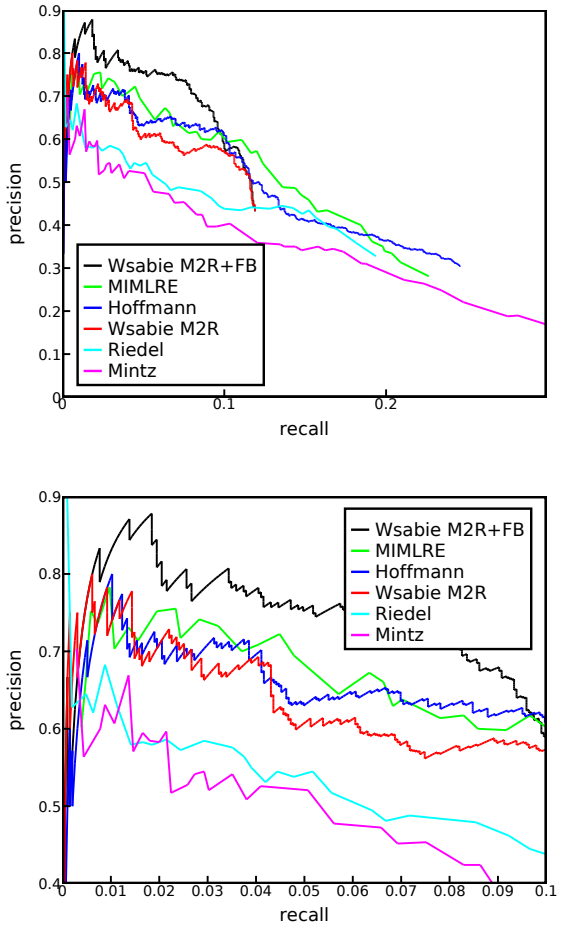


Figure 1: Top: Aggregate extraction precision/recall curves for a variety of methods. Bottom: the same plot zoomed to the recall [0-0.1] region. Wsabie$_{M2R}$ is our method trained only on mentions, Wsabie$_{M2R+FB}$ uses Freebase annotations as well.

Modeling Following (Bordes et al., 2013) we set the embedding dimension k to 50. The learning rate for SGD was selected using a validation set: we obtained 0.001 for S_{m2r}, and 0.1 for S_{kb}. For the calibration of $\hat{S}_{kb}$, $\tau = 10$ (note, here we are ranking all 23k Freebase relationships). Training S_{m2r} took 5 minutes, whilst training S_{kb} took 2 days due to the large scale of the data set.

Results Figure 1 displays the aggregate precision / recall curves of our approach WSABIE$_{M2R+FB}$ which uses the combination of $S_{m2r} + S_{kb}$, as well as WSABIE$_{M2R}$, which only uses S_{m2r}, and existing state-of-the-art approaches: HOFFMANN (Hoffmann et al., 2011)[2], MIMLRE (Surdeanu et al., 2012). RIEDEL (Riedel et al., 2010) and MINTZ (Mintz et al., 2009).

WSABIE$_{M2R}$ is comparable to, but slightly worse than, the MIMLRE and HOFFMANN methods, possibly due to its simplified assumptions (e.g. predicting a single relationship per entity pair). However, the addition of extra knowledge from other Freebase entities in WSABIE$_{M2R+FB}$ provides superior performance to all other methods, by a wide margin, at least between 0 and 0.1 recall (see bottom plot).

Performance of WSABIE$_{M2R}$ and WSABIE$_{M2R+FB}$ for recall > 0.1 degrades rapidly, faster than that of other methods. This is also caused by the simplifications of WSABIE$_{M2R}$ that prevent it from reaching high precision when the recall is greater than 0.1. We recall that Freebase data is not used to detect relationships i.e. to discriminate between NA and the rest, but only to select the best relationship in case of detection. That is WSABIE$_{M2R+FB}$ only improves precision, not recall, so both versions of Wsabie are similar w.r.t. recall. This could be improved by borrowing ideas from HOFFMANN (Hoffmann et al., 2011) or MIMLRE (Surdeanu et al., 2012) for dealing with the multi-label case. Our approach, which uses Freebase to increase precision, is general and could improve any other method.

5 Conclusion

In this paper we described a framework for leveraging large scale knowledge bases to improve relation extraction by training not only on (mention, relationship) pairs but using all other KB triples as well. We empirically showed that it allows to significantly improve precision on extracted relations. Our modeling approach is general and should apply to other settings, e.g. for the task of entity linking.

Acknowledgments

This work was carried out in the framework of the Labex MS2T (ANR-11-IDEX-0004-02), and funded by the French National Agency for Research (EVEREST-12-JS02-005-01).

[2]There is an error in the plot from (Hoffmann et al., 2011), which we have corrected. The authors acknowledged this issue.

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 103–110.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proc. of the 25th Conf. on Artif. Intel. (AAAI)*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *Proc. of the 15th Intern. Conf. on Artif. Intel. and Stat.*, volume 22, pages 127–135. JMLR.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS 26)*.

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 541–550.

Rohit J Kate and Raymond J Mooney. 2007. Learning language semantics from ambiguous supervision. In *AAAI*, volume 7, pages 895–900.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 809–816.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, pages 74–84.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Simple Customization of Recursive Neural Networks for Semantic Relation Classification

Kazuma Hashimoto†, Makoto Miwa††, Yoshimasa Tsuruoka†, and Takashi Chikayama†
†The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{hassy, tsuruoka, chikayama}@logos.t.u-tokyo.ac.jp
††The University of Manchester, 131 Princess Street, Manchester, M1 7DN, UK
makoto.miwa@manchester.ac.uk

Abstract

In this paper, we present a recursive neural network (RNN) model that works on a syntactic tree. Our model differs from previous RNN models in that the model allows for an explicit weighting of important phrases for the target task. We also propose to average parameters in training. Our experimental results on semantic relation classification show that both phrase categories and task-specific weighting significantly improve the prediction accuracy of the model. We also show that averaging the model parameters is effective in stabilizing the learning and improves generalization capacity. The proposed model marks scores competitive with state-of-the-art RNN-based models.

1 Introduction

Recursive Neural Network (RNN) models are promising deep learning models which have been applied to a variety of natural language processing (NLP) tasks, such as sentiment classification, compound similarity, relation classification and syntactic parsing (Hermann and Blunsom, 2013; Socher et al., 2012; Socher et al., 2013). RNN models can represent phrases of arbitrary length in a vector space of a fixed dimension. Most of them use minimal syntactic information (Socher et al., 2012).

Recently, Hermann and Blunsom (2013) proposed a method for leveraging syntactic information, namely CCG combinatory operators, to guide composition of phrases in RNN models. While their models were successfully applied to binary sentiment classification and compound similarity tasks, there are questions yet to be answered, e.g., whether such enhancement is beneficial in other NLP tasks as well, and whether a similar improvement can be achieved by using syntactic information of more commonly available types such as phrase categories and syntactic heads.

In this paper, we present a supervised RNN model for a semantic relation classification task. Our model is different from existing RNN models in that important phrases can be explicitly weighted for the task. Syntactic information used in our model includes part-of-speech (POS) tags, phrase categories and syntactic heads. POS tags are used to assign vector representations to word-POS pairs. Phrase categories are used to determine which weight matrices are chosen to combine phrases. Syntactic heads are used to determine which phrase is weighted during combining phrases. To incorporate task-specific information, phrases on the path between entity pairs are further weighted.

The second contribution of our work is the introduction of parameter averaging into RNN models. In our preliminary experiments, we observed that the prediction performance of the model often fluctuates significantly between training iterations. This fluctuation not only leads to unstable performance of the resulting models, but also makes it difficult to fine-tune the hyperparameters of the model. Inspired by Swersky et al. (2010), we propose to average the model parameters in the course of training. A recent technique for deep learning models of similar vein is *dropout* (Hinton et al., 2012), but averaging is simpler to implement.

Our experimental results show that our model per-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1372–1376,
Seattle, Washington, USA, 18-21 October 2013.

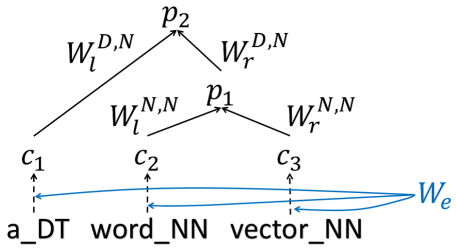


Figure 1: A recursive representations of a phrase "a word vector" with POS tags of the words (DT, NN and NN respectively). For example, the two word-POS pairs "word_NN" and "vector_NN" with a syntactic category N are combined to represent the phrase "word vector".

forms better than standard RNN models. By averaging the model parameters, our model achieves performance competitive with the MV-RNN model in Socher et al. (2012), without using computationally expensive word-dependent matrices.

2 An Averaged RNN Model with Syntax

Our model is a supervised RNN that works on a binary syntactic tree. As our first step to leverage information available in the tree, we distinguish words with the same spelling but POS tags in the vector space. Our model also uses different weight matrices dependent on the phrase categories of child nodes (phrases or words) in combining phrases. Our model further weights those nodes that appear to be important.

Compositional functions of our model follow those of the SU-RNN model in Socher et al. (2013).

2.1 Word-POS Vector Representations

Our model simply assigns vector representations to word-POS pairs. For example, a word "caused" can be represented in two ways: "caused_VBD" and "caused_VBN". The vectors are represented as column vectors in a matrix $\boldsymbol{W}_e \in \mathbb{R}^{d\times|\mathbb{V}|}$, where d is the dimension of a vector and $\mathbb{V}$ is a set of all word-POS pairs we consider.

2.2 Compositional Functions with Syntax

In construction of parse trees, we associate each of the tree node with its d-dimensional vector representation computed from vector representations of its subtrees. For leaf nodes, we look up word-POS vector representations in $\mathbb{V}$. Figure 1 shows an example of such recursive representations. A parent vector $\boldsymbol{p} \in \mathbb{R}^{d\times 1}$ is computed from its direct child vectors $\boldsymbol{c}_l$ and $\boldsymbol{c}_r \in \mathbb{R}^{d\times 1}$:

$$\boldsymbol{p} = \tanh(\alpha_l \boldsymbol{W}_l^{T_{c_l},T_{c_r}} \boldsymbol{c}_l + \alpha_r \boldsymbol{W}_r^{T_{c_l},T_{c_r}} \boldsymbol{c}_r + \boldsymbol{b}^{T_{c_l},T_{c_r}}),$$

where $\boldsymbol{W}_l^{T_{c_l},T_{c_r}}$ and $\boldsymbol{W}_r^{T_{c_l},T_{c_r}} \in \mathbb{R}^{d\times d}$ are weight matrices that depend on the phrase categories of $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$. Here, $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$ have phrase categories T_{c_l} and T_{c_r} respectively (such as *N*, *V*, etc.). $\boldsymbol{b}^{T_{c_l},T_{c_r}} \in \mathbb{R}^{d\times 1}$ is a bias vector. To incorporate the importance of phrases into the model, two subtrees of a node may have different weights $\alpha_l \in [0,1]$ and $\alpha_r (= 1-\alpha_l)$, taking phrase importance into account. The value of α_l is manually specified and automatically applied to all nodes based on prior knowledge about the task. In this way, we can compute vector representations for phrases of arbitrary length. We denote a set of such matrices as $\boldsymbol{W}_{lr}$ and bias vectors as $\boldsymbol{b}$.

2.3 Objective Function and Learning

As with other RNN models, we add on the top of a node $\boldsymbol{x}$ a $\mathrm{softmax}$ classifier. The classifier is used to predict a K-class distribution $\boldsymbol{d}(\boldsymbol{x}) \in \mathbb{R}^{K\times 1}$ over a specific task to train our model:

$$\boldsymbol{d}(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{W}^{label}\boldsymbol{x} + \boldsymbol{b}^{label}), \qquad (1)$$

where $\boldsymbol{W}^{label} \in \mathbb{R}^{K\times d}$ is a weight matrix and $\boldsymbol{b}^{label} \in \mathbb{R}^{K\times 1}$ is a bias vector. We denote $\boldsymbol{t}(\boldsymbol{x}) \in \mathbb{R}^{K\times 1}$ as the target distribution vector at node $\boldsymbol{x}$. $\boldsymbol{t}(\boldsymbol{x})$ has a 0-1 encoding: the entry at the correct label of $\boldsymbol{t}(\boldsymbol{x})$ is 1, and the remaining entries are 0. We then compute the cross entropy error between $\boldsymbol{d}(\boldsymbol{x})$ and $\boldsymbol{t}(\boldsymbol{x})$:

$$E(\boldsymbol{x}) = -\sum_{k=1}^{K} \boldsymbol{t}_k(\boldsymbol{x}) \log \boldsymbol{d}_k(\boldsymbol{x}),$$

and define an objective function as the sum of $E(\boldsymbol{x})$ over all training data:

$$J(\theta) = \sum_{\boldsymbol{x}} E(\boldsymbol{x}) + \frac{\lambda}{2}\|\theta\|^2,$$

where $\theta = (\boldsymbol{W}_e, \boldsymbol{W}_{lr}, \boldsymbol{b}, \boldsymbol{W}^{label}, \boldsymbol{b}^{label})$ is the set of our model parameters that should be learned. λ is a vector of regularization parameters.

To compute $\boldsymbol{d}(\boldsymbol{x})$, we can directly leverage any other nodes' feature vectors in the same tree. We denote such additional feature vectors as $\boldsymbol{x}'_i \in \mathbb{R}^{d \times 1}$, and extend Eq. (1):

$$\boldsymbol{d}(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{W}^{label}\boldsymbol{x} + \sum_i \boldsymbol{W}_i^{add}\boldsymbol{x}'_i + \boldsymbol{b}^{label}),$$

where $\boldsymbol{W}_i^{add} \in \mathbb{R}^{K \times d}$ are weight matrices for additional features. We denote these matrices $\boldsymbol{W}_i^{add}$ as $\boldsymbol{W}^{add}$. We also add $\boldsymbol{W}^{add}$ to θ:

$$\theta = (\boldsymbol{W}_e, \boldsymbol{W}_{lr}, \boldsymbol{b}, \boldsymbol{W}^{label}, \boldsymbol{W}^{add}, \boldsymbol{b}^{label}).$$

The gradient of $J(\theta)$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{\boldsymbol{x}} \frac{\partial E(\boldsymbol{x})}{\partial \theta} + \lambda\theta$$

is efficiently computed via backpropagation through structure (Goller and Küchler, 1996). To minimize $J(\theta)$, we use batch L-BFGS[1] (Hermann and Blunsom, 2013; Socher et al., 2012).

2.4 Averaging

We use averaged model parameters

$$\overline{\theta} = \frac{1}{T+1}\sum_{t=0}^{T}\theta_t$$

at test time, where θ_t is the vector of model parameters after t iterations of the L-BFGS optimization. Our preliminary experimental results suggest that averaging θ except $\boldsymbol{W}_e$ works well.

3 Experimental Settings

We used the Enju parser (Miyao and Tsujii, 2008) for syntactic parsing. We used 13 phrase categories given by Enju.

3.1 Task: Semantic Relation Classification

We evaluated our model on a semantic relation classification task: SemEval 2010 Task 8 (Hendrickx et al., 2010). Following Socher et al. (2012), we regarded the task as a 19-class classification problem. There are 8,000 samples for training, and 2,717 for test. For the validation set, we randomly sampled 2,182 samples from the training data.

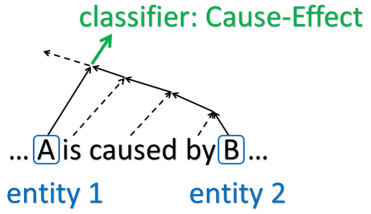


Figure 2: Classifying the relation between two entities.

To predict a class label, we first find the minimal phrase that covers the target entities and then use the vector representation of the phrase (Figure 2).

As explained in Section 2.3, we can directly connect features on any other nodes to the softmax classifier. In this work, we used three such internal features: two vector representations of target entities and one averaged vector representation of words between the entities[2].

3.2 Weights on Phrases

We tuned the weight α_l (or α_r) introduced in Section 2.2 for this particular task. There are two factors: syntactic heads and syntactic path between target entities. Our model puts a weight $\beta \in [0.5, 1]$ on head phrases, and $1-\beta$ on the others. For relation classification tasks, syntactic paths between target entities are important (Zhang et al., 2006), so our model also puts another weight $\gamma \in [0.5, 1]$ on phrases on the path, and $1-\gamma$ on the others. When both child nodes are on the path or neither of them on the path, we set $\gamma = 0.5$. The two weight factors are summed up and divided by 2 to be the final weights α_l and α_r to combine the phrases. For example, we set $\alpha_l = \frac{(1-\beta)+\gamma}{2}$ and $\alpha_r = \frac{\beta+(1-\gamma)}{2}$ when the right child node is the head and the left child node is on the path.

3.3 Initialization of Model Parameters and Tuning of Hyperparameters

We initialized $\boldsymbol{W}_e$ with 50-dimensional word vectors[3] trained with the model of Collobert et

[1] We used libLBFGS provided at `http://www.chokkan.org/software/liblbfgs/`.

[2] Socher et al. (2012) used richer features including words around entity pairs in their implementation.

[3] The word vectors are provided at `http://ronan.collobert.com/senna/`. We used the vectors without any modifications such as normalization.

Method	F1 (%)
Our model	79.4
RNN	74.8
MV-RNN	79.1
RNN w/ POS, WordNet, NER	77.6
MV-RNN w/ POS, WordNet, NER	82.4
SVM w/ bag-of-words	73.1
SVM w/ lexical and semantic features	82.2

Table 1: Comparison of our model with other methods on SemEval 2010 task 8.

Method	F1 (%)
Our model	79.4
Our model w/o phrase categories (PC)	77.7
Our model w/o head weights (HW)	78.8
Our model w/o path weights (PW)	78.7
Our model w/o averaging (AVE)	76.9
Our model w/o PC, HW, PW, AVE	74.1

Table 2: Contributions of syntactic and task-specific information and averaging.

al. (2011), and $\boldsymbol{W}_{lr}$ with $\frac{I}{2} + \varepsilon$, where $I \in \mathbb{R}^{d \times d}$ is an identity matrix. Here, ε is zero-mean gaussian random variable with a variance of 0.01. The initialization of $\boldsymbol{W}_{lr}$ is the same as that of Socher et al. (2013). The remaining model parameters were initialized with 0.

We tuned hyperparameters in our model using the validation set for each experimental setting. The hyperparameters include the regularization parameters for $\boldsymbol{W}_e$, $\boldsymbol{W}_{lr}$, $\boldsymbol{W}^{label}$ and $\boldsymbol{W}^{add}$, and the weights β and γ. For example, the best performance for our model with all the proposed methods was obtained with the values: 10^{-6}, 10^{-4}, 10^{-3}, 10^{-3}, 0.7 and 0.9 respectively.

4 Results and Discussion

Table 1 shows the performance of our model and that of previously reported systems on the test set. The performance of an SVM system with bag-of-words features was reported in Rink and Harabagiu (2010), and the performance of the RNN and MV-RNN models was reported in Socher et al. (2012). Our model achieves an F1 score of 79.4% and outperforms the RNN model (74.8% F1) as well as the simple SVM-based system (73.1% F1). More notably, the score of our model is competitive with that of the MV-RNN model (79.1% F1), which is computationally much more expensive. Readers are referred to Hermann and Blunsom (2013) for the discussion about the computational complexity of the MV-RNN model. We improved the performance of RNN models on the task without much increasing the complexity. This is a significant practical advantage of our model, although its expressive power is not the same as that of the MV-RNN model.

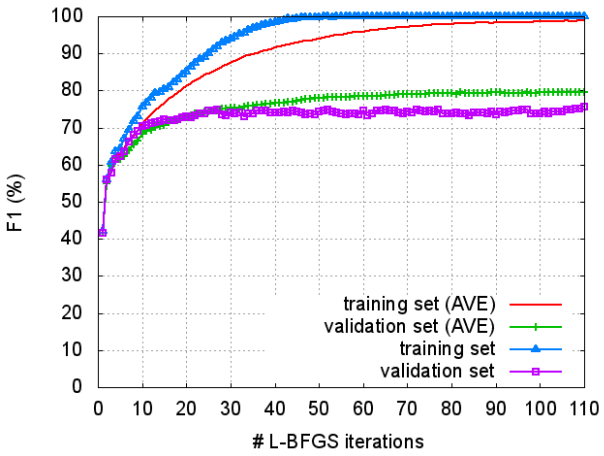


Figure 3: F1 vs Training iterations.

Our model outperforms the RNN model with one lexical and two semantic external features: POS tags, WordNet hypernyms and named entity tags (NER) of target word pairs (external features). The MV-RNN model with external features shows better performance than our model. An SVM with rich lexical and semantic features (Rink and Harabagiu, 2010) also outperforms ours. Note, however, that this is not a fair comparison because those models use rich external resources such as WordNet and named entity tags.

4.1 Contributions of Proposed Methods

We conducted additional experiments to quantify the contributions of phrase categories, heads, paths and averaging to our classification score. As shown in Table 2, our model without phrase categories, heads or paths still outperforms the RNN model with external features. On the other hand, our model without averaging yields a lower score than the RNN model with external features, though it is still bet-

ter than the RNN model alone. Without utilizing these four properties, our model obtained only 74.1% F1. These results indicate that syntactic and task-specific information and averaging contribute to the performance improvement. The improvement is achieved by a simple modification of compositional functions in RNN models.

4.2 Effects of Averaging in Training

Figure 3 shows the training curves in terms of F1 scores. These curves clearly demonstrate that parameter averaging helps to stabilize the learning and improve generalization capacity.

5 Conclusion

We have presented an averaged RNN model for semantic relation classification. Our experimental results show that syntactic information such as phrase categories and heads improves the performance, and the task-specific weighting is also beneficial. The results also demonstrate that averaging the model parameters not only stabilizes the learning but also improves the generalization capacity of the model. As future work, we plan to combine deep learning models with richer information such as predicate-argument structures.

Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. *Natural Language Processing (almost) from Scratch*. In *JMLR*, 12:2493–2537.

Christoph Goller and Andreas Küchler. 1996. *Learning Task-Dependent Distributed Representations by Backpropagation Through Structure*. In *ICNN*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano and Stan Szpakowicz. 2010. *SemEval-2010 Task 8: Multi-Way Classication of Semantic Relations Between Pairs of Nominals*. In *SemEval 2010*.

Karl Moritz Hermann and Phil Blunsom. 2013. *The Role of Syntax in Vector Space Models of Compositional Semantics*. In *ACL*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever and Ruslan R. Salakhutdinov. 2012. *Improving neural networks by preventing co-adaptation of feature detectors*. In *arXiv:1207.0580*.

Yusuke Miyao and Jun'ichi Tsujii. 2008. *Feature Forest Models for Probabilistic HPSG Parsing*. In *Computational Linguistics*, 34(1):35–80, MIT Press.

Bryan Rink and Sanda Harabagiu. 2010. *UTD: Classifying Semantic Relations by Combining Lexical and Semantic Resources*. In *SemEval 2010*.

Richard Socher, Brody Huval, Christopher D. Manning and Andrew Y. Ng. 2012. *Semantic Compositionality Through Recursive Matrix-Vector Spaces*. In *EMNLP*.

Richard Socher, John Bauer, Christopher D. Manning and Andrew Y. Ng. 2013. *Parsing with Compositional Vector Grammars*. In *ACL*.

Kevin Swersky, Bo Chen, Ben Marlin and Nando de Freitas. 2010. *A tutorial on stochastic approximation algorithms for training Restricted Boltzmann Machines and Deep Belief Nets*. In *ITA workshop*.

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features*. In *COLING/ACL*.

Improving Statistical Machine Translation with Word Class Models

Joern Wuebker, Stephan Peitz, Felix Rietig and Hermann Ney
Human Language Technology and Pattern Recognition Group
RWTH Aachen University
Aachen, Germany
<surname>@cs.rwth-aachen.de

Abstract

Automatically clustering words from a monolingual or bilingual training corpus into classes is a widely used technique in statistical natural language processing. We present a very simple and easy to implement method for using these word classes to improve translation quality. It can be applied across different machine translation paradigms and with arbitrary types of models. We show its efficacy on a small German→English and a larger French→German translation task with both standard phrase-based and hierarchical phrase-based translation systems for a common set of models. Our results show that with word class models, the baseline can be improved by up to 1.4% BLEU and 1.0% TER on the French→German task and 0.3% BLEU and 1.1% TER on the German→English task.

1 Introduction

Data sparsity is one of the major problems for statistical learning methods in natural language processing (NLP) today. Even with the huge training data sets available in some tasks, for many phenomena that need to be modeled only few training instances can be observed. This is partly due to the large vocabularies of natural languages. One possiblity to reduce the sparsity for model estimation is to reduce the vocabulary size. By clustering the vocabulary into a fixed number of word classes, it is possible to train models that are less prone to sparsity issues. This work investigates the performance of standard models used in statistical machine translation when they are trained on automatically learned word classes rather than the actual word identities.

In the popular tooklit GIZA++ (Och and Ney, 2003), word classes are an essential ingredient to model alignment probabilities with the HMM or IBM translation models. It contains the `mkcls` tool (Och, 1999), which can automatically cluster the vocabulary into classes.

Using this tool, we propose to re-parameterize the standard models used in statistical machine translation (SMT), which are usually conditioned on word identities rather than word classes. The idea is that this should lead to a smoother distribution, which is more reliable due to less sparsity. Here, we focus on the phrase-based and lexical channel models in both directions, simple count models identifying frequency thresholds, lexicalized reordering models and an n-gram language model. Although our results show that it is not a good idea to replace the original models, we argue that adding them to the log-linear feature combination can improve translation quality. They can easily be computed for different translation paradigms and arbitrary models. Training and decoding is possible without or with only little change to the code base.

Our experiments are conducted on a medium-sized French→German task and a small German→English task and with both phrase-based and hierarchical phrase-based translation decoders. By using word class models, we can improve our respective baselines by 1.4% BLEU and 1.0% TER on the French→German task and 0.3% BLEU and 1.1% TER on the German→English task.

Training an additional language model for trans-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1377–1381,
Seattle, Washington, USA, 18-21 October 2013.

lation based on word classes has been proposed in (Wuebker et al., 2012; Mediani et al., 2012; Koehn and Hoang, 2007). In addition to the reduced sparsity, an advantage of the smaller vocabulary is that longer n-gram context can be modeled efficiently.

Mathematically, our idea is equivalent to a special case of the Factored Translation Models proposed by Koehn and Hoang (2007). We will go into more detail in Section 4. Also related to our work, Cherry (2013) proposes to parameterize a hierarchical reordering model with sparse features that are conditioned on word classes trained with `mkcls`. However, the features are trained with MIRA rather than estimated by relative frequencies.

2 Word Class Models

2.1 Standard Models

The translation model of most phrase-based and hierarchical phrase-based SMT systems is parameterized by two phrasal and two lexical channel models (Koehn et al., 2003) which are estimated as relative frequencies. Their counts are extracted heuristically from a word aligned bilingual training corpus.

In addition to the four channel models, our baseline contains binary count features that fire, if the extraction count of the corresponding phrase pair is greater or equal to a given threshold τ. We use the thresholds $\tau = \{2, 3, 4\}$.

Our phrase-based baseline contains the hierarchical reordering model (HRM) described by Galley and Manning (2008). Similar to (Cherry et al., 2012), we apply it in both translation directions with separate scaling factors for the three orientation classes, leading to a total of six feature weights.

An n-gram language model (LM) is another important feature of our translation systems. The baselines apply 4-gram LMs trained by the SRILM toolkit (Stolcke, 2002) with interpolated modified Kneser-Ney smoothing (Chen and Goodman, 1998). The smaller vocabulary size allows us to efficiently model larger context, so in addition to the 4-gram LM, we also train a 7-gram LM based on word classes. In contrast to an LM of the same size trained on word identities, the increase in computational resources needed for translation is negligible for the 7-gram word class LM (wcLM).

2.2 Training

By replacing the words on both source and target side of the training data with their respective word classes and keeping the word alignment unchanged, all of the above models can easily be trained conditioned on word classes by using the same training procedure as usual. We end up with two separate model files, usually in the form of large tables, one with word identities and one with classes. Next, we sort both tables by their word classes. By walking through both sorted tables simultaneously, we can then efficiently augment the standard model file with an additonal feature (or additional features) based on word classes. The word class LM is directly passed on to the decoder.

2.3 Decoding

The decoder searches for the best translation given a set of models $h_m(e_1^I, s_1^K, f_1^J)$ by maximizing the log-linear feature score (Och and Ney, 2004):

$$\hat{e}_1^{\hat{I}} = \arg\max_{I, e_1^I} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, s_1^K, f_1^J) \right\}, \quad (1)$$

where $f_1^J = f_1 \dots f_J$ is the source sentence, $e_1^I = e_1 \dots e_I$ the target sentence and $s_1^K = s_1 \dots s_K$ the hidden alignment or derivation.

All the above mentioned models can easily be integrated into this framework as additional features h_m. The feature weights λ_m are tuned with minimum error rate training (MERT) (Och, 2003).

3 Experiments

3.1 Data

Our experiments are performed on a French→German task. In addition to some project-internal data, we train the system on the data provided for the WMT 2012 shared task[1]. Both the `dev` and the `test` set are composed of a mixture of broadcast news and broadcast conversations crawled from the web and have two references. Table 1 shows the data statistics.

To confirm our results we also run experiments on the German→English task of the IWSLT 2012 evaluation campaign[2].

[1] `http://www.statmt.org/wmt12/`
[2] `http://hltc.cs.ust.hk/iwslt/`

		French	German
train	Sentences	1.9M	
	Running Words	57M	50M
dev	Sentences	1900	
	Running Words	61K	55K
test	Sentences	2037	
	Running Words	60K	54K

Table 1: Corpus statistics for the French→German task. The running word counts for the German side of dev and test are averaged over both references.

3.2 Setup

In the French→German task, our baseline is a standard phrase-based system augmented with the hierarchical reordering model (HRM) described in Section 2.1. The language model is a 4-gram LM trained on all German monolingual sources provided for WMT 2012. For the class-based models, we run mkcls on the source and target side of the bilingual training data to cluster the vocabulary into 100 classes each. This clustering is used to train the models described above for word classes on the same training data as their counterparts based on word identity. This also holds for the wcLM, which is a 4-gram LM trained on the same data as the baseline LM. Further, the smaller vocabulary allows us to build an additional wcLM with a 7-gram context length. On this task we also run additional experiments with 200 and 500 classes.

On the German→English task, we evaluate our method for both a standard phrase-based and the hierarchical phrase-based baseline. Again, the phrase-based baseline contains the HRM model. As bilingual training data we use the *TED* talks, which we cluster into 100 classes on both source and target side. The 4-gram LM is trained on the *TED*, *Europarl* and *news-commentary* corpora. On this data set, we directly use a 7-gram wcLM.

In all setups, the feature weights are optimized with MERT. Results are reported in BLEU (Papineni et al., 2002) and TER (Snover et al., 2006), confidence level computation is based on (Koehn, 2004). Our experiments are conducted with the open source toolkit *Jane* (Wuebker et al., 2012; Vilar et al., 2010).

	dev		test	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
-TM +wcTM	21.2	64.2	24.7	59.5
-LM +wcLM	22.2	62.9	25.9	58.9
-HRM +wcHRM	24.6	61.9	27.5	58.1
phrase-based	24.6	61.8	27.8	57.6
+ wcTM	24.7	61.4	28.1	57.1
+ wcLM	24.9	61.2	28.4	57.1
+ wcHRM	25.4‡	60.9‡	28.9‡	56.9‡
+ wcLM^7	25.5‡	60.7‡	29.2‡	56.6‡
+ wcModels$_{200}$	25.5‡	60.8‡	29.3‡	56.4‡
+ wcModels$_{500}$	25.2†	60.8‡	29.0‡	56.6‡

Table 2: BLEU and TER results on the French→German task. Results marked with ‡ are statistically significant with 95% confidence, results marked with † with 90% confidence. *-X +wcX* denote the systems, where the model *X* in the baseline is replaced by its word class counterpart. The 7-gram word class LM is denoted as *wcLM*^{7}. *wcModels*$_X$ denotes all word class models trained on *X* classes.

3.3 Results

Results for the French→German task are given in Table 2. In a first set of experiments we replaced one of the standard TM, LM and HRM models by the same model based on word classes. Unsurprisingly, this degrades performance with different levels of severity. The strongest degradation can be seen when replacing the TM, while replacing the HRM only leads to a small drop in performance. However, when the word class models are added as additional features to the baseline, we observe improvements. The wcTM yields 0.3% BLEU and 0.5% TER on test. By adding the 4-gram wcLM, we get another 0.3% BLEU and the wcHRM shows further improvements of 0.5% BLEU and 0.2% TER. Extending the context length of the wcLM to 7-grams gives an additional boost, reaching a total gain over the baseline of 1.4% BLEU and 1.0% TER. Using 200 classes instead of 100 seems to perform slightly better on test, but with 500 classes, translation quality degrades again.

On the German→English task, the results shown in Table 3 are similar in TER, but less pronounced in BLEU. Here we are able to improve over the phrase-based baseline by 0.3% BLEU and 1.1% TER

	dev		test	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
phrase-based	30.2	49.6	28.6	51.6
+ wcTM	30.2	49.2	28.9	51.3
+ wcLM^7	30.5	48.3‡	29.0	50.6†
+ wcHRM	30.8	48.3‡	28.9	50.5‡
hiero	29.6	50.3	27.9	52.5
+ wcTM	29.8	50.3	28.1	52.3
+ wcLM^7	30.0	49.8	28.2	51.7

Table 3: BLEU and TER results on the German→English task. Results marked with ‡ are statistically significant with 95% confidence, results marked with † with 90% confidence.

by adding the wcTM, the 7-gram wcLM and the wcHRM. With the hierarchical decoder we gain 0.3% BLEU and 0.8% TER by adding the wcTM and the 7-gram wcLM.

4 Equivalence to Factored Translation

Koehn and Hoang (2007) propose to integrate different levels of annotation (e.g. morphologial analysis) as *factors* into the translation process. Here, the surface form of the source word is analyzed to produce the factors, which are then translated and finally the surface form of the target word is generated from the target factors. Although the translations of the factors operate on the same phrase segmentation, they are assumed to be independent. In practice this is done by *phrase expansion*, which generates a joint phrase table as the cross product from the phrase tables of the individual factors.

In contrast, in this work each word is mapped to a single class, which means that when we have selected a translation option for the surface form, the target side on the word class level is predetermined. Thus, no phrase expansion or generation steps are necessary to incorporate the word class information. The phrase table can simply be extended with additional scores, keeping the set of phrases constant.

Although the implementation is simpler, our approach is mathematically equivalent to a special case of the factored translation framework, which is shown in Figure 1. The generation step from target word e to its target class $c(e)$ assigns all probability

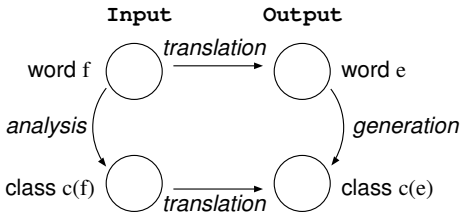


Figure 1: The factored translation model equivalent to our approach. The generation step assigns all probability mass to a single event: $p_{gen}(c(e)|e) = 1$.

mass to a single event:

$$p_{gen}(c|e) = \begin{cases} 1, \text{ if } c = c(e) \\ 0, \text{ else} \end{cases} \quad (2)$$

5 Conclusion

We have presented a simple and very easy to implement method to make use of word clusters for improving machine translation quality. It is applicable across different paradigms and for arbitrary types of models. Depending on the model type, it requires little or no change to the training and decoding software. We have shown the efficacy of this method on two translation tasks and with both the standard phrase-based and the hierarchical phrase-based translation paradigm. It was applied to relative frequency translation probabilities, the n-gram language model and a hierarchical reordering model. In our experiments, the baseline is improved by 1.4% BLEU and 1.0% TER on the French→German task and by 0.3% BLEU and 1.1% TER on the German→English task.

In future work we plan to apply our method to a wider range of languages. Intuitively, it should be most effective for morphologically rich languages, which naturally have stronger sparsity problems.

Acknowledgments

This work was partially realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation. The research leading to these results has also received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287658.

References

Stanley F. Chen and Joshuo Goodman. 1998. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, August.

Colin Cherry, Robert C. Moore, and Chris Quirk. 2012. On Hierarchical Re-ordering and Permutation Parsing for Phrase-based Decoding. In *Proceedings of the 7th Workshop on Statistical Machine Translation*, WMT '12, pages 200–209, Montral, Canada.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *The 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 22–31, Atlanta, Georgia, USA, June.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 847–855, Honolulu, Hawaii, USA, October.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868–876, Prague, Czech Republic, June.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, pages 127–133, Edmonton, Alberta.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain, July.

Mohammed Mediani, Yuqi Zhang, Thanh-Le Ha, Jan Niehues, Eunah Cho, Teresa Herrmann, and Alex Waibel. 2012. The kit translation systems for iwslt 2012. In *Proceedings of the International Workshop for Spoken Language Translation (IWSLT 2012)*, Hong Kong.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, December.

F. J. Och. 1999. An efficient method for determining bilingual word classes. In *Proc. of the Ninth Conf. of the Europ. Chapter of the Association of Computational Linguistics*, pages 71–76, Bergen, Norway, June.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.

Andreas Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proc. of the Int. Conf. on Speech and Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO, September.

David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. 2010. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, pages 262–270, Uppsala, Sweden, July.

Joern Wuebker, Matthias Huck, Stephan Peitz, Malte Nuhn, Markus Freitag, Jan-Thorsten Peter, Saab Mansour, and Hermann Ney. 2012. Jane 2: Open source phrase-based and hierarchical statistical machine translation. In *International Conference on Computational Linguistics*, pages 483–491, Mumbai, India, December.

Shift-Reduce Word Reordering for Machine Translation

Katsuhiko Hayashi†, Katsuhito Sudoh, Hajime Tsukada, Jun Suzuki, Masaaki Nagata
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
†hayashi.katsuhiko@lab.ntt.co.jp

Abstract

This paper presents a novel word reordering model that employs a shift-reduce parser for inversion transduction grammars. Our model uses rich syntax parsing features for word reordering and runs in linear time. We apply it to postordering of phrase-based machine translation (PBMT) for Japanese-to-English patent tasks. Our experimental results show that our method achieves a significant improvement of $+3.1$ BLEU scores against 30.15 BLEU scores of the baseline PBMT system.

1 Introduction

Even though phrase-based machine translation (PBMT) (Koehn et al., 2007) and tree-based MT (Graehl and Knight, 2004; Chiang, 2005; Galley et al., 2006) systems have achieved great success, many problems remain for distinct language pairs, including long-distant word reordering.

To improve such word reordering, one promising way is to separate it from the translation process as preordering (Collins et al., 2005; DeNero and Uszkoreit, 2011) or postordering (Sudoh et al., 2011; Goto et al., 2012). Many studies utilize a rule-based or a probabilistic model to perform a reordering decision at each node of a syntactic parse tree.

This paper presents a parser-based word reordering model that employs a shift-reduce parser for inversion transduction grammars (ITG) (Wu, 1997). To the best of our knowledge, this is the first study on a shift-reduce parser for word reordering.

The parser-based reordering approach uses rich syntax parsing features for reordering decisions. Our propoesd method can also easily define such non-local features as the N-gram words of reordered strings. Even when using these non-local features, the complexity of the shift-reduce parser does not increase at all due to give up achieving an optimal solution. Therefore, it works much more efficient.

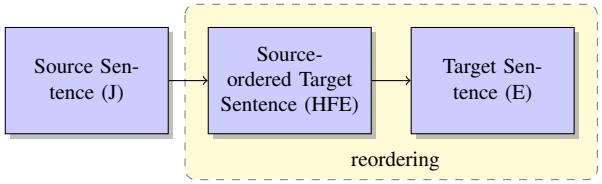


Figure 1: A description of the postordering MT system.

In our experiments, we apply our proposed method to postordering for J-to-E patent tasks because their training data for reordering have little noise and they are ideal for evaluating reordering methods. Although our used J-to-E setups need a language-dependent scheme and we describe our proposed method as a J-to-E postordering method, the key algorithm is language-independent and it can be applicable to preordering as well as postordering if the training data for reordering are available.

2 Postordering by Parsing

As shown in Fig.1, postordering (Sudoh et al., 2011) has two steps; the first is a translation step that translates an input sentence into source-ordered translations. The second is a reordering step in which the translations are reordered in the target language order. The key to postordering is the second step.

Goto et al. (2012) modeled the second step by parsing and created training data for a postordering parser using a language-dependent rule called **head-finalization**. The rule moves syntactic heads of a lexicalized parse tree of an English sentence to the

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1382–1386,
Seattle, Washington, USA, 18-21 October 2013.

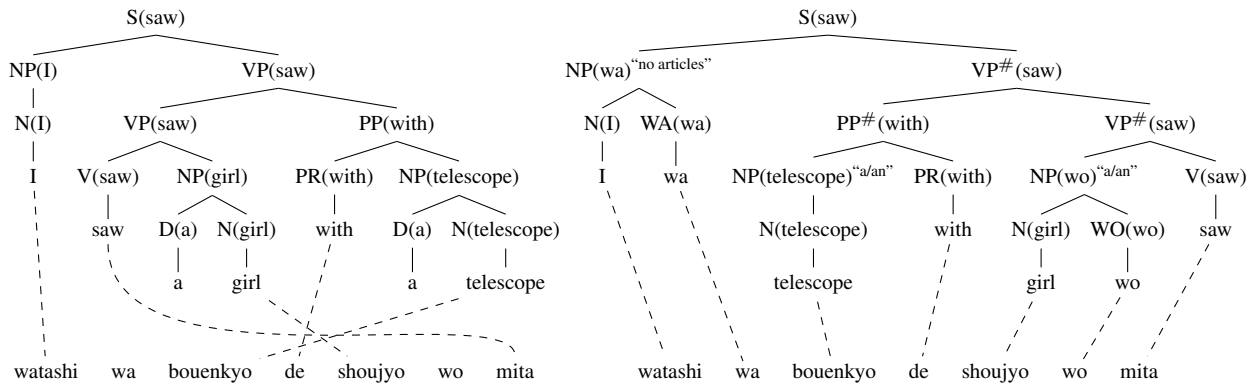


Figure 2: An example of the head-finzalizaton process for an English-Japanese sentence pair: the left-hand side tree is the original English tree, and the right-hand side tree is its head-final English tree.

end of the corresponding syntactic constituents. As a result, the terminal symbols of the English tree are sorted in a Japanese-like order. In Fig.2, we show an example of head-finalization and a tree on the right-hand side is a head-finalized English (HFE) tree of an English tree on the left-hand side. We annotate each parent node of the swapped edge with # symbol. For example, a nonterminal symbol PP$^{\#}$(with) shows that a noun phrase "a/an telescope" and a word "with" are inverted.

For better word alignments, Isozaki et al. (2012) also deleted articles "the" "a" "an" from English because Japanese has no articles, and inserted Japanese particles "ga" "wo" "wa" into English sentences. We privilege the nonterminals of a phrase modified by a deleted article to determine which "the" "a/an" or "no articles" should be inserted at the front of the phrase. Note that an original English sentence can be recovered from its HFE tree by using # symbols and annotated articles and deleting Japanese particles.

As well as Goto et al. (2012), we solve postordering by a parser whose model is trained with a set of HFE trees. The main difference between Goto et al. (2012)'s model and ours is that while the former simply used the Berkeley parser (Petrov and Klein, 2007), our shift-reduce parsing model can use such non-local task specific features as the N-gram words of reordered strings without sacrificing efficiency.

Our method integrates postediting (Knight and Chander, 1994) with reordering and inserts articles into English translations by learning an additional "insert" action of the parser. Goto et al. (2012) solved the article generation problem by using an N-gram language model, but this somewhat complicates their approach. Compared with other parsers, one advantage of the shift-reduce parser is to easily define such additional operations as "insert".

HFE trees can be defined as monolingual ITG trees (DeNero and Uszkoreit, 2011). Our monolingual ITG G is a tuple $G = (V, T, P, I, S)$ where V is a set of nonterminals, T is a set of terminals, P is a set of production rules, I is a set of nonterminals on which "the" "a/an" or "no articles" must be determined, and S is the start symbol.

Set P consists of terminal production rules that are responsible for generating word $w(\in T)$:

$$\mathrm{X} \rightarrow w$$

and binary production rules in two forms:

$$\begin{aligned} \mathrm{X} &\rightarrow \mathrm{Y}\,\mathrm{Z} \\ \mathrm{X}^{\#} &\rightarrow \mathrm{Y}\,\mathrm{Z} \end{aligned}$$

where X, X$^{\#}$, Y and Z are nonterminals. On the right-hand side, the second rule generates two phrases Y and Z in the reverse order. In our experiments, we removed all unary production rules.

3 Shift-Reduce Parsing

Given an input sentence $w_1 \ldots w_n$, the shift-reduce parser uses a stack of partial derivations, a buffer of input words, and a set of actions to build a parse tree.

The following is the parser's configuration:

$$\ell : \langle i, j, S \rangle : \pi$$

where ℓ is the step size, S is a stack of elements $s_0, s_1, \ldots$, i is the leftmost span index of the stack

top element s_0, j is an index of the next input word of the buffer, and π is a set of **predictor states**[1].

Each stack element has at least the following components of its partial derivation tree:

$$s = \{\mathrm{H}, h, w_{left}, w_{right}, a\}$$

where H is a root nonterminal or a part-of-speech tag of the subtree, h is a head index of H, a is a variable to which "the" "a/an" "no articles" or null are assigned, and w_{left}, w_{right} are the leftmost and rightmost words of phrase H. When referring to component $*$, we use a $s.*$ notation.

Our proposed system has 4 actions shift-X, insert-x, reduce-MR-X and reduce-SR-X.

The shift-X action pushes the next input word onto the stack and assigns a part-of-speech tag X to the word. The deduction step is as follows:

$$\frac{\mathrm{X} \rightarrow w_j \in P \qquad \overbrace{\ell : \langle i, j, S | s_0' \rangle : \pi}^{p}}{\ell + 1 : \langle j, j+1, S | s_0' | s_0 \rangle \rangle : \{p\}}$$

where s_0 is $\{\mathrm{X}, j, w_j, w_j, \mathrm{null}\}$.

The insert-x action determines whether to generate "the" "a/an" or "no articles" ($= x$):

$$\frac{s_0'.\mathrm{X} \in I \wedge (s_0'.a \neq \text{“the”} \wedge s_0'.a \neq \text{“a/an”}) \quad \ell : \langle i, j, S | s_0' \rangle \rangle : \pi}{\ell + 1 : \langle i, j, S | s_0 \rangle : \pi}$$

where s_0' is $\{\mathrm{X}, h, w_{left}, w_{right}, a\}$ and s_0 is $\{\mathrm{X}, h, w_{left}, w_{right}, x\}$ ($i \leq h, left, right < j$). The side condition prevents the parser from inserting articles into phrase X more than twice. During parsing, articles are not explicitly inserted into the input string: they are inserted into it when backtracking to generate a reordered string after parsing.

The reduce-MR-X action has a deduction rule:

$$\frac{\mathrm{X} \rightarrow \mathrm{Y\ Z} \in P \wedge q \in \pi \qquad \overbrace{\_ : \langle k, i, S | s_2' | s_1' \rangle : \pi'}^{q} \quad \ell : \langle i, j, S | s_1' | s_0' \rangle : \pi}{\ell + 1 : \langle k, j, S | s_2' | s_0 \rangle : \pi'}$$

[1]Since our notion of predictor states is identical to that in (Huang and Sagae, 2010), we omit the details here.

$s_0.w_h \circ s_0.t_h$ $s_0.\mathrm{H}$ $s_0.\mathrm{H} \circ s_0.t_h$ $s_0.w_h \circ s_0.\mathrm{H}$ $s_1.w_h \circ s_1.t_h$ $s_1.\mathrm{H}$ $s_1.\mathrm{H} \circ s_1.t_h$ $s_1.w_h \circ s_1.\mathrm{H}$ $s_2.t_h \circ s_2.\mathrm{H}$ $s_2.w_h \circ s_2.\mathrm{H}$ $q_0.w$ $q_1.w$ $q_2.w$
$s_0.t_l \circ s_0.\mathrm{L}$ $s_0.w_l \circ s_0.\mathrm{L}$ $s_1.t_l \circ s_1.\mathrm{L}$ $s_1.w_l \circ s_1.\mathrm{L}$
$s_0.w_h \circ s_0.\mathrm{H} \circ s_1.w_h \circ s_1.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.w_h$ $s_0.w_h \circ s_1.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.\mathrm{H}$ $s_0.w_h \circ s_0.\mathrm{H} \circ q_0.w$ $s_0.\mathrm{H} \circ q_0.w$ $s_1.w_h \circ s_1.\mathrm{H} \circ q_0.w$ $s_1.\mathrm{H} \circ q_0.w$ $s_1.t_h \circ q_0.w \circ q_1.w$
$s_0.w_h \circ s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ q_0.w$ $s_0.\mathrm{H} \circ s_1.w_h \circ s_1.\mathrm{H} \circ q_0.w$ $s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ q_0.w$ $s_0.t_h \circ s_1.t_h \circ q_0.w$ $s_0.w_h \circ s_1.\mathrm{H} \circ q_0.w \circ q_1.w$ $s_0.\mathrm{H} \circ q_0.w \circ q_1.w$ $s_0.t_h \circ q_0.w \circ q_1.w$ $s_0.w_h \circ s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_2.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.w_h \circ s_1.\mathrm{H} \circ s_2.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_2.w_h \circ s_2.\mathrm{H}$ $s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_2.\mathrm{H}$ $s_0.t_h \circ s_1.t_h \circ s_2.t_h$
$s_0.\mathrm{H} \circ s_0.\mathrm{R} \circ s_0.\mathrm{L}$ $s_1.\mathrm{H} \circ s_1.\mathrm{R} \circ s_1.\mathrm{L}$ $s_0.\mathrm{H} \circ s_0.\mathrm{R} \circ q_0.w$ $s_0.\mathrm{H} \circ s_0.\mathrm{L} \circ s_1.\mathrm{H}$ $s_0.\mathrm{H} \circ s_0.\mathrm{L} \circ s_1.w_h$ $s_0.\mathrm{H} \circ s_1.\mathrm{H} \circ s_1.\mathrm{L}$ $s_0.w_h \circ s_1.\mathrm{H} \circ s_1.\mathrm{R}$
$s_0.w_{left} \circ s_1.w_{right}$ $s_0.t_{left} \circ s_1.t_{right}$ $s_0.w_{right} \circ s_1.w_{left}$ $s_0.t_{right} \circ s_1.t_{left}$
$s_0.a \circ s_0.w_{left}$ $s_0.a \circ s_0.t_{left}$ $s_0.a \circ s_0.w_{left} \circ s_1.w_{right}$ $s_0.a \circ s_0.t_{left} \circ s_1.t_{right}$ $s_0.a \circ s_0.w_h$ $s_0.a \circ s_0.t_h$

Table 1: Feature templates: $s.\mathrm{L}$ and $s.\mathrm{R}$ denote the left and right subnodes of s. l and r are head indices of L and R. q denotes a buffer element. t is a part-of-speech tag.

where s_0' is $\{\mathrm{Z}, h_0, w_{left0}, w_{right0}, a_0\}$ and s_1' is $\{\mathrm{Y}, h_1, w_{left1}, w_{right1}, a_1\}$. The action generates s_0 by combining s_0' and s_1' with binary rule X→Y Z:

$$s_0 = \{\mathrm{X}, h_0, w_{left1}, w_{right0}, a_1\}.$$

New nonterminal X is lexicalized with head word w_{h_0} of right nonterminal Z. This action expands Y and Z in a straight order. The leftmost word of phrase X is set to leftmost word w_{left1} of Y, and the rightmost word of phrase X is set to rightmost word w_{right0} of Z. Variable a is set to a_1 of Y.

The difference between reduce-MR-X and reduce-SR-X actions is new stack element s_0. The reduce-SR-X action generates s_0 by combining s_0' and s_1' with binary rule $\mathrm{X}^{\#}$ →Y Z:

$$s_0 = \{\mathrm{X}^{\#}, h_0, w_{left0}, w_{right1}, a_0\}.$$

This action expands Y and Z in a reverse order, and the leftmost word of $\mathrm{X}^{\#}$ is set to w_{left0} of Z, and the rightmost word of $\mathrm{X}^{\#}$ is set to w_{right1} of Y. Variable a is set to a_0 of Z.

We use a linear model that is discriminatively trained with the averaged perceptron (Collins and Roark, 2004). Table 1 shows the feature templates used in our experiments and we call the features in the bottom two rows "non-local" features.

	train	dev	test9	test10
# of sent.	3,191,228	2,000	2,000	2,300
ave. leng. (J)	36.4	36.6	37.0	43.1
ave. leng. (E)	33.3	33.3	33.7	39.6

Table 2: NTCIR-9 and 10 data statistics.

4 Experiments

4.1 Experimental Setups

We conducted experiments for NTCIR-9 and 10 patent data using a Japanese-English language pair. Mecab[2] was used for the Japanese morphological analysis. The data are summarized in Table 2.

We used Enju (Miyao and Tsujii, 2008) for parsing the English training data and converted parse trees into HFE trees by a head-finalization scheme. We extracted grammar rules from all the HFE trees and randomly selected 500,000 HFE trees to train the shift-reduce parser.

We used Moses (Koehn et al., 2007) with lexicalized reordering and a 6-gram language model (LM) trained using SRILM (Stolcke et al., 2011) to translate the Japanese sentences into HFE sentences.

To recover the English sentences, our shift-reduce parser reordered only the 1-best HFE sentence. Our strategy is much simpler than Goto et al. (2012)'s because they used a linear inteporation of MT cost, parser cost and N-gram LM cost to generate the best English sentence from the n-best HFE sentences.

4.2 Main Results

The main results in Table 3 indicate our method was significantly better and faster than the conventional PBMT system. Our method also ourperformed Goto et al. (2012)'s reported systems as well as a tree-based (moses-chart) system[3]. Our proposed model with "non-local" features (w/ nf.) achieved gains against that without the features (w/o nf.). Further feature engineering may improve the accuracy more.

4.3 Analysis

We show N-gram precisions of PBMT (dist=6, dist=20) and proposed systems in Table 5. The results clearly show that improvements of 1-gram precisions are the main factors that contribute to better performance of our proposed system than PBMT systems. It seems that the gains of 1-gram presicions come from postediting (article generation).

In table 4, we show the effectiveness of our joint reordering and postediting approach ("proposed"). The "w/o art." results clearly show that generating articles has great effects on MT evaluations especially for BLEU metric. Comparing "proposed" and "HFE w/ art." systems, these results show that postediting is much more effective than generating articles by MT. Our joint approach also outperformed "N-gram" postediting system.

	test9		test10	
	BLEU	RIBES	BLEU	RIBES
HFE w/ art.	28.86	73.45	29.9	73.52
proposed	**32.93**	**76.68**	**33.25**	**76.74**
w/o art.	19.86	75.62	20.17	75.63
N-gram	32.15	76.52	32.28	76.46

Table 4: The effects of article generation: "w/o art." denotes evaluation scores for translations of the best system ("proposed") in Table 3 from which articles are removed. "HFE w/ art." system used HFE data with articles and generated them by MT system and the shift-reduce parser performed only reordering. "N-gram" system inserted articles into the translations of "w/o art." by Goto et al. (2012)'s article generation method.

	(1–4)-gram precision
moses (dist=6)	67.1 / 36.9 / 20.7 / 11.5
moses (dist=20)	67.7 / 38.9 / 23.0 / 13.7
proposed	68.9 / 40.6 / 25.7 / 16.7

Table 5: N-gram precisions of moses (dist=6, dist=20) and proposed systems for test9 data.

5 Conclusion

We proposed a shift-reduce word ordering model and applied it to J-to-E postordering. Our experimental results indicate our method can significantly improve the performance of a PBMT system.

Future work will investigate our method's usefulness on various language datasets. We plan to study more general methods that use word alignments to embed swap information in trees (Galley et al., 2006).

[2] https://code.google.com/p/mecab/

[3] All the data and the MT toolkits used in our experiments are the same as theirs.

	test9			test10		
	BLEU	RIBES	time (sec.)	BLEU	RIBES	time (sec.)
PBMT (dist=6)	27.1	67.76	2.66	27.92	68.13	3.18
PBMT (dist=12)	29.55	69.84	4.15	30.03	69.88	4.93
PBMT (dist=20)	29.98	69.87	6.22	30.15	69.43	7.19
Tree-based MT** (Goto et al., 2012)	29.53	69.22	–	–	–	–
PBMT (dist=20)** (Goto et al., 2012)	30.13	68.86	–	–	–	–
Goto et al. (2012)**	31.75	72.57	–	–	–	–
PBMT (dist=0) + proposed w/o nf. (beam=12)	32.59	76.35	1.46 + 0.01	32.83	76.44	1.7 + 0.01
PBMT (dist=0) + proposed w/o nf. (beam=48)	32.61	76.58	1.46 + 0.06	32.86	76.6	1.7 + 0.06
PBMT (dist=0) + proposed w/ nf. (beam=12)	32.91	76.38	1.46 + 0.01	33.15	76.53	1.7 + 0.02
PBMT (dist=0) + proposed w/ nf. (beam=48)	**32.93**	**76.68**	1.46 + 0.07	**33.25**	**76.74**	1.7 + 0.07

Table 3: System comparison: time represents the average second per sentence. ** denotes "not our experiments".

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 531–540.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968.

Isao Goto, Masao Utiyama, and Eiichiro Sumita. 2012. Post-ordering by parsing for japanese-english statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 311–316.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086.

Hideki Isozaki, Jun Suzuki, Hajime Tsukada, Masaaki Nagata, Sho Hoshino, and Yusuke Miyao. 2012. HPSG-based preprocessing for English-to-Japanese translation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 11(3).

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 779–779.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34(1):35–80.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics*, pages 404–411.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*.

Katsuhito Sudoh, Xianchao Wu, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2011. Post-ordering in statistical machine translation. In *Proc. MT Summit*.

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

Decoding with Large-Scale Neural Language Models Improves Translation

Ashish Vaswani
University of Southern California
Department of Computer Science
avaswani@isi.edu

Yinggong Zhao
Nanjing University, State Key Laboratory
for Novel Software Technology
zhaoyg@nlp.nju.edu.cn

Victoria Fossum and **David Chiang**
University of Southern California
Information Sciences Institute
{vfossum,chiang}@isi.edu

Abstract

We explore the application of neural language models to machine translation. We develop a new model that combines the neural probabilistic language model of Bengio et al., rectified linear units, and noise-contrastive estimation, and we incorporate it into a machine translation system both by reranking k-best lists and by direct integration into the decoder. Our large-scale, large-vocabulary experiments across four language pairs show that our neural language model improves translation quality by up to 1.1 B .

1 Introduction

Machine translation (MT) systems rely upon language models (LMs) during decoding to ensure fluent output in the target language. Typically, these LMs are n-gram models over discrete representations of words. Such models are susceptible to data sparsity–that is, the probability of an n-gram observed only few times is difficult to estimate reliably, because these models do not use any information about similarities between words.

To address this issue, Bengio et al. (2003) propose distributed word representations, in which each word is represented as a real-valued vector in a high-dimensional feature space. Bengio et al. (2003) introduce a feed-forward neural probabilistic LM (NPLM) that operates over these distributed representations. During training, the NPLM learns both a distributed representation for each word in the vocabulary and an n-gram probability distribution over words in terms of these distributed representations.

Although neural LMs have begun to rival or even surpass traditional n-gram LMs (Mnih and Hinton, 2009; Mikolov et al., 2011), they have not yet been widely adopted in large-vocabulary applications such as MT, because standard maximum likelihood estimation (MLE) requires repeated summations over all words in the vocabulary. A variety of strategies have been proposed to combat this issue, many of which require severe restrictions on the size of the network or the size of the data.

In this work, we extend the NPLM of Bengio et al. (2003) in two ways. First, we use rectified linear units (Nair and Hinton, 2010), whose activations are cheaper to compute than sigmoid or tanh units. There is also evidence that deep neural networks with rectified linear units can be trained successfully without pre-training (Zeiler et al., 2013). Second, we train using noise-contrastive estimation or NCE (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012), which does not require repeated summations over the whole vocabulary. This enables us to efficiently build NPLMs on a larger scale than would be possible otherwise.

We then apply this LM to MT in two ways. First, we use it to rerank the k-best output of a hierarchical phrase-based decoder (Chiang, 2007). Second, we integrate it directly into the decoder, allowing the neural LM to more strongly influence the model. We achieve gains of up to 0.6 B translating French, German, and Spanish to English, and up to 1.1 B on Chinese-English translation.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1387–1392,
Seattle, Washington, USA, 18-21 October 2013.

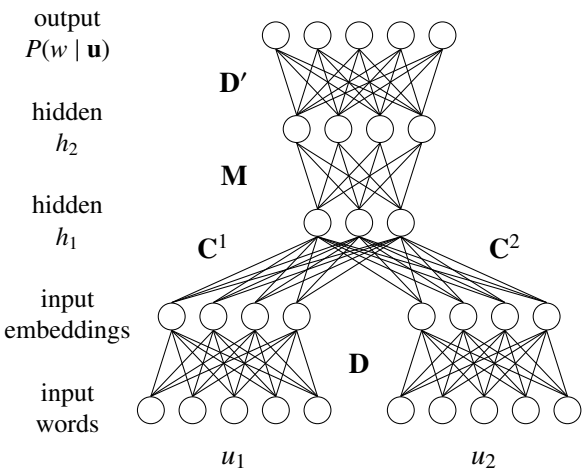


Figure 1: Neural probabilistic language model (Bengio et al., 2003).

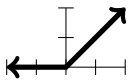
2 Neural Language Models

Let V be the vocabulary, and n be the order of the language model; let $\mathbf{u}$ range over contexts, i.e., strings of length $(n-1)$, and w range over words. For simplicity, we assume that the training data is a single very long string, $w_1 \cdots w_N$, where w_N is a special stop symbol, </s>. We write $\mathbf{u}_i$ for $w_{i-n+1} \cdots w_{i-1}$, where, for $i \leq 0$, w_i is a special start symbol, <s>.

2.1 Model

We use a feedforward neural network as shown in Figure 1, following Bengio et al. (2003). The input to the network is a sequence of one-hot representations of the words in context $\mathbf{u}$, which we write u_j $(1 \leq j \leq n-1)$. The output is the probability $P(w \mid \mathbf{u})$ for each word w, which the network computes as follows.

The hidden layers consist of rectified linear units (Nair and Hinton, 2010), which use the activation function $\phi(x) = \max(0, x)$ (see graph at right).



The output of the first hidden layer h_1 is

$$h_1 = \phi\left(\sum_{j=1}^{n-1} \mathbf{C}^j \mathbf{D} u_j\right) \tag{1}$$

where $\mathbf{D}$ is a matrix of input word embeddings which is shared across all positions, the $\mathbf{C}^j$ are the context matrices for each word in $\mathbf{u}$, and ϕ is applied elementwise. The output of the second layer h_2 is

$$h_2 = \phi\left(\mathbf{M} h_1\right),$$

where $\mathbf{M}$ is the matrix of connection weights between h_1 and h_2. Finally, the output layer is a softmax layer,

$$P(w \mid \mathbf{u}) \propto \exp\left(\mathbf{D}' h_2 + \mathbf{b}\right) \tag{2}$$

where $\mathbf{D}'$ is the output word embedding matrix and $\mathbf{b}$ is a vector of biases for every word in the vocabulary.

2.2 Training

The typical way to train neural LMs is to maximize the likelihood of the training data by gradient ascent. But the softmax layer requires, at each iteration, a summation over all the units in the output layer, that is, all words in the whole vocabulary. If the vocabulary is large, this can be prohibitively expensive.

Noise-contrastive estimation or NCE (Gutmann and Hyvärinen, 2010) is an alternative estimation principle that allows one to avoid these repeated summations. It has been applied previously to log-bilinear LMs (Mnih and Teh, 2012), and we apply it here to the NPLM described above.

We can write the probability of a word w given a context $\mathbf{u}$ under the NPLM as

$$\begin{aligned} P(w \mid \mathbf{u}) &= \frac{1}{Z(\mathbf{u})} p(w \mid \mathbf{u}) \\ p(w \mid \mathbf{u}) &= \exp\left(\mathbf{D}' h_2 + \mathbf{b}\right) \\ Z(\mathbf{u}) &= \sum_{w'} p(w' \mid \mathbf{u}) \end{aligned} \tag{3}$$

where $p(w \mid \mathbf{u})$ is the *unnormalized* output of the unit corresponding to w, and $Z(\mathbf{u})$ is the normalization factor. Let θ stand for the parameters of the model.

One possibility would be to treat $Z(\mathbf{u})$, instead of being defined by (3), as an additional set of model parameters which are learned along with θ. But it is easy to see that we can make the likelihood arbitrarily large by making the $Z(\mathbf{u})$ arbitrarily small.

In NCE, we create a noise distribution $q(w)$. For each example $\mathbf{u}_i w_i$, we add k noise samples $\bar{w}_{i1}, \ldots, \bar{w}_{ik}$ into the data, and extend the model to account for noise samples by introducing a random

variable C which is 1 for training examples and 0 for noise samples:

$$P(C = 1, w \mid \mathbf{u}) = \frac{1}{1+k} \cdot \frac{1}{Z(\mathbf{u})} p(w \mid \mathbf{u})$$
$$P(C = 0, w \mid \mathbf{u}) = \frac{k}{1+k} \cdot q(w).$$

We then train the model to classify examples as training data or noise, that is, to maximize the *conditional* likelihood,

$$L = \sum_{i=1}^{N} \Bigg(\log P(C = 1 \mid \mathbf{u}_i w_i) + \sum_{j=1}^{k} \log P(C = 0 \mid \mathbf{u}_i \bar{w}_{ij}) \Bigg)$$

with respect to both θ and $Z(\mathbf{u})$.

We do this by stochastic gradient ascent. The gradient with respect to θ turns out to be

$$\frac{\partial L}{\partial \theta} = \sum_{i=1}^{N} \Bigg(P(C = 0 \mid \mathbf{u}_i w_i) \frac{\partial}{\partial \theta} \log p(w_i \mid \mathbf{u}_i) - \sum_{j=1}^{k} P(C = 1 \mid \mathbf{u}_i \bar{w}_{ij}) \frac{\partial}{\partial \theta} \log p(\bar{w}_{ij} \mid \mathbf{u}_i) \Bigg)$$

and similarly for the gradient with respect to $Z(\mathbf{u})$. These can be computed by backpropagation. Unlike before, the $Z(\mathbf{u})$ will converge to a value that normalizes the model, satisfying (3), and, under appropriate conditions, the parameters will converge to a value that maximizes the likelihood of the data.

3 Implementation

Both training and scoring of neural LMs are computationally expensive at the scale needed for machine translation. In this section, we describe some of the techniques used to make them practical for translation.

3.1 Training

During training, we compute gradients on an entire minibatch at a time, allowing the use of matrix-matrix multiplications instead of matrix-vector multiplications (Bengio, 2012). We represent the inputs as a sparse matrix, allowing the computation of the input layer (1) to use sparse matrix-matrix multiplications. The output activations (2) are computed only for the word types that occur as the positive example or one of the noise samples, yielding a sparse matrix of outputs. Similarly, during backpropagation, sparse matrix multiplications are used at both the output and input layer.

In most of these operations, the examples in a minibatch can be processed in parallel. However, in the sparse-dense products used when updating the parameters $\mathbf{D}$ and $\mathbf{D}'$, we found it was best to divide the vocabulary into blocks (16 per thread) and to process the blocks in parallel.

3.2 Translation

To incorporate this neural LM into a MT system, we can use the LM to rerank k-best lists, as has been done in previous work. But since the NPLM scores n-grams, it can also be integrated into a phrase-based or hierarchical phrase-based decoder just as a conventional n-gram model can, unlike a RNN.

The most time-consuming step in computing n-gram probabilities is the computation of the normalization constants $Z(\mathbf{u})$. Following Mnih and Teh (2012), we set all the normalization constants to one during training, so that the model learns to produce approximately normalized probabilities. Then, when applying the LM, we can simply ignore normalization. A similar strategy was taken by Niehues and Waibel (2012). We find that a single n-gram lookup takes about 40 μs.

The technique, described above, of grouping examples into minibatches works for scoring of k-best lists, but not while decoding. But caching n-gram probabilities helps to reduce the cost of the many lookups required during decoding.

A final issue when decoding with a neural LM is that, in order to estimate future costs, we need to be able to estimate probabilities of n'-grams for $n' < n$. In conventional LMs, this information is readily available,[1] but not in NPLMs. Therefore, we defined a special word <null> whose embedding is the weighted average of the (input) embeddings of all the other words in the vocabulary. Then, to estimate the probability of an n'-gram $\mathbf{u}'w$, we used the probability of $P(w \mid \texttt{<null>}^{n-n'}\mathbf{u}')$.

[1] However, in Kneser-Ney smoothed LMs, this information is also incorrect (Heafield et al., 2012).

setting	dev	2004	2005	2006
baseline	38.2	38.4	37.7	34.3
reranking	38.5	38.6	37.8	34.7
decoding	39.1	39.5	38.8	34.9

Table 1: Results for Chinese-English experiments, without neural LM (baseline) and with neural LM for reranking and integrated decoding. Reranking with the neural LM improves translation quality, while integrating it into the decoder improves even more.

	Fr-En		De-En		Es-En	
setting	dev	test	dev	test	dev	test
baseline	33.5	25.5	28.8	21.5	33.5	32.0
reranking	33.9	26.0	29.1	21.5	34.1	32.2
decoding	34.1[2]	26.1[2]	29.3	21.9	34.2[2]	32.1[2]

Table 2: Results for Europarl MT experiments, without neural LM (baseline) and with neural LM for reranking and integrated decoding. The neural LM gives improvements across three different language pairs. Superscript 2 indicates a score averaged between two runs; all other scores were averaged over three runs.

4 Experiments

We ran experiments on four language pairs – Chinese to English and French, German, and Spanish to English – using a hierarchical phrase-based MT system (Chiang, 2007) and GIZA++ (Och and Ney, 2003) for word alignments.

For all experiments, we used four LMs. The baselines used conventional 5-gram LMs, estimated with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the English side of the bitext and the 329M-word Xinhua portion of English Gigaword (LDC2011T07). Against these baselines, we tested systems that included the two conventional LMs as well as two 5-gram NPLMs trained on the same datasets. The Europarl bitext NPLMs had a vocabulary size of 50k, while the other NPLMs had a vocabulary size of 100k. We used 150 dimensions for word embeddings, 750 units in hidden layer h_1, and 150 units in hidden layer h_2. We initialized the network parameters uniformly from $(-0.01, 0.01)$ and the output biases to $-\log|V|$, and optimized them by 10 epochs of stochastic gradient ascent, using minibatches of size 1000 and a learning rate of 1. We drew 100 noise samples per training example from the unigram distribution, using the alias method for efficiency (Kronmal and Peterson, 1979).

We trained the discriminative models with MERT (Och, 2003) and the discriminative rerankers on 1000-best lists with MERT. Except where noted, we ran MERT three times and report the average score. We evaluated using case-insensitive NIST B .

4.1 NIST Chinese-English

For the Chinese-English task (Table 1), the training data came from the NIST 2012 constrained track, excluding sentences longer than 60 words. Rules without nonterminals were extracted from all training data, while rules with nonterminals were extracted from the FBIS corpus (LDC2003E14). We ran MERT on the development data, which was the NIST 2003 test data, and tested on the NIST 2004–2006 test data.

Reranking using the neural LM yielded improvements of 0.2–0.4 B , while integrating the neural LM yielded larger improvements, between 0.6 and 1.1 B .

4.2 Europarl

For French, German, and Spanish translation, we used a parallel text of about 50M words from Europarl v7. Rules without nonterminals were extracted from all the data, while rules with nonterminals were extracted from the first 200k words. We ran MERT on the development data, which was the WMT 2005 test data, and tested on the WMT 2006 news commentary test data (nc-test2006).

The improvements, shown in Table 2, were more modest than on Chinese-English. Reranking with the neural LM yielded improvements of up to 0.5 B , and integrating the neural LM into the decoder yielded improvements of up to 0.6 B . In one case (Spanish-English), integrated decoding scored higher than reranking on the development data but lower on the test data – perhaps due to the difference in domain between the two. On the other tasks, integrated decoding outperformed reranking.

4.3 Speed comparison

We measured the speed of training a NPLM by NCE, compared with MLE as implemented by the CSLM toolkit (Schwenk, 2013). We used the first 200k

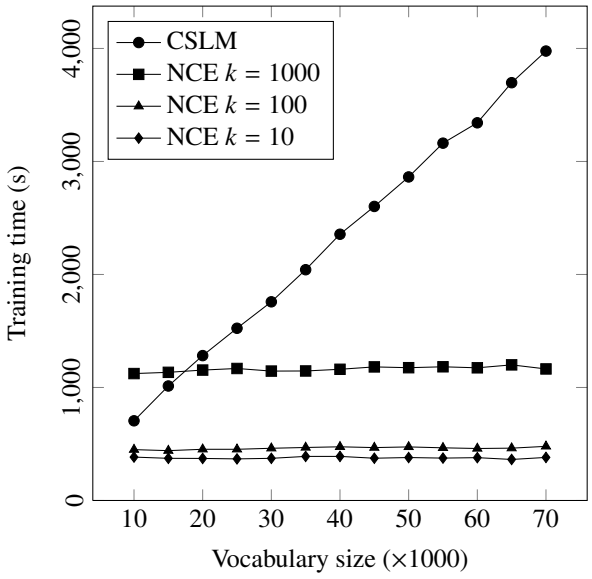


Figure 2: Noise contrastive estimation (NCE) is much faster, and much less dependent on vocabulary size, than MLE as implemented by the CSLM toolkit (Schwenk, 2013).

lines (5.2M words) of the Xinhua portion of Gigaword and timed one epoch of training, for various values of k and $|V|$, on a dual hex-core 2.67 GHz Xeon X5650 machine. For these experiments, we used minibatches of 128 examples. The timings are plotted in Figure 2. We see that NCE is considerably faster than MLE; moreover, as expected, the MLE training time is roughly linear in $|V|$, whereas the NCE training time is basically constant.

5 Related Work

The problem of training with large vocabularies in NPLMs has received much attention. One strategy has been to restructure the network to be more hierarchical (Morin and Bengio, 2005; Mnih and Hinton, 2009) or to group words into classes (Le et al., 2011). Other strategies include restricting the vocabulary of the NPLM to a *shortlist* and reverting to a traditional n-gram LM for other words (Schwenk, 2004), and limiting the number of training examples using resampling (Schwenk and Gauvain, 2005) or selecting a subset of the training data (Schwenk et al., 2012). Our approach can be efficiently applied to large-scale tasks without limiting either the model or the data.

NPLMs have previously been applied to MT, most notably feed-forward NPLMs (Schwenk, 2007; Schwenk, 2010) and RNN-LMs (Mikolov, 2012). However, their use in MT has largely been limited to reranking k-best lists for MT tasks with restricted vocabularies. Niehues and Waibel (2012) integrate a RBM-based language model directly into a decoder, but they only train the RBM LM on a small amount of data. To our knowledge, our approach is the first to integrate a large-vocabulary NPLM directly into a decoder for a large-scale MT task.

6 Conclusion

We introduced a new variant of NPLMs that combines the network architecture of Bengio et al. (2003), rectified linear units (Nair and Hinton, 2010), and noise-contrastive estimation (Gutmann and Hyvärinen, 2010). This model is dramatically faster to train than previous neural LMs, and can be trained on a large corpus with a large vocabulary and directly integrated into the decoder of a MT system. Our experiments across four language pairs demonstrated improvements of up to 1.1 B . Code for training and using our NPLMs is available for download.[2]

Acknowledgements

We would like to thank the anonymous reviewers for their very helpful comments. This research was supported in part by DOI IBC grant D12AP00225. This work was done while the second author was visiting USC/ISI supported by China Scholarship Council. He was also supported by the Research Fund for the Doctoral Program of Higher Education of China (No. 20110091110003) and the National Fundamental Research Program of China (2010CB327903).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language mod-

[2] `http://nlg.isi.edu/software/nplm`

eling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

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

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of AISTATS*.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2012. Language model rest costs and space-efficient storage. In *Proceedings of EMNLP-CoNLL*, pages 1169–1178.

Richard Kronmal and Arthur Peterson. 1979. On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Proceedings of ICASSP*, pages 5524–5527.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan "Honza" Černocký. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of INTERSPEECH*, pages 605–608.

Tomáš Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Andriy Mnih and Geoffrey Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of ICML*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*, pages 246–252.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using Restricted Boltzmann Machines. In *Proceedings of IWSLT*.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Holger Schwenk and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *Proceedings of EMNLP*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a GPU for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19.

Holger Schwenk. 2004. Efficient training of large neural networks for language modeling. In *Proceedings of IJCNN*, pages 3059–3062.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21:492–518.

Holger Schwenk. 2010. Continuous-space language models for statistical machine translation. *Prague Bulletin of Mathematical Linguistics*, 93:137–146.

Holger Schwenk. 2013. CSLM - a modular open-source continuous space language modeling toolkit. In *Proceedings of Interspeech*.

M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton. 2013. On rectified linear units for speech processing. In *Proceedings of ICASSP*.

Bilingual Word Embeddings for Phrase-Based Machine Translation

Will Y. Zou[†], **Richard Socher, Daniel Cer, Christopher D. Manning**
Department of Electrical Engineering[†] and Computer Science Department
Stanford University, Stanford, CA 94305, USA
{wzou,danielcer,manning}@stanford.edu, richard@socher.org

Abstract

We introduce bilingual word embeddings: semantic embeddings associated across two languages in the context of neural language models. We propose a method to learn bilingual embeddings from a large unlabeled corpus, while utilizing MT word alignments to constrain translational equivalence. The new embeddings significantly out-perform baselines in word semantic similarity. A single semantic similarity feature induced with bilingual embeddings adds near half a BLEU point to the results of NIST08 Chinese-English machine translation task.

1 Introduction

It is difficult to recognize and quantify semantic similarities across languages. The Fr-En phrase-pair {'*un cas de force majeure*', '*case of absolute necessity*'}, Zh-En phrase pair {'依然故我','*persist in a stubborn manner*'} are similar in semantics. If co-occurrences of exact word combinations are rare in the training parallel text, it can be difficult for classical statistical MT methods to identify this similarity, or produce a reasonable translation given the source phrase.

We introduce an unsupervised neural model to learn bilingual semantic embedding for words across two languages. As an extension to their monolingual counter-part (Turian et al., 2010; Huang et al., 2012; Bengio et al., 2003), bilingual embeddings capture not only semantic information of monolingual words, but also semantic relationships across different languages. This property allows them to define semantic similarity metrics across phrase-pairs, making them perfect features for machine translation.

To learn bilingual embeddings, we use a new objective function which embodies both monolingual semantics and bilingual translation equivalence. The latter utilizes word alignments, a natural sub-task in the machine translation pipeline. Through large-scale curriculum training (Bengio et al., 2009), we obtain bilingual distributed representations which lie in the same feature space. Embeddings of direct translations overlap, and semantic relationships across bilingual embeddings were further improved through unsupervised learning on a large unlabeled corpus.

Consequently, we produce for the research community a first set of Mandarin Chinese word embeddings with 100,000 words trained on the Chinese Gigaword corpus. We evaluate these embedding on Chinese word semantic similarity from *SemEval-2012* (Jin and Wu, 2012). The embeddings significantly out-perform prior work and pruned *tf-idf* base-lines. In addition, the learned embeddings give rise to 0.11 F1 improvement in Named Entity Recognition on the OntoNotes dataset (Hovy et al., 2006) with a neural network model.

We apply the bilingual embeddings in an end-to-end phrase-based MT system by computing semantic similarities between phrase pairs. On NIST08 Chinese-English translation task, we obtain an improvement of 0.48 BLEU from a competitive baseline (30.01 BLEU to 30.49 BLEU) with the Stanford Phrasal MT system.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1393–1398,
Seattle, Washington, USA, 18-21 October 2013.

2 Review of prior work

Distributed word representations are useful in NLP applications such as information retrieval (Paşca et al., 2006; Manning et al., 2008), search query expansions (Jones et al., 2006), or representing semantics of words (Reisinger et al., 2010). A number of methods have been explored to train and apply word embeddings using continuous models for language. Collobert et al. (2008) learn embeddings in an unsupervised manner through a contrastive estimation technique. Mnih and Hinton (2008), Morin and Bengio (2005) proposed efficient hierarchical continuous-space models. To systematically compare embeddings, Turian et al. (2010) evaluated improvements they bring to state-of-the-art NLP benchmarks. Huang et al. (2012) introduced global document context and multiple word prototypes. Recently, morphology is explored to learn better word representations through Recursive Neural Networks (Luong et al., 2013).

Bilingual word representations have been explored with hand-designed vector space models (Peirsman and Padó , 2010; Sumita, 2000), and with unsupervised algorithms such as LDA and LSA (Boyd-Graber and Resnik, 2010; Tam et al., 2007; Zhao and Xing, 2006). Only recently have continuous space models been applied to machine translation (Le et al., 2012). Despite growing interest in these models, little work has been done along the same lines to train bilingual distributioned word represenations to improve machine translation. In this paper, we learn bilingual word embeddings which achieve competitive performance on semantic word similarity, and apply them in a practical phrase-based MT system.

3 Algorithm and methods

3.1 Unsupervised training with global context

Our method starts with embedding learning formulations in Collobert et al. (2008). Given a context window c in a document d, the optimization minimizes the following Context Objective for a word w in the vocabulary:

$$J_{CO}^{(c,d)} = \sum_{w^r \in V_R} \max(0, 1 - f(c^w, d) + f(c^{w^r}, d)) \quad (1)$$

Here f is a function defined by a neural network. w^r is a word chosen in a random subset V_R of the vocabulary, and c^{w^r} is the context window containing word w^r. This unsupervised objective function contrasts the score between when the correct word is placed in context with when a random word is placed in the same context. We incorporate the global context information as in Huang et al. (2012), shown to improve performance of word embeddings.

3.2 Bilingual initialization and training

In the joint semantic space of words across two languages, the Chinese word '政府' is expected to be close to its English translation 'government'. At the same time, when two words are not direct translations, e.g. 'lake' and the Chinese word '潭' (deep pond), their semantic proximity could be correctly quantified.

We describe in the next sub-sections the methods to intialize and train bilingual embeddings. These methods ensure that bilingual embeddings retain their translational equivalence while their distributional semantics are improved during online training with a monolingual corpus.

3.2.1 Initialization by MT alignments

First, we use MT Alignment counts as weighting to initialize Chinese word embeddings. In our experiments, we use MT word alignments extracted with the Berkeley Aligner (Liang et al., 2006) [1]. Specifically, we use the following equation to compute starting word embeddings:

$$W_{t\text{-}init} = \sum_{s=1}^{S} \frac{C_{ts} + 1}{C_t + S} W_s \quad (2)$$

In this equation, S is the number of possible target language words that are aligned with the source word. C_{ts} denotes the number of times when word t in the target and word s in the source are aligned in the training parallel text; C_t denotes the total number of counts of word t that appeared in the target language. Finally, Laplace smoothing is applied to this weighting function.

[1] On NIST08 Zh-En training data and data from GALE MT evaluation in the past 5 years

Single-prototype English embeddings by Huang et al. (2012) are used to initialize Chinese embeddings. The initialization readily provides a set (*Align-Init*) of benchmark embeddings in experiments (Section 4), and ensures translation equivalence in the embeddings at start of training.

3.2.2 Bilingual training

Using the alignment counts, we form alignment matrices $A_{en \to zh}$ and $A_{zh \to en}$. For $A_{en \to zh}$, each row corresponds to a Chinese word, and each column an English word. An element a_{ij} is first assigned the counts of when the ith Chinese word is aligned with the jth English word in parallel text. After assignments, each row is normalized such that it sums to one. The matrix $A_{zh \to en}$ is defined similarly. Denote the set of Chinese word embeddings as V_{zh}, with each row a word embedding, and the set of English word embeddings as V_{en}. With the two alignment matrices, we define the Translation Equivalence Objective:

$$J_{TEO\text{-}en \to zh} = \|V_{zh} - A_{en \to zh} V_{en}\|^2 \quad (3)$$

$$J_{TEO\text{-}zh \to en} = \|V_{en} - A_{zh \to en} V_{zh}\|^2 \quad (4)$$

We optimize for a combined objective during training. For the Chinese embeddings we optimize for:

$$J_{CO\text{-}zh} + \lambda J_{TEO\text{-}en \to zh} \quad (5)$$

For the English embeddings we optimize for:

$$J_{CO\text{-}en} + \lambda J_{TEO\text{-}zh \to en} \quad (6)$$

During bilingual training, we chose the value of λ such that convergence is achieved for both J_{CO} and J_{TEO}. A small validation set of word similarities from (Jin and Wu, 2012) is used to ensure the embeddings have reasonable semantics. [2]

In the next sections, 'bilingual trained' embeddings refer to those initialized with MT alignments and trained with the objective defined by Equation 5. 'Monolingual trained' embeddings refer to those intialized by alignment but trained without $J_{TEO\text{-}en \to zh}$.

[2] In our experiments, $\lambda = 50$.

3.3 Curriculum training

We train 100k-vocabulary word embeddings using curriculum training (Turian et al., 2010) with Equation 5. For each curriculum, we sort the vocabulary by frequency and segment the vocabulary by a band-size taken from {5k, 10k, 25k, 50k}. Separate bands of the vocabulary are trained in parallel using minibatch L-BFGS on the Chinese Gigaword corpus [3]. We train 100,000 iterations for each curriculum, and the entire 100k vocabulary is trained for 500,000 iterations. The process takes approximately 19 days on a eight-core machine. We show visualization of learned embeddings overlaid with English in Figure 1. The two-dimensional vectors for this visualization is obtained with t-SNE (van der Maaten and Hinton, 2008). To make the figure comprehensible, subsets of Chinese words are provided with reference translations in boxes with green borders. Words across the two languages are positioned by the semantic relationships implied by their embeddings.

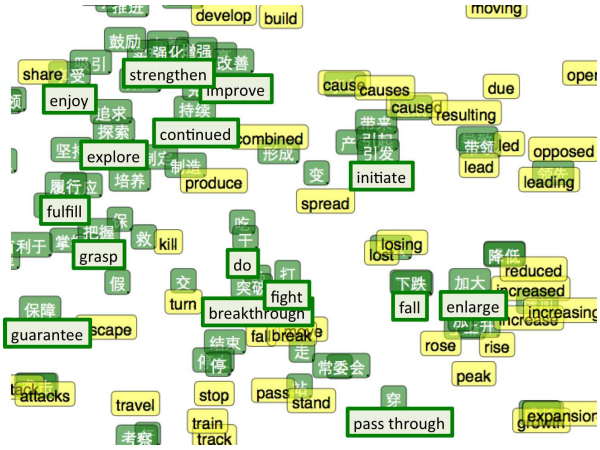


Figure 1: Overlaid bilingual embeddings: English words are plotted in yellow boxes, and Chinese words in green; reference translations to English are provided in boxes with green borders directly below the original word.

4 Experiments

4.1 Semantic Similarity

We evaluate the Mandarin Chinese embeddings with the semantic similarity test-set provided by the or-

[3] Fifth Edition. LDC catelog number *LDC2011T13*. We only exclude *cna_cmn*, the Traditional Chinese segment of the corpus.

Table 1: Results on Chinese Semantic Similarity

Method	*Sp. Corr.* ($\times$100)	*K. Tau* ($\times$100)
Prior work (Jin and Wu, 2012)		5.0
Tf-idf		
Naive tf-idf	41.5	28.7
Pruned tf-idf	46.7	32.3
Word Embeddings		
Align-Init	52.9	37.6
Mono-trained	59.3	42.1
Biling-trained	**60.8**	**43.3**

Table 2: Results on Named Entity Recognition

Embeddings	*Prec.*	*Rec.*	*F1*	*Improve*
Align-Init	0.34	0.52	0.41	
Mono-trained	0.54	0.62	**0.58**	**0.17**
Biling-trained	0.48	0.55	0.52	0.11

Table 3: Vector Matching Alignment *AER* (lower is better)

Embeddings	*Prec.*	*Rec.*	*AER*
Mono-trained	0.27	0.32	0.71
Biling-trained	0.37	0.45	**0.59**

ganizers of *SemEval-2012* Task 4. This test-set contains 297 Chinese word pairs with similarity scores estimated by humans.

The results for semantic similarity are shown in Table 1. We show two evaluation metrics: Spearman Correlation and Kendall's Tau. For both, bilingual embeddings trained with the combined objective defined by Equation 5 perform best. For pruned tf-idf, we follow Reisinger et al. (2010; Huang et al. (2012) and count word co-occurrences in a 10-word window. We use the best results from a range of pruning and feature thresholds to compare against our method. The bilingual and monolingual trained embeddings[4] out-perform pruned *tf-idf* by 14.1 and 12.6 Spearman Correlation ($\times$100), respectively. Further, they out-perform embeddings initialized from alignment by 7.9 and 6.4. Both our *tf-idf* implementation and the word embeddings have significantly higher Kendall's Tau value compared to Prior work (Jin and Wu, 2012). We verified Tau calculations with original submissions provided by the authors.

4.2 Named Entity Recognition

We perform NER experiments on OntoNotes (v4.0) (Hovy et al., 2006) to validate the quality of the Chinese word embeddings. Our experimental setup is the same as Wang et al. (2013). With embeddings, we build a naive feed-forward neural network (Collobert et al., 2008) with 2000 hidden neurons and a sliding window of five words. This naive setting, without sequence modeling or sophisticated join optimization, is not competitive with state-of-the-art (Wang et al., 2013). Table 2 shows that the bilingual embeddings obtains 0.11 F1 improvement, lagging monolingual, but significantly better than *Align-Init* (as in Section3.2.1) on the NER task.

4.3 Vector matching alignment

Translation equivalence of the bilingual embeddings is evaluated by naive word alignment to match word embeddings by cosine distance.[5] The Alignment Error Rates (AER) reported in Table 3 suggest that bilingual training using Equation 5 produces embeddings with better translation equivalence compared to those produced by monolingual training.

4.4 Phrase-based machine translation

Our experiments are performed using the Stanford Phrasal phrase-based machine translation system (Cer et al., 2010). In addition to NIST08 training data, we perform phrase extraction, filtering and phrase table learning with additional data from GALE MT evaluations in the past 5 years. In turn, our baseline is established at 30.01 BLEU and reasonably competitive relative to NIST08 results. We use Minimum Error Rate Training (MERT) (Och, 2003) to tune the decoder.

In the phrase-based MT system, we add one feature to bilingual phrase-pairs. For each phrase, the word embeddings are averaged to obtain a feature vector. If a word is not found in the vocabulary, we disregard and assume it is not in the phrase; if no word is found in a phrase, a zero vector is assigned

[4]Due to variations caused by online minibatch L-BFGS, we take embeddings from five random points out of last 10^5 minibatch iterations, and average their semantic similarity results.

[5]This is evaluated on 10,000 randomly selected sentence pairs from the MT training set.

Table 4: NIST08 Chinese-English translation *BLEU*

Method	*BLEU*
Our baseline	30.01
Embeddings	
Random-Init Mono-trained	30.09
Align-Init	30.31
Mono-trained	30.40
Biling-trained	**30.49**

to it. We then compute the cosine distance between the feature vectors of a phrase pair to form a semantic similarity feature for the decoder.

Results on NIST08 Chinese-English translation task are reported in Table 4[6]. An increase of 0.48 BLEU is obtained with semantic similarity with bilingual embeddings. The increase is modest, just surpassing a reference standard deviation 0.29 BLEU Cer et al. (2010)[7] evaluated on a similar system. We intend to publish further analysis on statistical significance of this result as an appendix. From these suggestive evidence in the MT results, random initialized monolingual trained embeddings add little gains to the baseline. Bilingual initialization and training seem to be offering relatively more consistent gains by introducing translational equivalence.

5 Conclusion

In this paper, we introduce bilingual word embeddings through initialization and optimization constraint using MT alignments The embeddings are learned through curriculum training on the Chinese Gigaword corpus. We show good performance on Chinese semantic similarity with bilingual trained embeddings. When used to compute semantic similarity of phrase pairs, bilingual embeddings improve NIST08 end-to-end machine translation results by just below half a BLEU point. This implies that semantic embeddings are useful features for improving MT systems. Further, our results offer suggestive evidence that bilingual word embeddings act as high-quality semantic features and embody bilingual translation equivalence across languages.

[6]We report case-insensitive BLEU

[7]With 4-gram BLEU metric from Table 4

Acknowledgments

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, or the US government. We thank John Bauer and Thang Luong for helpful discussions.

References

A. Klementiev, I. Titov and B. Bhattarai. 2012. Inducing Crosslingual Distributed Representation of Words. *COLING.*

Y. Bengio, J. Louradour, R. Collobert and J. Weston. 2009. Curriculum Learning. *ICML.*

Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research.*

Y. Bengio and Y. LeCunn. 2007. Scaling learning algorithms towards AI. *Large-Scale Kernal Machines.*

J. Boyd-Graber and P. Resnik. 2010. Holistic sentiment analysis across languages: multilingual supervised latent dirichlet allocation. *EMNLP.*

D. Cer, M. Galley, D. Jurafsky and C. Manning. 2010. Phrasal: A Toolkit for Statistical Machine Translation with Facilities for Extraction and Incorporation of Arbitrary Model Features. *In Proceedings of the North American Association of Computational Linguistics - Demo Session (NAACL-10).*

D. Cer, C. Manning and D. Jurafsky. 2010. The Best Lexical Metric for Phrase-Based Statistical MT System Optimization. *NAACL.*

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *ICML.*

G. Foster and R. Kuhn. 2009. Stabilizing minimum error rate training. *Proceedings of the Fourth Workshop on Statistical Machine Translation.*

M. Galley, P. Chang, D. Cer, J. R. Finkel and C. D. Manning. 2008. NIST Open Machine Translation 2008 Evaluation: Stanford University's System Description. *Unpublished working notes of the 2008 NIST Open Machine Translation Evaluation Workshop.*

S. Green, S. Wang, D. Cer and C. Manning. 2013. Fast and adaptive online training of feature-rich translation models. *ACL.*

G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath

and B. Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine.*

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw and R. Weischedel. 2006. OntoNotes: the 90% solution. *NAACL-HLT.*

E. H. Huang, R. Socher, C. D. Manning and A. Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. *ACL.*

P. Jin and Y. Wu. 2012. SemEval-2012 Task 4: Evaluating Chinese Word Similarity. *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. Association for Computational Linguistics.*

R. Jones. 2006. Generating query substitutions. *In Proceedings of the 15th international conference on World Wide Web.*

P. Koehn, F. J. Och and D. Marcu. 2003. Statistical Phrase-Based Translation. *HLT.*

H. Le, A. Allauzen and F. Yvon 2012. Continuous space translation models with neural networks. *NAACL.*

P. Liang, B. Taskar and D. Klein. 2006. Alignment by agreement. *NAACL.*

M. Luong, R. Socher and C. Manning. 2013. Better word representations with recursive neural networks for morphology. *CONLL.*

L. van der Maaten and G. Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research.*

A. Maas and R. E. Daly and P. T. Pham and D. Huang and A. Y. Ng and C. Potts. 2011. Learning word vectors for sentiment analysis. *ACL.*

C. Manning and P. Raghavan and H. Schtze. 2008. Introduction to Information Retrieval. *Cambridge University Press, New York, NY, USA.*

T. Mikolov, M. Karafiat, L. Burget, J. Cernocky and S. Khudanpur. 2010. Recurrent neural network based language model. *INTERSPEECH.*

T. Mikolov, K. Chen, G. Corrado and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781v1.*

A. Mnih and G. Hinton. 2008. A scalable hierarchical distributed language model. *NIPS.*

F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. *AISTATS.*

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

M. Paşca, D. Lin, J. Bigham, A. Lifchits and A. Jain. 2006. Names and similarities on the web: fact extraction in the fast lane. *ACL.*

Y. Peirsman and S. Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. *ACL.*

J. Reisinger and R. J. Mooney. 2010. Multi-prototype vector-space models of word meaning. *NAACL.*

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv., 34:1-47, March.*

R. Socher, J. Pennington, E. Huang, A. Y. Ng and C. D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. *EMNLP.*

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. *NIPS.*

E. Sumita. 2000. Lexical transfer using a vector-space model. *ACL.*

Y. Tam, I. Lane and T. Schultz. 2007. Bilingual-LSA based LM adaptation for spoken language translation. *ACL.*

S. Tellex and B. Katz and J. Lin and A. Fernandes and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. *In Proceedings of the 26th Annual International ACM SIGIR Conference on Search and Development in Information Retrieval, pages 41-47. ACM Press.*

J. Turian and L. Ratinov and Y. Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. *ACL.*

M. Wang, W. Che and C. D. Manning. 2013. Joint Word Alignment and Bilingual Named Entity Recognition Using Dual Decomposition. *ACL.*

K. Yamada and K. Knight. 2001. A Syntax-based Statistical Translation Model. *ACL.*

B. Zhao and E. P. Xing 2006. BiTAM: Bilingual topic AdMixture Models for word alignment. *ACL.*

Application of Localized Similarity for Web Documents

Peter Reberšek
Zemanta
Celovška cesta 32
Ljubljana, Slovenia
peter.rebersek@zemanta.com

Mateja Verlič
Zemanta
Celovška cesta 32
Ljubljana, Slovenia
mateja.verlic@zemanta.com

Abstract

In this paper we present a novel approach to automatic creation of anchor texts for hyperlinks in a document pointing to similar documents. Methods used in this approach rank parts of a document based on the similarity to a presumably related document. Ranks are then used to automatically construct the best anchor text for a link inside original document to the compared document. A number of different methods from information retrieval and natural language processing are adapted for this task. Automatically constructed anchor texts are manually evaluated in terms of relatedness to linked documents and compared to baseline consisting of originally inserted anchor texts. Additionally we use crowdsourcing for evaluation of original anchors and automatically constructed anchors. Results show that our best adapted methods rival the precision of the baseline method.

1 Introduction

One of the features of hypertext documents are hyperlinks that point to other resources – pictures, videos, tweets, or other hypertext documents. A fairly familiar category of the latter is *related articles*; these usually appear at the end of a news article or a blog post with the title of the target document as anchor text. The target document is similar in content to original document; it may tell the story from another point of view, it may be a more detailed version of a part of the events in the original document, etc. Another category are the in-text links; these appear inside the main body of text and use some of the existing text as anchor. Ideally the anchor text is selected in such a way that it conveys some information about the target document; in reality sometimes just an adverb (e.g. here, there) is used, or even the destination URL may serve as anchor.

Our goal is to develop a system that automatically constructs in-text links, i.e. for a query document finds a target document and an appropriate part of the text of the query document that serves as the anchor text for the hyperlink. We want the target document to be similar in content to the query document and the anchor text to indicate that content.

There are many potential uses for such a system, especially for simplifying and streamlining document creation. This includes authors of blogs that may use the system for adding related content from other sources without exhausting manual search for such material. It may also be used when writing a scientific paper, automatically adding citations to other relevant papers inside the main body. This accelerates the writing, again reducing the time spent searching for possible existing research in the field. A citation can be considered an in-text link without a defined starting point.

We have addressed the problem in two steps, separately finding a similar document, and finding the anchor text for it. Since the retrieval of similar documents was a research focus for many years and is thus better researched, we have decided in this paper to focus on the placement of the anchor text for a link to a preselected document.

This paper is organized as follows: related work is discussed in Section 2, the methods, corpus, and evaluation are described in Section 3, followed by

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1399–1404,
Seattle, Washington, USA, 18-21 October 2013.

results and discussion in Section 4 and ending with conclusions in Section 5.

2 Related Work

Semantic similarity of textual documents offers a way to organize the increasing number of available documents. It can be used in many applications such as summarization, educational systems, finding duplicated bug reports in software testing (Lintean et al., 2010), plagiarism detection (Kasprzak and Brandejs, 2010), and research of a scientific field (Koberstein and Ng, 2006). Documents can vary in length from microblogs (Twitter) and sentences (Li et al., 2006; Koberstein and Ng, 2006) to paragraphs (Lintean et al., 2010) and larger documents (Budanitsky and Hirst, 2006).

There is also commercial software such as nRelate[1], Zemanta[2] and OpenCalais[3] with functionality that ranges from named entity recognition (NER) and event detection to related content. Publishers use in-house tools that offer automatic retrieval of in-house similar documents.

Most of the methods for comparing documents focus on the query document as a whole. The calculated score therefore belongs to the whole document and nothing can be said about more or less similar parts of the document. Our goal is to localize the similarity to a part of the query document, a paragraph, sentence, or even a part of the sentence that is most similar to another document. This part of the query document can then serve as anchor text for a hyperlink connection to the similar document.

Plagiarism detection methods (Alzahrani et al., 2012; Monostori et al., 2002) have a task of verifying the originality of the document. Extrinsic plagiarism detection methods compare two documents to determine if some of the material in one is plagiarised from the other. Methods range from simple exact substring matching to more advanced ones like semantic based methods that are able to recognize paraphrasing and refactoring (Alzahrani et al., 2012). These methods have localization of similarity already built-in as they are searching for parts of the text that seem to be plagiarised. We have focused on one such method, the winner of the PAN 2010 challenge (Kasprzak and Brandejs, 2010). This method uses shared n-grams from the two documents in order to determine if one of them is plagiarised.

Another similar research is automatic citation placement for scientific papers. Most of the work (Strohman et al., 2007; McNee et al., 2002) is concerned with putting citations at the end of the paper (non-localized), which is a task similar to inserting related articles for a news article at the of the text. There have been some attempts to place the citations in the main body of text (Tang and Zhang, 2009; He et al., 2011), typically used when referring to an idea or method.

Tang and Zang (2009) used a placeholder constraint: the query document must contain placeholders for citations, i.e. the places in text where citation might be inserted. Their method then just ranks all possible documents for a particular placeholder and chooses the best ranked document as a result. Documents are ranked on the basis of a learned topic model, obtained by a two-layer Restricted Boltzmann Machine.

He et al.(2011) made a step further towards generality of a citation location; they divide the text into overlapping windows and then decide which windows are viable citation context. The best method for deciding which citation context to use was a dependency feature model, an ensemble method using 17 different features and decision trees.

Named entity recognition (NER) also offers a useful insight into document similarity. If two documents share a named entity (NE), it is more likely they are similar. Detected NEs may also serve as anchor text for the link. NER is a fairly researched field (Finkel et al., 2005; Ratinov et al., 2011; Bunescu and Pasca, 2006; Kulkarni et al., 2009; Milne and Witten, 2008) and is also used in several commercial applications such as Zemanta, OpenCalais and AlchemyAPI[4], which are able to automatically insert links for a NE pointing to a knowledge base such as Wikipedia or IMDB. However, at this point they are unable to link to arbitrary documents, but may be useful in conjunction with other methods.

[1] nRelate: `http://www.nrelate.com/`

[2] Zemanta: `http://www.zemanta.com/`

[3] OpenCalais: `http://www.opencalais.com/`

[4] AlchemyAPI: `http://www.alchemyapi.com/`

3 Methodology

3.1 Corpus

We have chosen 100 web articles (posts) at random from the end of January 2012. We extracted the body and title of each document. All the present in-text links were also extracted and filtered. First, automatic filtering was applied to remove unwanted categories of links (videos, definition pages on wikipedia and imdb, etc.), and articles that were deemed too short for similarity comparison. The threshold was set at 200 words of automatically scraped body text of a linked document.

All the remaining links were manually checked to ensure the integrity of link targets. This way we collected 265 articles (hereinafter *related articles* - RA). A number of different methods were then used to calculate similarity rank and select the best part of the post text to be used as anchor text for a hyperlink pointing to the originally linked RA.

We have used CrowdFlower[5], a crowdsourcing platform, to evaluate how many of the 265 post–RA pairs were really related; the final corpus thus consisted of 236 pairs.

3.2 Evaluation

We have used each of the methods described in Subsection 3.3 to automatically construct anchor text for each of the 236 pairs of documents in the final corpus. If a method could not find a suitable anchor, no result was returned; on average there were 147 anchors per method. All the automatically created links were then manually scored by the authors with an in-house evaluation tool using scores and guidelines summarized in Table 1. To calculate precision and recall, we have counted scores 2 and 3 as positive result.

Additionally we crowdsourced the evaluation of results for some of the methods. For this task we prepared a special description of evaluation tasks and defined a set of questions for collecting results. We provided simplified guidelines for assigning scores to automatically created anchors and set a confidence threshold of 0.55 for an assignment to be considered valid. It is important to mention that the use of crowdsourcing for such tasks has to be carefully planned, because many issues related to monetary incentives, which are out of the scope of this paper, may arise.

[5]CrowdFlower: `http://crowdflower.com/`

Score	Description
0	Anchor does not signify anything about RA or gets it wrong
1	Some connection can be established (anchor is a shared Named Entity, Noun Phrase, Verb Phrase, etc.)
2	Anchor is a good estimation of RA topics, but not wholly (anchor is a non-main topic in RA)
3	RA topics can be directly inferred from the anchor

Table 1: Scores used for internal evaluation of automatically created anchors

3.3 Methods for constructing anchor texts

We have adapted a number of methods from a variety of sources to test how they perform for our exact purpose. Below is a short overview of the different methods used in this work.

3.3.1 Longest chunk

This method is based on natural language processing and extensively uses NLTK package (Bird et al., 2009); the text is first tokenized with the default NLTK tokenizer, and then POS tagged with one of the included POS taggers. After much testing, we have decided on a combination of Brill – Trigram – Bigram – Unigram – Affix – Regex backoff tagger with *noun* as default tag. The trainable parts of the tagger were trained on the included CoNLL 2000 tagged corpus.

Before chunking was applied, we also simplified some tags and removed some others to get a simpler structure of POS tags. We then used a regex chunker to find a sequence of a proper noun and a verb separated by zero or more other tokens. We have also tested a proper noun - verb - proper noun combination, but there were even fewer results, so this direction was abandoned.

3.3.2 Latent Semantic Indexing (LSI) based

A corpus is represented in LSI (Deerwester et al., 1990) as a large matrix of term occurrences

in individual documents. The rank of the matrix is then reduced using singular value decomposition that groups together terms that occur in similar context which should therefore account for synonyms.

We have used a tool called gensim (Řehůřek and Sojka, 2010) that enabled us to quickly train a LSI model using the whole corpus and index just the related articles. In order to localize the similarity and place an anchor, we split the source document into paragraphs and compute similarity scores between target document and each paragraph of the source document. We then split the paragraph with the highest score into sentences and again obtain scores for each. The sentence with the best score is then chosen as the result.

3.3.3 Sorted n-grams

Drawing on plagiarism detection, the winning method from the PAN 2010 (Kasprzak and Brandejs, 2010) seemed a viable choice. The basis of the method is comparing n-grams of the source and the destination documents. First, the text was again tokenized with NLTK, removed stopwords and tokens with two or less characters. Then overlapping n-grams were constructed. We have deviated from Kasprzak's merging policy and decided to merge two results if they are less than 20 tokens apart. We also required only one shared n-gram to consider the documents similar. Results were ranked based on the number of shared tokens within each.

3.3.4 Unigrams tf*idf

This method uses unigram tf*idf weighted scores. Since we had a closed system, we used corpus-wide frequencies; stopwords were also removed. We have scored tokens in the source document with tf*idf summary of the destination document; tokens not in summary are given a zero weight. We have experimentally determined that a summary of just top 150 tokens improves results. Sentences were ranked based on the sum of its tokens weights. We also included NEs from Zemanta API response for both source and destination document. Sentences containing shared NEs get their score multiplied by the sum of shared NE tf*idf weights. The result was then the sentence with the highest score.

	Manual		CrowdFlower	
	P	R	P	R
Original links	0.691	0.691	0.981	0.432
Sorted 5-grams	0.822	0.254		
Sorted 4-grams	0.741	0.352		
Sorted 3-grams	0.680	0.424	0.956	0.275
Longest Chunk	0.080	0.075	0.907	0.165
Unigrams tf*idf	0.626	0.242	0.882	0.127
LSI based	0.648	0.640		

Table 2: Precision and recall for manual and CrowdFlower evaluation

3.3.5 Baseline

Our baseline was a method that inserted links that were originally present in the source documents. This method was used to compare our automatic methods to what people are actually linking in the real world.

4 Evaluation Results and Discussion

Results are presented as precision and recall for different methods and both evaluations in Table 2. Empty cells in the table indicate that these methods were not evaluated using CrowdFlower. Recall is the fraction of relevant results out of all the possible results (236) and precision is the fraction of relevant results out of all the retrieved results.

The first thing we notice is the general disagreement between results from the authors and CrowdFlower workers; the latter tend to give higher scores, which leads to higher precision and recall. The reason for this might be in the authors' background knowledge and thus higher expectations.

As a contrast almost half of CrowdFlower workers stated they don't blog and of the rest, more than a third of them don't link out, i.e. do not use related articles. We also have only 74% median inter-annotator agreement leading us to believe that some of the annotators answered without being familiar with the question (monetary incentive issue).

Furthermore, CrowdFlower results for original links (our baseline) indicate that almost all of them were recognized as relevant, while our evaluators discarded 30% of them. Clearly seen in the results of different sorted n-grams methods is also the precision-recall trade-off.

5 Conclusion

Based on evaluation results and despite differences between the evaluators with background knowledge and the crowds, we can conclude that that our approach for automatic construction of in-text links rivals manual creation by professional writers and bloggers and is thus a promising direction for further research.

Acknowledgement

This work was partially funded by the Slovenian Ministry of Higher Education, Science and Technology, and the European Union – European Regional Development Fund.

References

S.M. Alzahrani, N. Salim, and A. Abraham. 2012. Understanding plagiarism linguistic patterns, textual features, and detection methods. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(2):133–149.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32(1):13–47, March.

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, Trento, Italy.

Scott Deerwester, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra, and C. Lee Giles. 2011. Citation recommendation without author supervision. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 755–764, New York, NY, USA. ACM.

Jan Kasprzak and Michal Brandejs. 2010. Improving the reliability of the plagiarism detection system lab report for pan at clef 2010.

Jonathan Koberstein and Yiu-Kai Ng. 2006. Using word clusters to detect similar web documents. In *Proceedings of the First international conference on Knowledge Science, Engineering and Management*, KSEM'06, pages 215–228, Berlin, Heidelberg. Springer-Verlag.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 457–466, New York, NY, USA. ACM.

Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. on Knowl. and Data Eng.*, 18(8):1138–1150, August.

Mihai Lintean, Cristian Moldovan, Vasile Rus, and Danielle McNamara. 2010. The role of local and global weighting in assessing the semantic similarity of texts using latent semantic analysis. In *Proceedings of the 23rd International Florida Artificial Intelligence Research Society Conference, Daytona Beach, FL.*

Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. 2002. On the recommending of citations for research papers. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, CSCW '02, pages 116–125, New York, NY, USA. ACM.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA. ACM.

Krisztián Monostori, Raphael Finkel, Arkady Zaslavsky, Gábor Hodász, and Máté Pataki. 2002. Comparison of overlap detection techniques. In PeterM.A. Sloot, AlfonsG. Hoekstra, C.J.Kenneth Tan, and JackJ. Dongarra, editors, *Computational Science — ICCS 2002*, volume 2329 of *Lecture Notes in Computer Science*, pages 51–60. Springer Berlin Heidelberg.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Pro-*

ceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, May. ELRA.

Trevor Strohman, W. Bruce Croft, and David Jensen. 2007. Recommending citations for academic papers. In *In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07*, pages 705–706.

Jie Tang and Jing Zhang. 2009. A discriminative approach to topic-based citation recommendation. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu-Bao Ho, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5476 of *Lecture Notes in Computer Science*, pages 572–579. Springer Berlin Heidelberg.

Dependency language models for sentence completion

Joseph Gubbins
Computer Laboratory
University of Cambridge
jsg52@cam.ac.uk

Andreas Vlachos
Computer Laboratory
University of Cambridge
av308@cam.ac.uk

Abstract

Sentence completion is a challenging semantic modeling task in which models must choose the most appropriate word from a given set to complete a sentence. Although a variety of language models have been applied to this task in previous work, none of the existing approaches incorporate syntactic information. In this paper we propose to tackle this task using a pair of simple language models in which the probability of a sentence is estimated as the probability of the lexicalisation of a given syntactic dependency tree. We apply our approach to the Microsoft Research Sentence Completion Challenge and show that it improves on n-gram language models by 8.7 percentage points, achieving the highest accuracy reported to date apart from neural language models that are more complex and expensive to train.

1 Introduction

The verbal reasoning sections of standardised tests such as the Scholastic Aptitude Test (SAT) feature problems where a partially complete sentence is given and the candidate must choose the word or phrase from a list of options which completes the sentence in a logically consistent way. Sentence completion is a challenging semantic modelling problem. Systematic approaches for solving such problems require models that can judge the global coherence of sentences. Such measures of global coherence may prove to be useful in various applications, including machine translation and natural language generation (Zweig and Burges, 2012).

Most approaches to sentence completion employ language models which use a window of immediate context around the missing word and choose the word that results in the completed sentence with the highest probability (Zweig and Burges, 2012; Mnih and Teh, 2012). However, such language models may fail to identify sentences that are locally coherent but are improbable due to long-range syntactic/semantic dependencies. Consider, for example, completing the sentence

I saw a tiger which was really very ...

with either *fierce* or *talkative*. A language model relying on up to five words of immediate context would ignore the crucial dependency between the missing word and the noun *tiger*.

In this paper we tackle sentence completion using language models based on dependency grammar. These models are similar to standard n-gram language models, but instead of using the linear ordering of the words in the sentence, they generate words along paths in the dependency tree of the sentence. Unlike other approaches incorporating syntax into language models (e.g., Chelba et al., 1997), our models are relatively easy to train and estimate, and can exploit standard smoothing methods. We apply them to the Microsoft Research Sentence Completion Challenge (Zweig and Burges, 2012) and show an improvement of 8.7 points in accuracy over n-gram models, giving the best results to date for any method apart from the more computationally demanding neural language models.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1405–1410,
Seattle, Washington, USA, 18-21 October 2013.

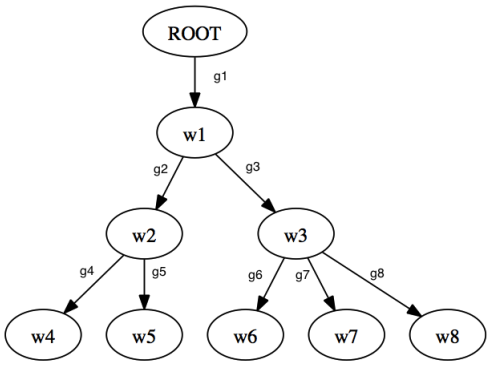


Figure 1: Dependency tree example

2 Unlabelled Dependency Language Models

In dependency grammar, each word in a sentence is associated with a node in a dependency tree (Figure 1). We define a dependency tree as a rooted, connected, acyclic directed graph together with a mapping from the nodes of the tree to a set of grammatical relation labels $\mathcal{R}$. We define a lexicalised dependency tree as a dependency tree along with a mapping from the vertices of the tree to a vocabulary $\mathcal{V}$.

We seek to model the probability distribution of the lexicalisation of a given dependency tree. We will use this as a language model; we neglect the fact that a given lexicalised dependency tree can correspond to more than one sentence due to variations in word order. Let S^T be a lexicalised dependency tree, where T is the unlexicalised tree and let $w_1 w_2 \dots w_m$ be an ordering of the words corresponding to a breadth-first enumeration of the tree. In order for this representation to be unique, when we parse a sentence, we will use the unique breadth-first ordering where the children of any node appear in the same order as they did in the sentence. We define w_0 to be a special symbol denoting the root of the tree. We denote the grammatical relation between w_k and its parent by $g_k \in \mathcal{R}$.
We apply the chain rule to the words in the tree in the order of this breadth-first enumeration:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|(w_k)_{k=0}^{i-1}, T] \tag{1}$$

Given a word w_i, we define the ancestor sequence $A(w)$ to be the subsequence of $(w_k)_{k=0}^{i-1}$ describing the path from the root node to the parent of w, where each element of the sequence is the parent of the next element. For example in Figure 1, $A(w_8) = (w_0, w_1, w_3)$. We make the following two assumptions:

- that each word w_i is conditionally independent of the words outside of its ancestor sequence $(w_k)_{k=0}^{i-1} \cap A(w_i)^c$, given the ancestor sequence $A(w_i)$;
- that the words are independent of the labels $(g_k)_{k=1}^{m}$.

Using these assumptions, we can write the probability as:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|A(w_i)] \tag{2}$$

Given a training data corpus consisting of sentences parsed into dependency trees, the maximum likelihood estimator for the probability $\mathbb{P}[w_i|A(w_i)]$ is given by the proportion of cases where the ancestor sequence $A(w_i)$ was followed by w_i. Let $C(\cdot)$ be the count of the number of observations of a pattern in the corpus. We have

$$\hat{\mathbb{P}}[w_i|A(w_i)] = \frac{C((A(w_i), w_i))}{\sum_{w \in \mathcal{V}} C((A(w_i), w))} \tag{3}$$

As is the case for n-gram language models, we can't hope to observe all possible sequences of words no matter how big the corpus. To deal with this data sparsity issue, we take inspiration from n-gram models and assume a Markov property of order $(N-1)$:

$$\mathbb{P}[w|A(w)] = \mathbb{P}[w|A^{(N-1)}(w)] \tag{4}$$

where $A^{(N-1)}(w)$ denotes the sequence of up to $(N-1)$ closest ancestors of w.

The maximum likelihood estimator for this probability is:

$$\hat{\mathbb{P}}[w_i|A^{(N-1)}(w_i)] = \frac{C((A^{(N-1)}(w_i), w_i))}{\sum_{w \in \mathcal{V}} C((A^{(N-1)}(w_i), w))}$$

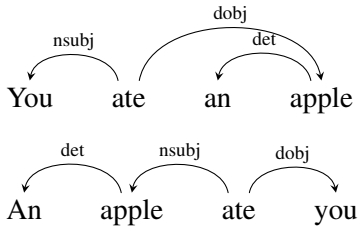
We have arrived at a model which is quite similar to n-gram language models. The main difference

is that each word in the tree can have several children, while in the n-gram models it can only be followed by one word. Thus the sum in the denominator above does not simplify to the count of the ancestor sequence in the way that it does for n-gram language models. However, we can calculate and store the denominators easily during training, so that we do not need to sum over the vocabulary each time we evaluate the estimator. We refer to this model as the **order N unlabelled dependency language model**.

As is the case for n-gram language models, even for low values of N, we will often encounter sequences $(A^{(N-1)}(w), w)$ which were not observed in training. In order to avoid assigning zero probability to the entire sentence, we need to use a smoothing method. We can use any of the smoothing methods used for n-gram language models. For simplicity, we use stupid backoff smoothing (Brants et al., 2007).

3 Labelled Dependency Language Models

We assumed above that the words are generated independently from the grammatical relations. However, we are likely to ignore valuable information in doing so. To illustrate this point, consider the following pair of sentences:



The dependency trees of the two sentences are very similar, with only the grammatical relations between *ate* and its arguments differing. The unlabelled dependency language model will assign the same probability to both of the sentences as it ignores the labels of grammatical relations. In order to be able to distinguish between them, the nature of the grammatical relations between the words in the dependency tree needs to be incorporated in the language model. We relax the assumption that the words are independent of the labels of the parse tree, assuming instead the each word is conditionally independent of the words and labels outside its ancestor path given the words and labels in its ancestor path. We define $G(w_i)$ to be the sequence of grammatical relations between the successive elements of $(A(w_i), w_i)$. $G(w_i)$ is the sequence of grammatical relations found on the path from the root node to w_i. For example, in Figure 1, $G(w_8) = (g_1, g_3, g_8)$. With our modified assumption we have:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|A(w_i), G(w_i)] \qquad (5)$$

Once again we apply a Markov assumption. Let $G^{(N-1)}(w)$ be the sequence of grammatical relations between successive elements of $(A^{(N-1)}(w), w)$. With an $(N-1)^{th}$ order Markov assumption, we have:

$$\mathbb{P}[S^T|T] = \prod_{i=1}^{m} \mathbb{P}[w_i|A^{(N-1)}(w_i), G^{(N-1)}(w_i)]$$

The maximum likelihood estimator for the probability is once again given by the ratio of the counts of labelled paths. We refer to this model as the **order N labelled dependency language model**.

4 Dataset and Implementation Details

We carried out experiments using the Microsoft Research Sentence (MSR) Completion Challenge (Zweig and Burges, 2012). This consists of a set of 1,040 sentence completion problems taken from five of the Sherlock Holmes novels by Arthur Conan Doyle. Each problem consists of a sentence in which one word has been removed and replaced with a blank and a set of 5 candidate words to complete the sentence. The task is to choose the candidate word which, when inserted into the blank, gives the most probable complete sentence. The set of candidates consists of the original word and 4 imposter words with similar distributional statistics. Human judges were tasked with choosing imposter words which would lead to grammatically correct sentences and such that, with some thought, the correct answer should be unambiguous. The training data set consists of 522 19th century novels from Project Gutenberg. We parsed the training data using the Nivre arc-eager deterministic dependency parsing algorithm (Nivre and Scholz, 2004) as implemented in MaltParser (Nivre et al., 2006). We trained order N labelled and unabelled dependency

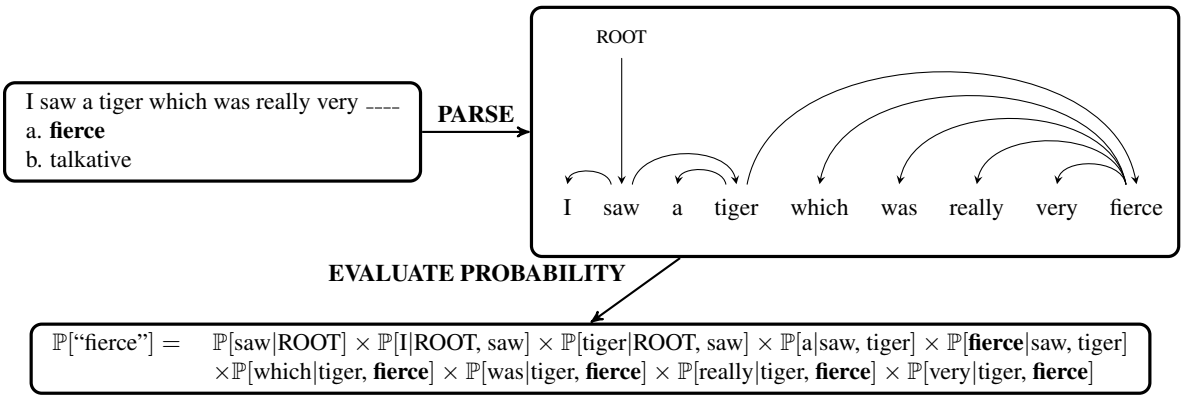


Figure 2: Procedure for evaluating sentence completion problems

N	Unlab-SB	Lab-SB	Ngm-SB	Ngm-KN
2	43.2%	43.0%	28.1%	27.8%
3	48.3%	49.8%	38.5%	38.4%
4	48.3%	**50.0%**	40.8%	41.1%
5	47.4%	49.9%	41.3%	40.8%

Table 1: Summary of results for Sentence Completion

Method	Accuracy
n-grams (Various)	39% - 41%
Skip-grams (Mikolov)	48%
Unlabelled Dependency Model	**48.3%**
Average LSA (Zweig)	49%
Labelled Dependency Model	**50.0%**
Log-bilinear Neural LM (Mnih)	54.7%
Recurrent Neural LM (Mikolov)	55.4%

Table 2: Comparison against previous results

language models for $2 \leq N \leq 5$. Words which occured fewer than 5 times were excluded from the vocabulary. In order to have a baseline to compare against, we also trained n-gram language models with Kneser-Ney smoothing and stupid backoff using the Berkeley Language Modeling Toolkit (Pauls and Klein, 2011).

To test a given language model, we calculated the scores it assigned to each candidate sentence and chose the completion with the highest score. For the dependency language models we parsed the sentence with each of the 5 possible completions and calculated the probability in each case. Figure 2 illustrates an example of this process for the order 3 unlabelled model.

5 Results

Table 1 summarises the results. **Unlab-SB** is the order N unlabelled dependency language model with Stupid Backoff, **Lab-SB** is the order N labelled dependency language model with Stupid Backoff, **Ngm-SB** is the n-gram language model with Stupid Backoff and **Ngm-KN** is the interpolated Kneser-Ney smoothed n-gram language model.

Both of the dependency language models outperfomed the n-gram language models by a substantial margin for all orders considered. The best result was achieved by the order 4 labelled dependency model which is 8.7 points in accuracy better than the best n-gram model. Furthermore, the labelled dependency models outperformed their unlabelled counterparts for every order except 2.

Comparing against previous work (Table 2), the performance of our n-gram baseline is slightly better than the accuracy reported by other authors (Mnih and Teh, 2012; Zweig et al., 2012) for models of this type. The performance of the labelled dependency language model is superior to the results reported for any single model method, apart from those relying on neural language models (Mnih and Teh, 2012; Mikolov et al., 2013) . However the superior performance of neural networks comes at the cost of long training times. The best result achieved in Zweig et al. (2012) using a single method was 49% accuracy with a method based on LSA. Mikolov et al. (2013) also reported accuracy of 48% for a method called skip-grams, which uses a log-linear classifier to predict which words will appear close to each other in sentences.

6 Related Work and Discussion

The best-known language model based on dependency parsing is that of Chelba et al. (1997). This model writes the probability in the familiar left-to-right chain rule decomposition in the linear order of the sentence, conditioning the probability of the next word on the linear trigram context, as well as some part of the dependency graph information relating to the words on its left. The language models we propose are far simpler to train and compute. A somewhat similar model to our unlabelled dependency language model was proposed in Graham and van Genabith (2010). However they seem to have used different probability estimators which ignore the fact that each node in the dependency tree can have multiple children. Other research on syntactic language modelling has focused on using phrase structure grammars (Pauls and Klein, 2012; Charniak, 2001; Roark, 2001; Hall and Johnson, 2003). The linear complexity of deterministic dependency parsing makes dependency language models such as ours more scalable than these approaches.

The most similar task to sentence completion is lexical substitution (McCarthy and Navigli, 2007). The main difference between them is that in the latter the word to be substituted provides a very important clue in choosing the right candidate, while in sentence completion this is not available. Another related task is selectional preference modeling (Séaghdha, 2010; Ritter et al., 2010), where the aim is to assess the plausibility of possible syntactic arguments for a given word.

The dependency language models described in this paper assign probabilities to full sentences. Language models which require full sentences can be used in automatic speech recognition (ASR) and machine translation (MT). The approach is to use a conventional ASR or MT decoder to produce an N-best list of the most likely candidate sentences and then re-score these with the language model. This was done by Chelba et al. (1997) for ASR using a dependency language model and by Pauls and Klein (2011) for MT using a PSG-based syntactic language model.

7 Conclusion

We have proposed a pair of language models which are probabilistic models for the lexicalisation of a given dependency tree. These models are simple to train and evaluate and are scalable to large data sets. We applied them to the Microsoft Research Sentence Completion Challenge. They performed substantially better than n-gram language models, achieving the best result reported for any single method except for the more expensive and complex to train neural language models.

Acknowledgments

Andreas Vlachos is funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 270019 (SPACEBOOK project www.spacebook-project.eu). The authors would like to thank Dr. Stephen Clark for his helpful comments.

References

Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867. Association for Computational Linguistics.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131. Association for Computational Linguistics.

Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, et al. 1997. Structure and performance of a dependency language model. In *Proceedings of Eurospeech*, volume 5, pages 2775–2778.

Yvette Graham and Josef van Genabith. 2010. Deep syntax language models and statistical machine translation. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 118–126. Coling 2010 Organizing Committee, August.

Keith Hall and Mark Johnson. 2003. Language modeling using efficient best-first bottom-up parsing. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 507–512. IEEE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Andriy Mnih and Yee W Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 64. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Adam Pauls and Dan Klein. 2011. Faster and Smaller N-Gram Language Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 258–267. Association for Computational Linguistics.

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 959–968. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, et al. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Diarmuid O Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics.

Geoffrey Zweig and Chris JC Burges. 2012. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 29–36. Association for Computational Linguistics.

Geoffrey Zweig, John C Platt, Christopher Meek, Christopher JC Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 601–610. Association for Computational Linguistics.

A Walk-based Semantically Enriched Tree Kernel Over Distributed Word Representations

Shashank Srivastava[1] **Dirk Hovy**[2] **Eduard Hovy**[1]
(1) Carnegie Mellon University, Pittsburgh
(2) Center for Language Technology, University of Copenhagen, Denmark
{ssrivastava,hovy}@cmu.edu, mail@dirkhovy.com

Abstract

In this paper, we propose a walk-based graph kernel that generalizes the notion of tree-kernels to continuous spaces. Our proposed approach subsumes a general framework for word-similarity, and in particular, provides a flexible way to incorporate distributed representations. Using vector representations, such an approach captures both distributional semantic similarities among words as well as the structural relations between them (encoded as the structure of the parse tree). We show an efficient formulation to compute this kernel using simple matrix operations. We present our results on three diverse NLP tasks, showing state-of-the-art results.

1 Introduction

Capturing semantic similarity between sentences is a fundamental issue in NLP, with applications in a wide range of tasks. Previously, tree kernels based on common substructures have been used to model similarity between parse trees (Collins and Duffy, 2002; Moschitti, 2004; Moschitti, 2006b). These kernels encode a high number of latent syntactic features within a concise representation, and compute the similarity between two parse trees based on the matching of node-labels (words, POS tags, etc.), as well as the overlap of tree structures. While this is sufficient to capture *syntactic similarity*, it does not capture *semantic similarity* very well, even when using discrete semantic types as node labels. This constrains the utility of many traditional tree kernels in two ways: i) two sentences that are syntactically identical, but have no semantic similarity can receive a high matching score (see Table 1, top) while ii) two sentences with only local syntactic overlap, but high semantic similarity can receive low scores (see Table 1, bottom).

tree pairs	semantic	syntactic	score
love(we, toys(green)) — crush(they, puppies(little))	×	✓	high
kissed(she, cat(her)) — gave(she, friend, kiss(her, feline, a))	✓	×	low

Table 1: Traditional tree kernels do not capture semantic similarity

In contrast, distributional vector representations of words have been successful in capturing fine-grained semantics, but lack syntactic knowledge. Resources such as Wordnet, dictionaries and ontologies that encode different semantic perspectives can also provide additional knowledge infusion.

In this paper, we describe a generic walk-based graph kernel for dependency parse trees that subsumes general notions of word-similarity, while focusing on vector representations of words to capture lexical semantics. Through a convolutional framework, our approach takes into account the distributional semantic similarities between words in a sentence as well as the structure of the parse tree. Our main contributions are:

1. We present a new graph kernel for NLP that extends to distributed word representations, and diverse word similarity measures.

2. Our proposed approach provides a flexible framework for incorporating both syntax and semantics of sentence level constructions.

3. Our generic kernel shows state-of-the-art performance on three eclectic NLP tasks.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1411–1416,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Tree kernels in NLP Tree kernels have been extensively used to capture syntactic information about parse trees in tasks such as parsing (Collins and Duffy, 2002), NER (Wang et al., 2010; Cumby and Roth, 2003), SRL (Moschitti et al., 2008) and relation extraction (Qian et al., 2008). These kernels are based on the paradigm that parse trees are similar if they contain many common substructures, consisting of nodes with identical labels (Vishwanathan and Smola, 2003; Collins and Duffy, 2002). Moschitti (2006a) proposed a partial tree kernel that adds flexibility in matching tree substructures. Croce et al. (2011) introduce a lexical semantic tree kernel that incorporates continuous similarity values between node labels, albeit with a different focus than ours and would not match words with different POS. This would miss the similarity of "feline friend" and "cat" in our examples, as it requires matching the adjective "feline" with "cat", and verb "kissed" with "kiss".

Walk based kernels Kernels for structured data derive from the seminal Convolution Kernel formalism by Haussler (1999) for designing kernels for structured objects through local decompositions. Our proposed kernel for parse trees is most closely associated with the random walk-based kernels defined by Gartner et al. (2003) and Kashima et al. (2003). The walk-based graph kernels proposed by Gartner et al. (2003) count the common walks between two input graphs, using the adjacency matrix of the product graph. This work extends to graphs with a finite set of edge and node labels by appropriately modifying the adjacency matrix. Our kernel differs from these kernels in two significant ways: (i) Our method extends beyond label matching to continuous similarity metrics (this conforms with the very general formalism for graph kernels in Vishwanathan et al. (2010)). (ii) Rather than using the adjacency matrix to model edge-strengths, we modify the product graph and the corresponding adjacency matrix to model node similarities.

3 Vector Tree Kernels

In this section, we describe our kernel and an algorithm to compute it as a simple matrix multiplication formulation.

3.1 Kernel description

The similarity kernel K between two dependency trees can be defined as:

$$K(T_1, T_2) = \sum_{\substack{h_1 \subseteq T_1, h_2 \subseteq T_2 \\ len(h_1)=len(h_2)}} k(h_1, h_2)$$

where the summation is over pairs of equal length walks h_1 and h_2 on the trees T_1 and T_2 respectively. The similarity between two n length walks, $k(h_1, h_2)$, is in turn given by the pairwise similarities of the corresponding nodes ${v_i}^h$ in the respective walks, measured via the node similarity kernel κ:

$$k(h_1, h_2) = \prod_{i:1}^{n} \kappa(v_i^{h_1}, v_i^{h_2})$$

In the context of parse trees, nodes $v_i^{h_1}$ and $v_i^{h_2}$ correspond to words in the two parse trees, and thus can often be conveniently represented as vectors over distributional/dependency contexts. The vector representation allows us several choices for the node kernel function κ. In particular, we consider:

1. Gaussian : $\kappa(v_1, v_2) = \exp\big(-\frac{\|v_1 - v_2\|^2}{2\sigma^2}\big)$
2. Positive-Linear: $\kappa(v_1, v_2) = \max(v_1^T v_2, 0)$
3. Sigmoid: $\kappa(v_1, v_2) = \big(1 + \tanh(\alpha v_1^T v_2)\big)/2$

We note that the kernels above take strictly non-negative values in $[0, 1]$ (assuming word vector representations are normalized). Non-negativity is necessary, since we define the walk kernel to be the product of the individual kernels. As walk kernels are products of individual node-kernels, boundedness by 1 ensures that the kernel contribution does not grow arbitrarily for longer length walks.

The kernel function K puts a high similarity weight between parse trees if they contain common walks with semantically similar words in corresponding positions. Apart from the Gaussian kernel, the other two kernels are based on the dot-product of the word vector representations. We observe that the positive-linear kernel defined above is *not* a Mercer kernel, since the max operation makes it non-positive semidefinite (PSD). However, this formulation has desirable properties, most significant being that all walks with one or more node-pair mismatches are strictly penalized and add no score to

the tree-kernel. This is a more selective condition than the other two kernels, where mediocre walk combinations could also add small contributions to the score. The sigmoid kernel is also non-PSD, but is known to work well empirically (Boughorbel et al., 2005). We also observe while the summation in the kernel is over equal length walks, the formalism can allow comparisons over different length paths by including self-loops at nodes in the tree.

With a notion of similarity between words that defines the local node kernels, we need computational machinery to enumerate all pairs of walks between two trees, and compute the summation over products in the kernel $K(T_1, T_2)$ efficiently. We now show a convenient way to compute this as a matrix geometric series.

3.2 Matrix Formulation for Kernel Computation

Walk-based kernels compute the number of common walks using the adjacency matrix of the product graph (Gartner et al., 2003). In our case, this computation is complicated by the fact that instead of *counting* common walks, we need to compute a product of node-similarities for each walk. Since we compute similarity scores over nodes, rather than edges, the product for a walk of length n involves $n+1$ factors.

However, we can still compute the tree kernel K as a simple sum of matrix products. Given two trees $T(V, E)$ and $T'(V', E')$, we define a modified product graph $G(V_p, E_p)$ with an additional ghost node u added to the vertex set. The vertex and edge sets for the modified product graph are given as:

$$V_p := \{(v_{i1}, {v_{j1}}') : v_{i1} \in V, {v_{j1}}' \in V'\} \cup u$$

$$E_p := \{((v_{i1}, {v_{j1}}'), (v_{i2}, {v_{j2}}')) : (v_{i1}, v_{i2}) \in E, ({v_{j1}}', {v_{j2}}') \in E'\} \bigcup \{(u, (v_{i1}, {v_{j1}}')) : v_{i1} \in V, {v_{j1}}' \in V'\}$$

The modified product graph thus has additional edges connecting u to all other nodes. In our formulation, u now serves as a starting location for all random walks on G, and a $k+1$ length walk of G corresponds to a pair of k length walks on T and T'. We now define the weighted adjacency matrix W for G, which incorporates the local node kernels.

$$W_{(v_{i1}, {v_{j1}}'), (v_{i2}, {v_{j2}}')} = \begin{cases} 0 : ((v_{i1}, {v_{j1}}'), (v_{i2}, {v_{j2}}')) \notin E_p \\ \kappa(v_{i2}, {v_{j2}}') : otherwise \end{cases}$$

$$W_{u, (v_{i1}, {v_{j1}}')} = \kappa(v_{i1}, {v_{j1}}')$$
$$W_{(v,u)} = 0 \ \ \forall\, v \in V_p$$

There is a straightforward bijective mapping from walks on G starting from u to pairs of walks on T and T'. Restricting ourselves to the case when the first node of a $k+1$ length walk is u, the next k steps allow us to efficiently compute the products of the node similarities along the k nodes in the corresponding k length walks in T and T'. Given this adjacency matrix for G, the sum of values of k length walk kernels is given by the u^{th} row of the $(k+1)^{th}$ exponent of the weighted adjacency matrix (denoted as W^{k+1}). This corresponds to $k+1$ length walks on G starting from u and ending at any node. Specifically, $W_{u,(v_i, v_j')}$ corresponds to the sum of similarities of all common walks of length n in T and T' that end in v_i in T and v_j' in T'. The kernel K for walks upto length N can now be calculated as :

$$K(T, T') = \sum_{i}^{|V_p|} S_{u,i}$$

where

$$S = W + W^2 + ...W^{N+1}$$

We note that in out formulation, longer walks are naturally discounted, since they involve products of more factors (generally all less than unity).

The above kernel provides a similarity measure between any two pairs of dependency parse-trees. Depending on whether we consider directional relations in the parse tree, the edge set E_p changes, while the procedure for the kernel computation remains the same. Finally, to avoid larger trees yielding larger values for the kernel, we normalize the kernel by the number of edges in the product graph.

4 Experiments

We evaluate the Vector Tree Kernel (VTK) on three NLP tasks. We create dependency trees using the FANSE parser (Tratz and Hovy, 2011), and use distribution-based SENNA word embeddings by Collobert et al. (2011) as word representations. These embeddings provide low-dimensional vector

representations of words, while encoding distributional semantic characteristics. We use LibSVM for classification. For sake of brevity, we only report results for the best performing kernel.

We first consider the Cornell Sentence Polarity dataset by Pang and Lee (2005). The task is to identify the polarity of a given sentence. The data consists of 5331 sentences from positive and negative movie reviews. Many phrases denoting sentiments are lexically ambiguous (cf. "*terribly* entertaining" vs "*terribly* written"), so simple lexical approaches are not expected to work well here, while syntactic context could help disambiguation.

Next, we try our approach on the MSR paraphrase corpus. The data contains a training set of 4077 pairs of sentences, annotated as paraphrases and non-paraphrases, and a test-set of 1726 sentence pairs. Each instance consists of a pair of sentences, so the VTK cannot be directly used by a kernel machine for classification. Instead, we generate 16 kernel values based for each pair on different parameter settings of the kernel, and feed these as features to a linear SVM.

We finally look at the annotated Metaphor corpus by (Hovy et al., 2013). The dataset consists of sentences with specified target phrases. The task here is to classify the target use as literal or metaphorical. We focus on target phrases by upweighting walks that pass through target nodes. This is done by simply multiplying the corresponding entries in the adjacency matrix by a constant factor.

5 Results

5.1 Sentence Polarity Dataset

	Prec	Rec	F1	Acc
Albornoz et al	0.63	–	–	0.63
WNA+synsets	0.61	–	–	0.61
WNA	0.53	–	–	0.51
DSM	0.54	0.55	0.55	0.54
SSTK	0.49	0.48	0.48	0.49
VTK	**0.65**	**0.58**	**0.62**	**0.67**

Table 2: Results on Sentence Polarity dataset

On the polarity data set, Vector Tree Kernel (VTK) significantly outperforms the state-of-the-art method by Carrillo de Albornoz et al. (2010), who use a hybrid model incorporating databases of affective lexicons, and also explicitly model the effect of negation and quantifiers (see Table 2). Lexical approaches using pairwise semantic similarity of SENNA embeddings (DSM), as well as Wordnet Affective Database-based (WNA) labels perform poorly (Carrillo de Albornoz et al., 2010), showing the importance of syntax for this particular problem. On the other hand, a syntactic tree kernel (SSTK) that ignores distributional semantic similarity between words, fails as expected.

5.2 MSR Paraphrase Dataset

	Prec	Rec	F1	Acc
BASE	0.72	0.86	0.79	0.69
Zhang et al	0.74	0.88	0.81	0.72
Qiu et al	0.73	0.93	0.82	0.72
Malakasiotis	0.74	0.94	0.83	0.74
Finch	**0.77**	0.90	**0.83**	**0.75**
VTK	0.72	**0.95**	0.82	0.72

Table 3: Results on MSR Paraphrase corpus

On the MSR paraphrase corpus, VTK performs competitively against state-of-the-art-methods. We expected paraphrasing to be challenging to our method, since it can involve little syntactic overlap. However, data analysis reveals that the corpus generally contains sentence pairs with high syntactic similarity. Results for this task are encouraging since ours is a general approach, while other systems use multiple task-specific features like semantic role labels, active-passive voice conversion, and synonymy resolution. In the future, incorporating such features to VTK should further improve results for this task .

5.3 Metaphor Identification

	Acc	P	R	F1
CRF	0.69	**0.74**	0.50	0.59
SVM+DSM	0.70	0.63	0.80	0.71
SSTK	0.75	0.70	0.80	0.75
VTK	**0.76**	0.67	**0.87**	**0.76**

Table 4: Results on Metaphor dataset

On the Metaphor corpus, VTK improves the previous score by Hovy et al. (2013), whose approach uses an conjunction of lexical and syntactic tree kernels (Moschitti, 2006b), and distributional vectors. VTK identified several templates of metaphor usage such as "warm heart" and "cold shoulder". We look towards approaches for automatedly mining such metaphor patterns from a corpus.

6 Conclusion

We present a general formalism for walk-based kernels to evaluate similarity of dependency trees.

Our method generalizes tree kernels to take distributed representations of nodes as input, and capture both lexical semantics and syntactic structures of parse trees. Our approach has tunable parameters to look for larger or smaller syntactic constructs. Our experiments shows state-of-the-art performance on three diverse NLP tasks. The approach can generalize to any task involving structural and local similarity, and arbitrary node similarity measures.

References

Sabri Boughorbel, Jean-Philippe Tarel, and Nozha Boujemaa. 2005. Conditionally positive definite kernels for svm based image recognition. In *ICME*, pages 113–116.

Jorge Carrillo de Albornoz, Laura Plaza, and Pablo Gervás. 2010. A hybrid approach to emotional sentence polarity and intensity classification. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 153–161. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics.

Chad Cumby and Dan Roth. 2003. On kernel methods for relational learning. In *In Proc. of the International Conference on Machine Learning*, pages 107–114.

Andrew Finch. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *In IWP2005*.

Thomas Gartner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 129–143. Springer.

David Haussler. 1999. Convolution kernels on discrete structures. Technical Report Technical Report UCS-CRL-99-10, UC Santa Cruz.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of NAACL HLT, Meta4NLP Workshop*.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press.

Prodromos Malakasiotis. 2009. Paraphrase recognition using machine learning to combine similarity measures. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, ACLstudent '09, pages 27–35, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 335–es. Association for Computational Linguistics.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer.

Alessandro Moschitti. 2006b. Making Tree Kernels Practical for Natural Language Learning. In *In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 18–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1257–1268, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. V. N. Vishwanathan and Alexander J. Smola. 2003. Fast kernels for string and tree matching. In *Advances In Neural Information Processing Systems 15*, pages 569–576. MIT Press.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph kernels. *J. Mach. Learn. Res.*, 99:1201–1242, August.

Xinglong Wang, Jun'ichi Tsujii, and Sophia Ananiadou. 2010. Disambiguating the species of biomedical named entities using natural language parsers. *Bioinformatics*, 26(5):661–667.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *In Proceedings of the Australasian Language Technology Workshop 2005*.

Automatic Idiom Identification in Wiktionary

Grace Muzny and **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{muznyg,lsz}@cs.washington.edu

Abstract

Online resources, such as Wiktionary, provide an accurate but incomplete source of idiomatic phrases. In this paper, we study the problem of automatically identifying idiomatic dictionary entries with such resources. We train an idiom classifier on a newly gathered corpus of over 60,000 Wiktionary multi-word definitions, incorporating features that model whether phrase meanings are constructed compositionally. Experiments demonstrate that the learned classifier can provide high quality idiom labels, more than doubling the number of idiomatic entries from 7,764 to 18,155 at precision levels of over 65%. These gains also translate to idiom detection in sentences, by simply using known word sense disambiguation algorithms to match phrases to their definitions. In a set of Wiktionary definition example sentences, the more complete set of idioms boosts detection recall by over 28 percentage points.

1 Introduction

Idiomatic language is common and provides unique challenges for language understanding systems. For example, a *diamond in the rough* can be the literal unpolished object or a crude but lovable person. Understanding such distinctions is important for many applications, including parsing (Sag et al., 2002) and machine translation (Shutova et al., 2012).

We use Wiktionary as a large, but incomplete, reference for idiomatic entries; individual entries can be marked as idiomatic but, in practice, most are not. Using these incomplete annotations as supervision, we train a binary Perceptron classifier for identifying idiomatic dictionary entries. We introduce new lexical and graph-based features that use WordNet and Wiktionary to compute semantic relatedness. This allows us to learn, for example, that the words in the phrase *diamond in the rough* are more closely related to the words in its literal definition than the idiomatic one. Experiments demonstrate that the classifier achieves precision of over 65% at recall over 52% and that, when used to fill in missing Wiktionary idiom labels, it more than doubles the number of idioms from 7,764 to 18,155.

These gains also translate to idiom detection in sentences, by simply using the Lesk word sense disambiguation (WSD) algorithm (1986) to match phrases to their definitions. This approach allows for scalable detection with no restrictions on the syntactic structure or context of the target phrase. In a set of Wiktionary definition example sentences, the more complete set of idioms boosts detection recall by over 28 percentage points.

2 Related Work

To the best of our knowledge, this work represents the first attempt to identify dictionary entries as idiomatic and the first to reduce idiom detection to identification via a dictionary.

Previous idiom detection systems fall in one of two paradigms: *phrase classification*, where a phrase p is always idiomatic or literal, e.g. (Gedigian et al., 2006; Shutova et al., 2010), or *token classification*, where each occurrence of a phrase p can be idiomatic or literal, e.g. (Katz and Giesbrecht, 2006;

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1417–1421,
Seattle, Washington, USA, 18-21 October 2013.

Birke and Sarkar, 2006; Li and Sporleder, 2009). Most previous idiom detection systems have focused on specific syntactic constructions. For instance, Shutova et al. (2010) consider subject/verb (*campaign surged*) and verb/direct-object idioms (*stir excitement*) while Fazly and Stevenson (2006), Cook et al. (2007), and Diab and Bhutada (2009) detect verb/noun idioms (*blow smoke*). Fothergill and Baldwin (2012) are syntactically unconstrained, but only study Japanese idioms. Although we focus on identifying idiomatic dictionary entries, one advantage of our approach is that it enables syntactically unconstrained token-level detection for any phrase in the dictionary.

3 Formal Problem Definitions

Identification For identification, we assume data of the form $\{(\langle p_i, d_i \rangle, y_i) : i = 1 \ldots n\}$ where p_i is the phrase associated with definition d_i and $y_i \in \{\text{literal, idiomatic}\}$. For example, this would include both the literal pair ⟨ "leave for dead", "To abandon a person or other living creature that is injured or otherwise incapacitated, assuming that the death of the one abandoned will soon follow."⟩ and the idiomatic pair ⟨ "leave for dead", "To disregard or bypass as unimportant." ⟩. Given $\langle p_i, d_i \rangle$, we aim to predict y_i.

Detection To evaluate identification in the context of detection, we assume data $\{(\langle p_i, e_i \rangle, y_i) : i = 1 \ldots n\}$. Here, p_i is the phrase in example sentence e_i whose idiomatic status is labeled $y_i \in \{\text{idiomatic, literal}\}$. One such idiomatic pair is ⟨"heart to heart", "They sat down and had a long overdue heart to heart about the future of their relationship."⟩. Given $\langle p_i, e_i \rangle$, we again aim to predict y_i.

4 Data

We gathered phrases, definitions, and example sentences from the English-language Wiktionary dump from November 13th, 2012.[1]

Identification Phrase, definition pairs $\langle p, d \rangle$ were gathered with the following restrictions: the title of the Wiktionary entry must be English, p must composed of two or more words w, and $\langle p, d \rangle$ must be in its base form—senses that are not defined as a different tense of a phrase—e.g. the pair ⟨ "weapons of mass destruction", "Plural form of weapon of mass destruction" ⟩ was removed while the pair ⟨ "weapon of mass destruction", "A chemical, biological, radiological, nuclear or other weapon that ... "⟩ was kept.

Data Set	Literal	Idiomatic	Total
All	56,037	7,764	63,801
Train	47,633	6,600	54,233
Unannotated Dev	2,801	388	3,189
Annotated Dev	2,212	958	3,170
Unannotated Test	5,603	776	6,379
Annotated Test	4,510	1,834	6,344

Figure 1: Number of dictionary entries with each class for the Wiktionary identification data.

Data Set	Literal	Idiomatic	Total
Dev	171	330	501
Test	360	695	1055

Figure 2: Number of sentences of each class for the Wiktionary detection data.

Each pair $\langle p, d \rangle$ was assigned label y according to the idiom labels in Wiktionary, producing the Train, Unannotated Dev, and Unannotated Test data sets. In practice, this produces a noisy assignment because a majority of the idiomatic senses are not marked. The development and test sets were annotated to correct these potential omissions. Annotators used the definition of an idiom as a "phrase with a non-compositional meaning" to produce the Annotated Dev and Annotated Test data sets. Figure 1 presents the data statistics.

We measured inter-annotator agreement on 1,000 examples. Two annotators marked each dictionary entry as literal, idiomatic, or indeterminable. Less than one half of one percent could not be determined[2]—the computed kappa was 81.85. Given this high level of agreement, the rest of the data were only labeled by a single annotator, following the methodology used with the VNC-Tokens Dataset (Cook et al., 2008).

Detection For detection, we gathered the example sentences provided, when available, for each definition used in our annotated identification data sets. These sentences provide a clean source of develop-

[1] We used the Java Wiktionary Library (Zesch et al., 2008).

[2] The indeterminable pairs were omitted from the data.

ment and test data containing idiomatic and literal phrase usages. In all, there were over 1,300 unique phrases, half of which had more than one possible dictionary definition in Wiktionary. Figure 2 provides the complete statistics.

5 Identification Model

For identification, we use a linear model that predicts class $y^* \in \{\text{literal, idiomatic}\}$ for an input pair $\langle p, d \rangle$ with phrase p and definition d. We assign the class:

$$y^* = \arg\max_y \theta \cdot \phi(p, d, y)$$

given features $\phi(p, d, y) \in \mathbb{R}^n$ with associated parameters $\theta \in \mathbb{R}^n$.

Learning In this work, we use the averaged Perceptron algorithm (Freund and Schapire, 1999) to perform learning, which was optimized in terms of iterations T, bounded by range [1, 100], by maximizing F-measure on the development set.

The models described correspond to the features they use. All models are trained on the same, unannotated training data.

Features The features that were developed fall into two categories: lexical and graph-based features. The lexical features were motivated by the intuition that literal phrases are more likely to have closely related words in d to those in p because literal phrases do not break the principle of compositionality. All words compared are stemmed versions. Let $count(w, t)$ = number of times word w appears in text t.

- synonym overlap: Let S be the set of synonyms as defined in Wiktionary for all words in p. Then, we define the synonym overlap = $\frac{1}{|S|} \sum_{s \in S} count(s, d)$.
- antonym overlap: Let A be the set of antonyms as defined in Wiktionary for all words in p. Then, we define the antonym overlap = $\frac{1}{|A|} \sum_{a \in A} count(a, d)$.
- average number of capitals:[3] The value of $\frac{\text{number of capital letters in } p}{\text{number of words in } p}$.

[3] In practice, this feature identifies most proper nouns.

Graph-based features use the graph structure of WordNet 3.0 to calculate path distances. Let $distance(w, v, rel, n)$ be the minimum distance via links of type rel in WordNet from a word w to a word v, up to a threshold max integer value n, and 0 otherwise. The features compute:

- closest synonym:

$$\min_{w \in p, v \in d} distance(w, v, synonym, 5)$$

- closest antonym:[4]

$$\min_{w \in p, v \in d} distance(w, v, antonym, 5)$$

- average synonym distance:

$$\frac{1}{|p|} \sum_{w \in p, v \in d} distance(w, v, synonym, 5)$$

- average hyponym:

$$\frac{1}{|p|} \sum_{w \in p, v \in d} distance(w, v, hyponym, 5)$$

- synsets connected by an antonym: This feature indicates whether the following is true. The set of synsets Syn_p, all synsets from all words in p, and the set of synsets Syn_d, all synsets from all words in d, are connected by a shared antonym. This feature follows an approach described by Budanitsky et al. (2006).

6 Experiments

We report identification and detection results, varying the data labeling and choice of feature sets.

6.1 Identification

Random Baseline We use a proportionally random baseline for the identification task that classifies according to the proportion of literal definitions seen in the training data.

Results Figure 3 provides the results for the baseline, the full approach, and variations with subsets of the features. Results are reported for the original, unannotated test set, and the same test examples with corrected idiom labels. All models increased

[4] The first relation expanded was the antonym relation. All subsequent expansions were via synonym relations.

Data Set	Model	Rec.	Prec.	F1
Unannotated	Lexical	85.8	21.9	34.9
	Graph	62.4	26.6	37.3
	Lexical+Graph	70.5	28.1	**40.1**
	Baseline	12.2	11.9	12.0
Annotated	Lexical	81.2	49.3	61.4
	Graph	64.3	51.3	57.1
	Lexical+Graph	75.0	52.9	**62.0**
	Baseline	29.5	12.5	17.6

Figure 3: Results for idiomatic definition identification.

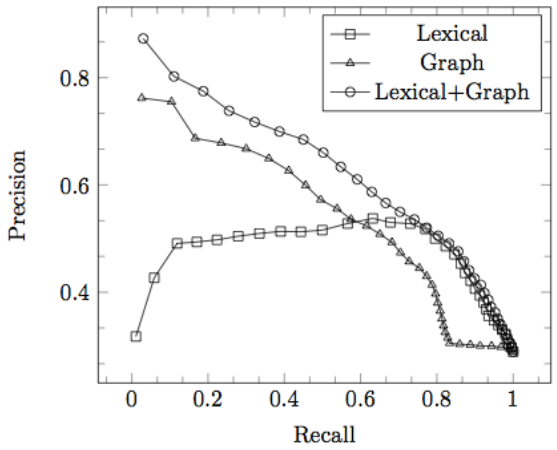


Figure 4: Precision and recall with varied features on the annotated test set.

over their corresponding baselines by more than 22 points and both feature families contributed.[5]

Figure 4 shows the complete precision, recall curve. We selected our operating point to optimize F-measure, but we see that the graph features perform well across all recall levels and that adding the lexical features provides consistent improvement in precision. However, other points are possible, especially when aiming for high precision to extend the labels in Wiktionary. For example, the original 7,764 entries can be extended to 18,155 at 65% precision, 9,594 at 80%, or 27,779 at 52.9%.

Finally, Figures 5 and 6 present qualitative results, including newly discovered idioms and high scoring false identifications. Analysis reveals where our system has room to improve—errors most often occur with phrases that are specific to a certain field, such as sports or mathematics, and with phrases whose words also appear in their definitions.

[5]We also ran ablations demonstrating that removing each feature from the Lexical+Graph model hurt performance, but omit the detailed results for space.

Phrase	Definition
feel free	You have my permission.
live down	To get used to something shameful.
nail down	To make something (e.g. a decision or plan) firm or certain.
make after	To chase.
get out	To say something with difficulty.
good riddance to bad rubbish	A welcome departure.
as all hell	To a great extent or degree; very.
roll around	To happen, occur, take place.

Figure 5: Newly discovered idioms.

Phrase	Definition
put asunder	To sunder; disjoin; separate; disunite; divorce; annul; dissolve.
add up	To take a sum.
peel off	To remove (an outer layer or covering, such as clothing).
straighten up	To become straight, or straighter.
wild potato	The edible root of this plant.
shallow embedding	The act of representing one logic or language with another by providing a syntactic translation.

Figure 6: High scoring false identifications.

6.2 Detection

Approach We use the Lesk (1986) algorithm to perform WSD, matching an input phrase p from sentence e to the definition d in Wiktionary that defines the sense p is being used in. The final classification y is then assigned to $\langle p, d \rangle$ by the identification model.

Results Figure 7 shows detection results. The baseline for this experiment is a model that assigns the default labels within Wiktionary to the disambiguated definition. The Annotated model is the Lexical+Graph model shown in Figure 3 evaluated on the annotated data. The +Default setting augments the identification model by labeling the $\langle p, e \rangle$ as idiomatic if either the model or the original label within Wiktionary identifies it as such.

7 Conclusions

We presented a supervised approach to classifying definitions as idiomatic or literal that more than dou-

Model	Rec.	Prec.	F1
Default	60.5	1	75.4
Annotated	78.3	76.7	77.5
Annotated+Default	89.2	79.0	**83.8**

Figure 7: Detection results.

bles the number of marked idioms in Wiktionary, even when training on incomplete data. When combined with the Lesk word sense algorithm, this approach provides a complete idiom detector for any phrase in the dictionary.

We expect that semi-supervised learning techniques could better recover the missing labels and boost overall performance. We also think it should be possible to scale the detection approach, perhaps with automatic dictionary definition discovery, and evaluate it on more varied sentence types.

Acknowledgments

The research was supported in part by the National Science Foundation (IIS-1115966) and a Mary Gates Research Scholarship. The authors thank Nicholas FitzGerald, Sarah Vieweg, and Mark Yatskar for helpful discussions and feedback.

References

J. Birke and A. Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

A. Budanitsky and G. Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

P. Cook, A. Fazly, and S. Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the workshop on a broader perspective on multiword expressions*.

P. Cook, A. Fazly, and S. Stevenson. 2008. The vnc-tokens dataset. In *Proceedings of the Language Resources and Evaluation Conference Workshop Towards a Shared Task for Multiword Expressions*.

M. Diab and P. Bhutada. 2009. Verb noun construction mwe token supervised classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*.

A. Fazly and S. Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

R. Fothergill and T. Baldwin. 2012. Combining resources for mwe-token classification. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*.

Y. Freund and R.E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

M. Gedigian, J. Bryant, S. Narayanan, and B. Ciric. 2006. Catching metaphors. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*.

G. Katz and E. Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*.

M. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of Special Interest Group on the Design of Communication*.

L. Li and C. Sporleder. 2009. Classifier combination for contextual idiom detection without labelled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

I. Sag, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing*. Springer.

E. Shutova, L. Sun, and A. Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the International Conference on Computational Linguistics*.

E. Shutova, S. Teufel, and A. Korhonen. 2012. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

T. Zesch, C. Müller, and I. Gurevych. 2008. Extracting lexical semantic knowledge from wikipedia and wiktionary. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Elephant: Sequence Labeling for Word and Sentence Segmentation

Kilian Evang[*], Valerio Basile[*], Grzegorz Chrupała[†] and Johan Bos[*]
[*]University of Groningen, Oude Kijk in 't Jatstraat 26, 9712 EK Groningen, The Netherlands
[†]Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands
[*]{k.evang, v.basile, johan.bos}@rug.nl [†]g.chrupala@uvt.nl

Abstract

Tokenization is widely regarded as a solved problem due to the high accuracy that rule-based tokenizers achieve. But rule-based tokenizers are hard to maintain and their rules language specific. We show that high-accuracy word and sentence segmentation can be achieved by using supervised sequence labeling on the character level combined with unsupervised feature learning. We evaluated our method on three languages and obtained error rates of 0.27 ‰ (English), 0.35 ‰ (Dutch) and 0.76 ‰ (Italian) for our best models.

1 An Elephant in the Room

Tokenization, the task of segmenting a text into words and sentences, is often regarded as a solved problem in natural language processing (Dridan and Oepen, 2012), probably because many corpora are already in tokenized format. But like an elephant in the living room, it is a problem that is impossible to overlook whenever new raw datasets need to be processed or when tokenization conventions are reconsidered. It is moreover an important problem, because any errors occurring early in the NLP pipeline affect further analysis negatively. And even though current tokenizers reach high performance, there are three issues that we feel haven't been addressed satisfactorily so far:

- Most tokenizers are rule-based and therefore hard to maintain and hard to adapt to new domains and new languages (Silla Jr. and Kaestner, 2004);
- Word and sentence segmentation are often seen as separate tasks, but they obviously inform each other and it could be advantageous to view them as a combined task;
- Most tokenization methods provide no alignment between raw and tokenized text, which makes mapping the tokenized version back onto the actual source hard or impossible.

In short, we believe that regarding tokenization, there is still room for improvement, in particular on the methodological side of the task. We are particularly interested in the following questions: Can we use supervised learning to avoid hand-crafting rules? Can we use unsupervised feature learning to reduce feature engineering effort and boost performance? Can we use the same method across languages? Can we combine word and sentence boundary detection into one task?

2 Related Work

Usually the text segmentation task is split into word tokenization and sentence boundary detection. Rule-based systems for finding word and sentence boundaries often are variations on matching hand-coded regular expressions (Grefenstette, 1999; Silla Jr. and Kaestner, 2004; Jurafsky and Martin, 2008; Dridan and Oepen, 2012).

Several unsupervised systems have been proposed for sentence boundary detection. Kiss and Strunk (2006) present a language-independent, unsupervised approach and note that abbreviations form a major source of ambiguity in sentence boundary detection and use collocation detection to build a high-accuracy abbreviation detector. The resulting system reaches high accuracy, rivalling handcrafted rule-based and supervised systems. A similar system was proposed earlier by Mikheev (2002).

Existing supervised learning approaches for sentence boundary detection use as features tokens preceding and following potential sentence boundary, part of speech, capitalization information and lists of abbreviations. Learning methods employed in

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1422–1426,
Seattle, Washington, USA, 18-21 October 2013.

these approaches include maximum entropy models (Reynar and Ratnaparkhi, 1997) decision trees (Riley, 1989), and neural networks (Palmer and Hearst, 1997).

Closest to our work are approaches that present token and sentence splitters using conditional random fields (Tomanek et al., 2007; Fares et al., 2013). However, these previous approaches consider tokens (i.e. character sequences) as basic units for labeling, whereas we consider single characters. As a consequence, labeling is more resource-intensive, but it also gives us more expressive power. In fact, our approach kills two birds with one stone, as it allows us to integrate token and sentence boundaries detection into one task.

3 Method

3.1 IOB Tokenization

IOB tagging is widely used in tasks identifying chunks of tokens. We use it to identify chunks of characters. Characters outside of tokens are labeled O, inside of tokens I. For characters at the beginning of tokens, we use S at sentence boundaries, otherwise T (for token). This scheme offers some nice features, like allowing for discontinuous tokens (e.g. hyphenated words at line breaks) and starting a new token in the middle of a typographic word if the tokenization scheme requires it, as e.g. in *did|n't*. An example is given in Figure 1.

```
It didn't matter if the faces were male,
SIOTIITIIOTIIIIIOTIOTIIOTIIIIOTIIIOTIIITO
female or those of children. Eighty-
TIIIIIOTIOTIIIIOTIOTIIIIIIITOSIIIIIIO
three percent of people in the 30-to-34
IIIIIOTIIIIIIOTIOTIIIIIOTIOTIIOTIIIIIIIO
year old age range gave correct responses.
TIIIOTIIOTIIOTIIIIOTIIIOTIIIIIIOTIIIIIIIIT
```

Figure 1: Example of IOB-labeled characters

3.2 Datasets

In our experiments we use three datasets to compare our method for different languages and for different domains: manually checked English newswire texts taken from the Groningen Meaning Bank, GMB (Basile et al., 2012), Dutch newswire texts, comprising two days from January 2000 extracted from the Twente News Corpus, TwNC (Ordelman et al., 2007), and a random sample of Italian texts from the PAISÀ corpus (Borghetti et al., 2011).

Table 1: Datasets characteristics.

Name	Language	Domain	Sentences	Tokens
GMB	English	Newswire	2,886	64,443
TNC	Dutch	Newswire	49,537	860,637
PAI	Italian	Web/various	42,674	869,095

The data was converted into IOB format by inferring an alignment between the raw text and the segmented text.

3.3 Sequence labeling

We apply the Wapiti implementation (Lavergne et al., 2010) of Conditional Random Fields (Lafferty et al., 2001), using as features the output label of each character, combined with 1) the character itself, 2) the output label on the previous character, 3) characters and/or their Unicode categories from context windows of varying sizes. For example, with a context size of 3, in Figure 1, features for the E in *Eighty-three* with the output label S would be E/S, O/S, ␣/S, i/S, Space/S, Lowercase/S. The intuition is that the 31 existing Unicode categories can generalize across similar characters whereas character features can identify specific contexts such as abbreviations or contractions (e.g. *didn't*). The context window sizes we use are 0, 1, 3, 5, 7, 9, 11 and 13, centered around the focus character.

3.4 Deep learning of features

Automatically learned word embeddings have been successfully used in NLP to reduce reliance on manual feature engineering and boost performance. We adapt this approach to the character level, and thus, in addition to hand-crafted features we use text representations induced in an unsupervised fashion from character strings. A complete discussion of our approach to learning text embeddings can be found in (Chrupała, 2013). Here we provide a brief overview.

Our representations correspond to the activation of the hidden layer in a simple recurrent neural (SRN) network (Elman, 1990; Elman, 1991), implemented in a customized version of Mikolov (2010)'s RNNLM toolkit. The network is sequentially presented with a large amount of raw text and learns to

predict the next character in the sequence. It uses the units in the hidden layer to store a generalized representation of the recent history. After training the network on large amounts on unlabeled text, we run it on the training and test data, and record the activation of the hidden layer at each position in the string as it tries to predict the next character. The vector of activations of the hidden layer provides additional features used to train and run the CRF. For each of the $K = 10$ most active units out of total $J = 400$ hidden units, we create features $(f(1) \dots f(K))$ defined as $f(k) = 1$ if $s_{j(k)} > 0.5$ and $f(k) = 0$ otherwise, where $s_j(k)$ returns the activation of the k^{th} most active unit. For training the SRN only raw text is necessary. We trained on the entire GMB 2.0.0 (2.5M characters), the portion of TwNC corresponding to January 2000 (43M characters) and a sample of the PAISÀ corpus (39M characters).

4 Results and Evaluation

In order to evaluate the quality of the tokenization produced by our models we conducted several experiments with different combinations of features and context sizes. For these tests, the models are trained on an 80% portion of the data sets and tested on a 10% development set. Final results are obtained on a 10% test set. We report both absolute number of errors and error rates per thousand (‰).

4.1 Feature sets

We experiment with two kinds of features at the character level, namely Unicode categories (31 different ones), Unicode character codes, and a combination of them. Unicode categories are less sparse than the character codes (there are 88, 134, and 502 unique characters for English, Dutch and Italian, respectively), so the combination provide some generalization over just character codes.

Table 2: Error rates obtained with different feature sets. Cat stands for Unicode category, Code for Unicode character code, and Cat-Code for a union of these features.

Feature set	Error rates per thousand (‰) English	Dutch	Italian
Cat-9	45 (1.40)	1,403 (2.87)	1,548 (2.67)
Code-9	**6 (0.19)**	782 (1.60)	692 (1.20)
Cat-Code-9	8 (0.25)	**774 (1.58)**	**657 (1.14)**

From these results we see that categories alone perform worse than only codes. For English there is no gain from the combination over using only character codes. For Dutch and Italian there is an improvement, although it is only significant for Italian ($p = 0.480$ and $p = 0.005$ respectively, binomial exact test). We use this feature combination in the experiments that follow. Note that these models are trained using a symmetrical context of 9 characters (four left and four right of the current character). In the next section we show performance of models with different window sizes.

4.2 Context window

We run an experiment to evaluate how the size of the context in the training phase impacts the classification. In Table 4.2 we show the results for symmetrical windows ranging in size from 1 to 13.

Table 3: Using different context window sizes.

Feature set	Error rates per thousand (‰) English	Dutch	Italian
Cat-Code-1	273 (8.51)	4,924 (10.06)	9,108 (15.86)
Cat-Code-3	118 (3.68)	3,525 (7.20)	2,013 (3.51)
Cat-Code-5	20 (0.62)	930 (1.90)	788 (1.37)
Cat-Code-7	10 (0.31)	778 (1.60)	667 (1.16)
Cat-Code-9	**8 (0.25)**	774 (1.58)	**657 (1.14)**
Cat-Code-11	9 (0.28)	761 (1.56)	692 (1.21)
Cat-Code-13	**8 (0.25)**	**751 (1.54)**	670 (1.17)

4.3 SRN features

We also tested the automatically learned features derived from the activation of the hidden layer of an SRN language model, as explained in Section 3. We combined these features with character code and Unicode category features in windows of different sizes. The results of this test are shown in Table 4. The first row shows the performance of SRN features on their own. The following rows show the combination of SRN features with the basic feature sets of varying window size. It can be seen that augmenting the feature sets with SRN features results in large reductions of error rates. The Cat-Code-1-SRN setting has error rates comparable to Cat-Code-9.

The addition of SRN features to the two best previous models, Cat-Code-9 and Cat-Code-13, reduces the error rate by 83% resp. 81% for Dutch,

and by 24% resp. 26% for Italian. All these differences are statistically significant according to the binomial test ($p < 0.001$). For English, there are too few errors to detect a statistically significant effect for Cat-Code-9 ($p = 0.07$), but for Cat-Code-13 we find $p = 0.016$.

Table 4: Results obtained using different context window sizes and addition of SRN features.

Feature set	Error rates per thousand (‰) English	Dutch	Italian
SRN	24 (0.75)	276 (0.56)	738 (1.28)
Cat-Code-1-SRN	7 (0.21)	212 (0.43)	549 (0.96)
Cat-Code-3-SRN	4 (0.13)	165 (0.34)	507 (0.88)
Cat-Code-5-SRN	3 (0.10)	136 (0.28)	476 (0.83)
Cat-Code-7-SRN	**1 (0.03)**	**111 (0.23)**	497 (0.86)
Cat-Code-9-SRN	2 (0.06)	135 (0.28)	497 (0.86)
Cat-Code-11-SRN	2 (0.06)	132 (0.27)	**468 (0.81)**
Cat-Code-13-SRN	**1 (0.03)**	142 (0.29)	496 (0.86)

In a final step, we selected the best models based on the development sets (Cat-Code-7-SRN for English and Dutch, Cat-Code-11-SRN for Italian), and checked their performance on the final test set. This resulted in 10 errors (0.27 ‰) for English (GMB corpus), 199 errors (0.35 ‰) for Dutch (TwNC corpus), and 454 errors (0.76 ‰) for Italian (PAISÀ corpus).

5 Discussion

It is interesting to examine what kind of errors the SRN features help avoid. In the English and Dutch datasets many errors are caused by failure to recognize personal titles and initials or misparsing of numbers. In the Italian data, a large fraction of errors is due to verbs with clitics, which are written as a single word, but treated as separate tokens. Table 5 shows examples of errors made by a simpler model that are fixed by adding SRN features. Table 6 shows the confusion matrices for the Cat-Code-7 and Cat-Code-7-SRN sets on the Dutch data. The mistake most improved by SRN features is T/I with 89% error reduction (see also Table 5). The is also the most common remaining mistake.

A comparison with other approaches is hard because of the difference in datasets and task definition (combined word/sentence segmentation). Here we just compare our results for sentence segmentation (sentence F_1 score) with Punkt, a state-of-the-art sentence boundary detection system (Kiss and Strunk, 2006). With its standard distributed models, Punkt achieves 98.51% on our English test set, 98.87% on Dutch and 98.34% on Italian, compared with 100%, 99.54% and 99.51% for our system. Our system benefits here from its ability to adapt to a new domain with relatively little (but annotated) training data.

Table 5: Positive impact of SRN features.

	`Ms. Hughes will joi`
Cat-Code-7	`SIIO`**`S`**`IIIIIOTIIIOTII`
Cat-Code-7-SRN	`SIIO`**`T`**`IIIIIOTIIIOTII`
	`$ 3.9 trillion by t`
Cat-Code-7	`TOT`**`T`**`IOTIIIIIIIOTIOT`
Cat-Code-7-SRN	`TOT`**`I`**`IOTIIIIIIIOTIOT`
	`bleek 0,4 procent`
Cat-Code-11	`OTIIIIOT`**`T`**`IOTIIIIIIO`
Cat-Code-11-SRN	`OTIIIIOT`**`I`**`IOTIIIIIIO`
	`toebedeeld: 6,2. In`
Cat-Code-11	`TIIIIIIIIITOT`**`T`**`ITOSI`
Cat-Code-11-SRN	`TIIIIIIIIITOT`**`I`**`ITOSI`
	`prof. Teulings het`
Cat-Code-11	`TIII`**`T`**`O`**`S`**`IIIIIIIOTIIO`
Cat-Code-11-SRN	`TIII`**`I`**`O`**`T`**`IIIIIIIOTIIO`
	`per costringerlo al`
Cat-Code-11	`TIIOTIIIIIIIII`**`I`**`IOTI`
Cat-Code-11-SRN	`TIIOTIIIIIIIII`**`T`**`IOTI`

Table 6: Confusion matrix for Dutch development set.

	Predicted, Cat-Code-7				Predicted, Cat-Code-7-SRN			
Gold	I	O	S	T	I	O	S	T
I	328128	0	2	469	328546	0	0	53
O	0	75234	0	0	0	75234	0	0
S	4	0	4323	18	1	0	4332	12
T	252	0	33	80828	35	0	10	81068

6 What Elephant?

Word and sentence segmentation can be recast as a combined tagging task. This way, tokenization is cast as a supervised learning task, causing a shift of labor from writing rules to manually correcting labels. Learning this task with CRF achieves high accuracy.[1] Furthermore, our tagging method does not lose the connection between original text and tokens.

In future work, we plan to broaden the scope of this work to other steps in document preparation,

[1] All software needed to replicate our experiments is available at `http://gmb.let.rug.nl/elephant/experiments.php`

such as normalization of punctuation, and their interaction with segmentation. We further plan to test our method on a wider range of datasets, allowing a more direct comparison with other approaches. Finally, we plan to explore the possibility of a statistical universal segmentation model for mutliple languages and domains.

In a famous scene with a live elephant on stage, the comedian Jimmy Durante was asked about it by a policeman and surprisedly answered: "What elephant?" We feel we can say the same now as far as tokenization is concerned.

References

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3196–3200, Istanbul, Turkey.

Claudia Borghetti, Sara Castagnoli, and Marco Brunello. 2011. I testi del web: una proposta di classificazione sulla base del corpus PAISÀ. In M. Cerruti, E. Corino, and C. Onesti, editors, *Formale e informale. La variazione di registro nella comunicazione elettronica*, pages 147–170. Carocci, Roma.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, USA.

Rebecca Dridan and Stephan Oepen. 2012. Tokenization: Returning to a long solved problem – a survey, contrastive experiment, recommendations, and toolkit –. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–382, Jeju Island, Korea. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jeffrey L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2):195–225.

Murhaf Fares, Stephan Oepen, and Zhang Yi. 2013. Machine learning for high-quality tokenization - replicating variable tokenization schemes. In A. Gelbukh, editor, *CICLING 2013*, volume 7816 of *Lecture Notes in Computer Science*, pages 231–244, Berlin Heidelberg. Springer-Verlag.

Gregory Grefenstette. 1999. Tokenization. In Hans van Halteren, editor, *Syntactic Wordclass Tagging*, pages 117–133. Kluwer Academic Publishers, Dordrecht.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2nd edition.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, pages 282–289.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, Uppsala, Sweden, July. Association for Computational Linguistics.

Andrei Mikheev. 2002. Periods, capitalized words, etc. *Computational Linguistics*, 28(3):289–318.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Roeland Ordelman, Franciska de Jong, Arjan van Hessen, and Hendri Hondorp. 2007. TwNC: a multifaceted Dutch news corpus. *ELRA Newsleter*, 12(3/4):4–7.

David D. Palmer and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, DC, USA. Association for Computational Linguistics.

Michael D. Riley. 1989. Some applications of tree-based modelling to speech and language. In *Proceedings of the workshop on Speech and Natural Language*, HLT '89, pages 339–352, Stroudsburg, PA, USA. Association for Computational Linguistics.

Carlos N. Silla Jr. and Celso A. A. Kaestner. 2004. An analysis of sentence boundary detection systems for English and Portuguese documents. In *Fifth International Conference on Intelligent Text Processing and Computational Linguistics*, volume 2945 of *Lecture Notes in Computer Science*, pages 135–141. Springer.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Sentence and token splitting based on conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 49–57, Melbourne, Australia.

Detecting Compositionality of Multi-Word Expressions using Nearest Neighbours in Vector Space Models

Douwe Kiela
University of Cambridge
Computer Laboratory
douwe.kiela@cl.cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
stephen.clark@cl.cam.ac.uk

Abstract

We present a novel unsupervised approach to detecting the compositionality of multi-word expressions. We compute the compositionality of a phrase through substituting the constituent words with their "neighbours" in a semantic vector space and averaging over the distance between the original phrase and the substituted neighbour phrases. Several methods of obtaining neighbours are presented. The results are compared to existing supervised results and achieve state-of-the-art performance on a verb-object dataset of human compositionality ratings.

1 Introduction

Multi-word expressions (MWEs) are defined as "idiosyncratic interpretations that cross word boundaries" (Sag et al., 2002). They tend to have a standard syntactic structure but are often semantically non-compositional; i.e. their meaning is not fully determined by their syntactic structure and the meanings of their constituents. A classic example is *kick the bucket*, which means *to die* rather than *to hit a bucket with the foot*. These types of expressions account for a large proportion of day-to-day language interactions (Schuler and Joshi, 2011) and present a significant problem for natural language processing systems (Sag et al., 2002).

This paper presents a novel unsupervised approach to detecting the compositionality of MWEs, specifically of verb-noun collocations. The idea is that we can recognize compositional phrases by substituting related words for constituent words in the phrase: if the result of a substitution yields a meaningful phrase, its individual constituents are likely to contribute toward the overall meaning of the phrase. Conversely, if a substitution yields a non-sensical phrase, its constituents are likely to contribute less or not at all to the overall meaning of the phrase. For the phrase *eat her hat*, for example, we might consider the following substituted phrases:

1. *consume* her hat
2. eat her *trousers*

Both phrases are semantically anomalous, implying that *eat hat* is a highly non-compositional verb-noun collocation. Following a similar procedure for *eat apple*, however, would not lead to an anomaly: *consume apple* and *eat pear* are perfectly meaningful, leading us to believe that *eat apple* is compositional.

In the context of distributional models, this idea can be formalised in terms of vector spaces:

> the average distance between a phrase vector and its substituted phrase vectors is related to its compositionality.

Since we are relying on the relative distances of *phrases* in semantic space, we require a method for computing vectors for phrases. We experimented with a number of composition operators from Mitchell and Lapata (2010), in order to compose constituent word vectors into phrase vectors. The relation between phrase vectors and substituted phrase vectors is most pronounced in the case of

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1427–1432,
Seattle, Washington, USA, 18-21 October 2013.

pointwise multiplication, which has the effect of placing semantically anomalous phrases relatively close together in space (since the vectors for the constituent words have little in common), whereas the semantically meaningful phrases are further apart. This implies that compositional phrases are less similar to their neighbours, which is to say that the greater the average distance between a phrase vector and its substituted phrase vectors, the greater its compositionality.

The contribution of this short focused research paper is a novel approach to detecting the compositionality of multi-word expressions that makes full use of the ability of semantic vector space models to calculate distances between words and phrases. Using this unsupervised approach, we achieve state-of-the-art performance in a direct comparison with existing supervised methods.

2 Dataset and Vectors

The verb-noun collocation dataset from Venkatapathy and Joshi (2005), which consists of 765 verb-object pairs with human compositionality ratings, was used for evaluation. Venkatapathy & Joshi used a support vector machine (SVM) to obtain a Spearman ρ_s correlation of 0.448. They employed a variety of features ranging from frequency to LSA-derived similarity measures and used 10% of the dataset as training data with tenfold cross-validation. McCarthy et al. (2007) used the same dataset and expanded on the original approach by adding WordNet and distributional prototypes to the SVM, achieving a ρ_s correlation of 0.454.

The distributional vectors for our experiments were constructed from the ukWaC corpus (Baroni et al., 2009). Vectors were obtained using a standard window method (with a window size of 5) and the 50,000 most frequent context words as features, with stopwords removed. We also experimented with syntax-based co-occurrence features extracted from a dependency-parsed version of ukWaC, but in agreement with results obtained by Schulte im Walde et al. (2013) for predicting compositionality in German, the window-based co-occurrence method produced better results.

We tried several weighting schemes from the literature, such as t-test (Curran, 2004), positive mutual information (Bullinaria and Levy, 2012) and the ratio of the probability of the context word given the target word[1] to the context word's overall probability (Mitchell and Lapata, 2010). We found that a tf-idf variant called LTU yielded the best results, defined as follows (Reed et al., 2006):

$$w_{ij} = \frac{(\log(f_{ij}) + 1.0)\ \log(\frac{N}{n_j})}{0.8 + 0.2 \times \frac{|\textit{context word}|}{|\textit{avg context word}|}}$$

where f_{ij} is the number of times that the target word and context word co-occur in the same window, n_j is the context word frequency, N is the total frequency and $|\textit{context word}|$ is the total number of occurrences of a context word. Distance is calculated using the standard cosine measure:

$$dist(v_1, v_2) = 1 - \frac{v_1 \cdot v_2}{|v_1||v_2|}$$

where v_1 and v_2 are vectors in the semantic vector space model.

3 Finding Neighbours and Computing Compositionality

We experimented with two different ways of obtaining neighbours for the constituent words in a phrase. Since vector space models lend themselves naturally to similarity computations, one way to get neighbours is to take the k-most similar vectors from a similarity matrix. This approach is straightforward, but has some potential drawbacks: it assumes that we have a large number of vectors to select neighbours from, and becomes computationally expensive when the number of neighbours is increased.

An alternative source for obtaining neighbours is the lexical database WordNet (Fellbaum, 1998). We define neighbours as siblings in the hypernym hierarchy, so that the neighbours of a word can be found by taking the hyponyms of its hypernyms. WordNet also allows us to extract only neighbours of the same grammatical type (yielding noun neighbours for nouns and verb neighbours for verbs, for example). Since not every word has the same number of neighbours in WordNet, we use only the first k

[1] We use *target word* to refer to the word for which a vector is being constructed.

neighbours, which means that the neighbours have to be ranked. An obvious ranking method is to use the frequency with which each neighbour co-occurs with the other constituent(s) of the same phrase. For example, for all the WordNet neighbours of *eat* (for all senses of *eat*), we count the co-occurrences with *hat* in a given window size and rank them accordingly. This ranking method also has the desirable side-effect of performing some word sense disambiguation, at least in some cases. For example, the highly ranked neighbours of *apple* for *eat apple* are likely to be items of food, and not (inedible) trees (apple is also a tree in WordNet).

In order to obtain frequency-ranked neighbours, we used the ukWaC corpus with a window size of 5. One reason for having multiple neighbours is that it allows us to correct for word sense disambiguation errors (as mentioned above), since averaging over results for several neighbours reduces the impact of including incorrect senses. For example, the first 20 neighbours of *eat*, ranked by co-occurrence frequency with all the objects of *eat* in the dataset, are:

> *eat use consume drink sample smoke swallow spend break hit save afford burn partake dine breakfast worry damage deplete drug*

One problem with the evaluation dataset is that it does not solely consist of verb-noun pairs: 84 phrases contain pronouns, while there are also several examples containing words that WordNet considers to be adjectives rather than nouns. This problem was mitigated by part-of-speech tagging the dataset. As neighbours for pronouns (which are not included in WordNet), we used the other pronouns present in the dataset. For the remaining words, we included the part-of-speech when looking up the word in WordNet.

3.1 Average distance compositionality score

We considered several different ways of constructing phrasal vectors. We chose not to use the compositional models of Baroni and Zamparelli (2010) and Socher et al. (2011) because we believe that it is important that our methods are completely unsupervised and do not require any initial learning phase. Hence, we experimented with different ways of constructing phrasal vectors according to Mitchell and Lapata (2010) and found that pointwise multiplication $\odot$ worked best in our experiments. Thus, we define the composed vector $\overrightarrow{eat\ hat}$ as:

$$\overrightarrow{eat} \odot \overrightarrow{hat}$$

We can now compute a compositionality score s_c by averaging the distance between the original phrase vector and its substituted neighbour phrase vectors via the following formula:

$$s_c(\overrightarrow{eat\ hat}) = \frac{1}{2k}(\sum_{i=1}^{k} dist(\overrightarrow{eat} \odot \overrightarrow{hat}, \overrightarrow{eat} \odot \overrightarrow{neighbour_i}) + \sum_{j=1}^{k} dist(\overrightarrow{eat} \odot \overrightarrow{hat}, \overrightarrow{neighbour_j} \odot \overrightarrow{hat}))$$

We also experimented with substituting only for the noun or the verb, and in fact found that only taking neighbours for the verb yields better results:

$$s_c(\overrightarrow{eat\ hat}) = \frac{1}{k} \sum_{j=1}^{k} dist(\overrightarrow{eat} \odot \overrightarrow{hat}, \overrightarrow{neighbour_j} \odot \overrightarrow{hat})$$

To illustrate the method, consider the collocations *take breath* and *lend money*. The annotators assigned these phrases a compositionality score of 1 out of 6 and 6 out of 6, respectively, meaning that the former is non-compositional and the latter is compositional. The distances between the first ten verb-substituted phrases and the original phrase, together with the average distance, are shown in Table 1 and Table 2.

Substituting the verb in the non-compositional phrase yields semantically anomalous vectors, which leads to very small changes in the distance between it and the original phrase vector. This is a result of using pointwise multiplication, where overlapping components are stressed: since the vectors for *take* and *breath* have little overlap outside of

Neighbour	Dist
get breath	0.049
find breath	0.051
use breath	0.050
work breath	0.060
hold breath	0.094
run breath	0.079
carry breath	0.076
look breath	0.065
play breath	0.071
buy breath	0.100
AvgDist	0.069

Table 1: Example *take breath*

Neighbour	Dist
pay money	0.446
put money	0.432
bring money	0.405
provide money	0.442
owe money	0.559
sell money	0.404
cost money	0.482
look money	0.425
distribute money	0.544
offer money	0.428
AvgDist	0.457

Table 2: Example *lend money*

the idiomatic sense in *take breath*, its neighbour-substituted phrases also have little overlap, resulting in a smaller change in distance upon substitution. Conversely, substituting the verb in the compositional phrase yields meaningful vectors, putting them in locations in semantic vector space which are sufficiently far apart to distinguish them from the non-compositional cases.

4 Results

Results are given for the two methods of obtaining neighbours: via frequency-ranked WordNet neighbours and via vector space neighbours. The compositionality score was computed by using only the verb, only the noun, or both constituent neighbours in the substituted phrase vectors.

System	ρ_s
Venkatapathy and Joshi (2005)	0.447
McCarthy et al. (2007)	0.454
AvgDist VSM neighbours-both	0.131
AvgDist VSM neighbours-verb	0.420
AvgDist VSM neighbours-noun	0.245
AvgDist WN-ranked neighbours-both	0.165
AvgDist WN-ranked neighbours-verb	**0.461**
AvgDist WN-ranked neighbours-noun	0.169

Table 3: Spearman ρ_s results

The results are compared with the scores reported in Venkatapathy and Joshi (2005) and McCarthy et al. (2007), which were achieved using SVMs with a wide variety of features. Values of $1 \leq k \leq 20$ were tried. If a phrase has fewer than k neighbours because not enough neighbours have been found to co-occur with the other constituent, we use all of them. The results for $k = 20$ are reported here because that gave the best overall score. The dataset has an inter-annotator agreement of Kendall's τ of 0.61 and a Spearman ρ_s of 0.71 and all reported differences in values are highly significant. Table 3 gives the results.

Note that, even though the current approach is unsupervised (in terms of not having access to compositionality ratings during training, although it does rely on WordNet), it outperforms SVMs that require an ensemble of complex feature sets (some of which are also based on WordNet).

It is interesting to observe that the state-of-the-art performance is reached when only using the verb's neighbours to compute substituted phrase vectors. One might initially expect this not to be the case, since e.g. *eat trousers*, where the noun has been substituted, does not make a lot of sense either — which we would expect to be informative for determining compositionality. There are two possible explanations for this, which might be at play simultaneously: since our dataset consists of verb-object pairs, the verb constituent is always the head word of the phrase, and the dataset contains several so-called "light verbs", which have little semantic content of their own. Head words have been found to have a higher impact on compositionality scores for compound nouns: Reddy et al. (2011) weighted

the contribution of individual constituents in such a way that the modifier's contribution is included but is weighted less highly than the head's contribution, which led to an improvement in performance. Our results might be improved by weighting the contribution of constituent words in a similar fashion, and by more closely examining the impact of light verbs for the compositionality of a phrase.

5 Related Work

The past decade has seen extensive work on computational and statistical methods in detecting the compositionality of MWEs (Lin, 1999; Schone and Jurafsky, 2001; Katz and Giesbrecht, 2006; Sporleder and Li, 2009; Biemann and Giesbrecht, 2011). Many of these methods rely on distributional models and vector space models (Schütze, 1993; Turney and Pantel, 2010; Erk, 2012). Work has been done on different types of phrases, including work on particle verbs (McCarthy et al., 2003; Bannard et al., 2003), verb-noun collocations (Venkatapathy and Joshi, 2005; McCarthy et al., 2007), adjective-noun combinations (Vecchi et al., 2011) and noun-noun compounds (Reddy et al., 2011), as well as on languages other than English (Schulte im Walde et al., 2013). Recent developments in distributional compositional models (Widdows, 2008; Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Coecke et al., 2010; Socher et al., 2011) have opened up a number of possibilities for constructing vectors for phrases, which have also been applied to compositionality tests (Giesbrecht, 2009; Kochmar and Briscoe, 2013).

This paper takes that work a step further: by constructing phrase vectors and evaluating these vectors on a dataset of human compositionality ratings, we show that existing compositional models allow us to detect compositionality of multi-word expressions in a straightforward and intuitive manner.

6 Conclusion

We have presented a novel unsupervised approach that can be used to detect the compositionality of multi-word expressions. Our results show that the underlying intuition appears to be sound: substituting neighbours may lead to meaningful or meaningless phrases depending on whether or not the phrase is compositional. This can be formalized in vector space models to obtain compositionality scores by computing the average distance to the original phrase's substituted neighbour phrases. In this short focused research paper, we show that, depending on how we obtain neighbours, we are able to achieve a higher performance than that achieved by supervised methods which rely on a complex feature set and support vector machines.

Acknowledgments

This work has been supported by EPSRC grant EP/I037512/1. The authors would like to thank Diana McCarthy for providing the dataset; and Ed Grefenstette, Eva Maria Vecchi, Laura Rimell and Tamara Polajnar and the anonymous reviewers for their helpful comments.

References

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *Proceedings of the ACL 2003 Workshop on Multiword expressions: analysis, acquisition and treatment*, MWE 03.

M. Baroni and R. Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1183–1193.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Chris Biemann and Eugenie Giesbrecht. 2011. Disco-11: Proceedings of the workshop on distributional semantics and compositionality.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting Semantic Representations from Word Co-occurrence Statistics: Stop-lists, Stemming and SVD. *Behavior Research Methods*, 44:890–907.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. In J. van Bentham, M. Moortgat, and W. Buszkowski, editors, *Linguistic Analysis (Lambek Festschrift)*, volume 36, pages 345–384.

James Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Eugenie Giesbrecht. 2009. In search of semantic compositionality in vector spaces. In Sebastian Rudolph, Frithjof Dau, and SergeiO. Kuznetsov, editors, *Conceptual Structures: Leveraging Semantic Technologies*, volume 5662 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin Heidelberg.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, MWE '06, pages 12–19.

Ekaterina Kochmar and Ted Briscoe. 2013. Capturing Anomalies in the Choice of Content Words in Compositional Distributional Semantic Space. In *Recent Advances in Natural Language Processing*.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 317–324.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment - Volume 18*, MWE '03, pages 73–80.

Diana McCarthy, Sriram Venkatapathy, and Aravind Joshi. 2007. Detecting compositionality of verb-object combinations using selectional preferences. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 369–379.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *Proceedings of The 5th International Joint Conference on Natural Language Processing 2011 (IJCNLP 2011), Thailand*.

J.W. Reed, Y. Jiao, T.E. Potok, B.A. Klump, M.T. Elmore, and A.R. Hurson. 2006. TF-ICF: A new term weighting scheme for clustering dynamic data streams. In *Machine Learning and Applications, 2006. ICMLA '06. 5th International Conference on*, pages 258–263.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A Pain in the Neck for NLP. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '02, pages 1–15.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of Empirical Methods in Natural Language Processing*, EMNLP '01.

William Schuler and Aravind K. Joshi. 2011. Tree-rewriting models of multi-word expressions. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, MWE '11, pages 25–30.

Sabine Schulte im Walde, Stefan Müller, and Stephen Roller. 2013. Exploring Vector Space Models to Predict the Compositionality of German Noun-Noun Compounds. In *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics*, pages 255–265, Atlanta, GA.

Hinrich Schütze. 1993. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann.

Richard Socher, Cliff Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *The 28th International Conference on Machine Learning*, ICML 2011.

Caroline Sporleder and Linlin Li. 2009. 2009. unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, EACL '09.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (linear) maps of the impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, Oregon, USA, June. Association for Computational Linguistics.

Sriram Venkatapathy and Aravind K. Joshi. 2005. Measuring the relative compositionality of verb-noun (v-n) collocations by integrating features. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 899–906.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, Oxford.

Naive Bayes Word Sense Induction

Do Kook Choe
Brown University
Providence, RI
dc65@cs.brown.edu

Eugene Charniak
Brown University
Providence, RI
ec@cs.brown.edu

Abstract

We introduce an extended naive Bayes model for word sense induction (WSI) and apply it to a WSI task. The extended model incorporates the idea the words closer to the target word are more relevant in predicting its sense. The proposed model is very simple yet effective when evaluated on SemEval-2010 WSI data.

1 Introduction

The task of word sense induction (WSI) is to find clusters of tokens of an ambiguous word in an unlabeled corpus that have the same sense. For instance, given a target word "crane," a good WSI system should find a cluster of tokens referring to avian cranes and another referring to mechanical cranes. We believe that neighboring words contain enough information that these clusters can be found from plain texts.

WSI is related to word sense disambiguation (WSD). In a WSD task, a system learns a sense classifier in a supervised manner from a sense-labeled corpus. The performance of the learned classifier is measured on some unseen data. WSD systems perform better than WSI systems, but building labeled data can be prohibitively expensive. In addition, WSD systems are not suitable for newly created words, new senses of existing words, or domain-specific words. On the other hand, WSI systems can learn new senses of words directly from texts because these programs do not rely on a predefined set of senses.

In Section 2 we describe relevant previous work. In Section 3 and 4 we introduce the naive Bayes model for WSI and inference schemes for the model. In Section 5 we evaluate the model on SemEval-2010 data. In Section 6 we conclude.

2 Related Work

Yarowsky (1995) introduces a semi-supervised bootstrapping algorithm with two assumptions that rivals supervised algorithms: one-sense-per-collocation and one-sense-per-discourse. But this algorithm cannot easily be scaled up because for any new ambiguous word humans need to pick a few seed words, which initialize the algorithm. In order to automate the semi-supervised system, Eisner and Karakos (2005) propose an unsupervised bootstrapping algorithm. Their system tries many different seeds for bootstrapping and chooses the "best" classifier at the end. Eisner and Karakos's algorithm is limited in that their system is designed for disambiguating words that have only 2 senses.

Bayesian WSI systems have been developed by several authors. Brody and Lapata (2009) apply Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to WSI. They run a topic modeling algorithm on texts with some fixed number of topics that correspond to senses and induce a cluster by finding target words assigned to the same topic. Their system is evaluated on SemEval-2007 noun data (Agirre and Soroa, 2007). Lau et al. (2012) apply a nonparametric model, Hierarchical Dirichlet Processes (HDP), to SemEval-2010 data (Manandhar et al., 2010).

3 Model

Following Yarowsky (1995), we assume that a word in a document has one sense. Multiple occurrences of a word in a document refer to the same object or concept. The naive Bayes model is well suited for this one-sense-per-document assumption. Each document has one topic corresponding to the sense of the target word that needs disambiguation. Context words in a document are drawn from the conditional distribution of words given the sense. Context words are assumed to be independent from each other given

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1433–1437,
Seattle, Washington, USA, 18-21 October 2013.

the sense, which is far from being true yet effective.

3.1 Naive Bayes

The naive Bayes model assumes that every word in a document is generated independently from the conditional distribution of words given a sense, $p(w|s)$. The mathematical definition of the naive Bayes model is as follows:

$$p(\boldsymbol{w}) = \sum_s p(s, \boldsymbol{w}) = \sum_s p(s)p(\boldsymbol{w}|s) = \sum_s p(s) \prod_w p(w|s), \quad (1)$$

where $\boldsymbol{w}$ is a vector of words in the document. With the model, a new document can be easily labeled using the following classifier:

$$s' = \operatorname*{argmax}_s p(s) \prod_w p(w|s), \quad (2)$$

where s' is the label of the new document. In contrast to LDA-like models, it is easy to construct the closed form classifier from the model. The parameters of the model, $p(s)$ and $p(w|s)$, can be learned by maximizing the probability of the corpus, $p(\boldsymbol{d}) = \prod_d p(d) = \prod_{\boldsymbol{w}} p(\boldsymbol{w})$ where $\boldsymbol{d}$ is a vector of documents and $d = \boldsymbol{w}$.

3.2 Distance Incorporated Naive Bayes

Intuitively, context words near a target word are more indicative of its sense than ones that are farther away. To account for this intuition, we propose a more sophisticated model that uses the distance between a context word and a target word. Before introducing the new model, we define a probability distribution, $f(w|s)$, that incorporates distances as follows:

$$f(w|s) = \frac{p(w|s)^{l(w)}}{\sum_{w' \in \mathcal{W}} p(w'|s)^{l(w)}}, \quad (3)$$

where $l(w) = \frac{1}{dist(w)^x}$. $\mathcal{W}$ is a set of types in the corpus. x is a tunable parameter that takes nonnegative real values. With the new probability distribution, the model and the classifier become:

$$p(\boldsymbol{w}) = \sum_s p(s) \prod_w f(w|s) \quad (4)$$

$$s' = \operatorname*{argmax}_s p(s) \prod_w f(w|s), \quad (5)$$

where $f(w|s)$ replaces $p(w|s)$. The naive Bayes model is a special case; set $x = 0$. The new model puts more weight on context words that are close to the target word. The distribution of words that are farther away approaches the uniform distribution. $l(w)$ smoothes the distribution more as x becomes larger.

4 Inference

Given the generative model, we employ two inference algorithms to learn the sense distribution and word distributions given a sense. Expectation Maximization (EM) is a natural choice for the naive Bayes (Dempster et al., 1977). When initialized with random parameters, EM gets stuck at local maxima. To avoid local maxima, we use a Gibbs sampler for the plain naive Bayes to learn parameters that initialize EM.

5 Experiments

5.1 Data

We evaluate the model on SemEval-2010 WSI task data (Manandhar et al., 2010). The task has 100 target words, 50 nouns and 50 verbs. For each target word, there are training and test documents. Table 1 have details. The training and test data are plain texts without sense tags. For evaluation, the inferred sense labels are compared with human annotations. To tune some parameters we use the trial data of SemEval-2010. The trial data consists of training and test portions of 4 verbs. On average there are 137 documents for each target word in the training part of the trial data.

	Training	Testing	Senses (#)
All	879807	8915	3.79
Nouns	716945	5285	4.46
Verbs	162862	3630	3.12

Table 1: Details of SemEval-2010 data

5.2 Task

Participants induce clusters from the training data and use them to label the test data. Resources other than NLP tools for morphology and syntax such as lemmatizer, POS-tagger, and parser are not allowed. Tuning parameters and inducing clusters are only allowed during the training phase. After training, participants submit their sense-labeled test data to organizers.

LDA models are not compatible with the scoring rules for the SemEval-2010 competition, and that is the work against which we most want to compare. These rules require that training be done strictly before the testing is done. Note however that LDA requires learning the mixture weights of topics for each

individual document p(topic | document). These are, of course, learned during training. But the documents in the testing corpus have never been seen before, so clearly their topic mixture weights are not learned during training, and thus not learned at all. The way to overcome this is by training on both train and test documents, but this is exactly what SemEval-2010 forbids.

5.3 Implementation Details

The documents are tokenized and stemmed by Stanford tokenizer and stemmer. Stop words and punctuation in the training and test data are discarded. Words that occur at most 10 times are discarded from the training data. Context words within a window of 50 about a target word are used to construct a bag-of-words.

When a target word appears more than once in a document, the distance between that target word and a context word is ambiguous. We define this distance to be minimum distance between a context word and an instance of the target word. For example, the word "chip" appears 3 times. For

$\cdots$ of memory **chip**s . Currently , **chip**s are produced by shining light through a mask to produce an image on the **chip** , much as $\cdots$

Example 1: an excerpt from "chip" test data

a context word, e.g., "shining" there are three possible distances: 8 away from the first "chip," 4 away from the second "chip" and 11 away from the last "chip." We set the distance of "shining" from the target to 4.

We model each target word individually. We set α, a Dirichlet prior for senses, to 0.02 and β, a Dirichlet prior for contextual words, to 0.1 for the Gibbs sampler as in Brody and Lapata (2009). We initialize EM with parameters learned from the sampler. We run EM until the likehood changes less than 1%. We run the sampler 2000 iterations including 1000 iterations of burn-in: 10 samples at an interval of 100 are averaged. For comparison, we also evaluate EM with random initialization. All reported scores (described in Section 5.4) are averaged over ten different runs of the program.[1]

5.3.1 Tuning Parameters

Two parameters, the number of senses and x of the function $l(w)$, need to be determined before running the program. To find a good setting we do grid search on the trial data with the number of senses ranging from 2 to 5 and x ranging from 0 to 1.1 with an interval 0.1. Due to the small size of the training portion of the trial data, words that occur once are thrown out in the training portion. All the other parameters are as described in Section 5.3. We choose (4, 0.4), which achieves the highest supervised recall. See Table 2 for the performance of the model with various parameter settings. With a fixed value of x, a column is nearly unimodal in the number of senses and vice versa. $x = 0$ is not optimal and there is some noticeable difference between scores with optimal x and scores with $x = 0$.

5.4 Evaluation

We compare our system to other WSI systems and discuss two metrics for unsupervised evaluation (V-Measure, paired F-Score) and one metric for supervised evaluation (supervised recall). We refer to the true group of tokens as a gold class and to an induced group of tokens as a cluster. We refer to the model learned with the sampler and EM as NB, and to the model learned with EM only as NB0.

5.4.1 Short Descriptions of Other WSI Systems Evaluated on SemEval-2010

The baseline assigns every instance of a target word with the most frequent sense (MFS). UoY runs a clustering algorithm on a graph with words as nodes and co-occurrences between words as edges (Korkontzelos and Manandhar, 2010). Hermit approximates co-occurrence space with Random Indexing and applies a hybrid of k-means and Hierarchical Agglomerate Clustering to co-occurrence space (Jurgens and Stevens, 2010). NMF$_{lib}$ factors a matrix using nonnegative matrix factorization and runs a clustering algorithm on test instances represented by factors (Van de Cruys et al., 2011).

5.4.2 V-Measure

V-Measure computes the quality of induced clusters as the harmonic mean of two values, homogeneity and completeness. Homogeneity measures whether instances of a cluster belong to a single gold class. Completeness measures whether instances of a gold class belong to a cluster. V-Measure is between 0 and 1; higher is better. See Table 3 for details of V-Measure evaluation (#cl is the number of induced clusters).

With respect to V-Measure, NB performs much better than NB0. This holds for paired F-Score and supervised recall evaluations. The sampler improves the log-likelihood of NB by 3.8% on average (4.8% on nouns and 2.9% on verbs).

Pedersen (2010) points out that it is possible to increase the V-Measure of bad models by increasing

[1]Code used for experiments is available for download at `http://cs.brown.edu/~dc65/`.

#s \ x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1
2	74.73	**74.76**	74.41	74.57	74.06	74.07	74.18	74.33	74.14	74.22	74.15	74.52
3	74.60	74.71	75.21	75.46	75.21	75.57	**75.61**	75.32	75.53	75.56	74.98	74.79
4	74.52	75.06	74.97	75.14	**76.02**	75.51	75.74	75.51	75.59	75.51	75.37	75.35
5	73.40	73.88	74.93	75.13	74.79	74.68	74.71	74.49	75.11	74.94	74.86	**75.25**

Table 2: Performance of the model with various parameters: supervised recall on the trial data. The best value from each row is bold-faced. The scores are averaged over 100 runs.

VM(%)	all	nouns	verbs	#cl
NB	**18.0**	**23.7**	9.9	3.42
NB0	14.9	19.0	9.0	3.77
Hermit	16.2	16.7	**15.6**	10.78
UoY	15.7	20.6	8.5	11.54
NMF_{lib}	11.8	13.5	9.4	4.80
MFS	0.0	0.0	0.0	1.00

Table 3: Unsupervised evaluation: V-Measure

the number of clusters. But increasing the number of clusters harms paired F-Score, which results in bad supervised recalls. NB attains a very high V-Measure with few induced clusters, which indicates that those clusters are high quality. Other systems use more induced clusters but fail to attain the V-Measure of NB.

5.4.3 Paired F-Score

Paired F-Score is the harmonic mean of paired recall and paired precision. Paired recall is fraction of pairs belonging to the same gold class that belong to the same cluster. Paired precision is fraction of pairs belonging to the same cluster that belong to the same class. See Table 4 for details of paired F-Score evaluation.

As with V-Measure, it is possible to attain a high paired F-Score by producing only one cluster. The baseline, MFS, attains 100% paired recall, which together with the poor performance of WSI systems makes its paired F-Score difficult to beat. V-Measure and paired F-Score are meaningful when systems produce about the same numbers of clusters as the numbers of classes and attain high scores on these metrics.

FS(%)	all	nouns	verbs	#cl
MFS	**63.5**	**57.0**	**72.7**	1.00
NB	52.9	52.5	53.5	3.42
NB0	46.8	47.4	46.0	3.77
UoY	49.8	38.2	66.6	11.54
NMF_{lib}	45.3	42.2	49.8	4.80
Hermit	26.7	24.4	30.1	10.78

Table 4: Unsupervised evaluation: paired F-Score

5.4.4 Supervised Recall

For the supervised task, the test data is split into two groups: one for mapping clusters to classes and the other for standard WSD evaluation. 2 different split schemes (80% mapping, 20% evaluation and 60% mapping, 40% evaluation) are evaluated. 5 random splits are averaged for each split scheme. Mapping is induced automatically by the program provided by organizers. See Table 5 for details of supervised recall evaluation (#s is the average number of classes mapped from clusters).[2]

SR(%)	all	nouns	verbs	#s
NB	**65.4**	**62.6**	69.5	1.72
NB0	63.5	59.8	69.0	1.76
NMF_{lib}	62.6	57.3	**70.2**	1.82
UoY	62.4	59.4	66.8	1.51
MFS	58.7	53.2	66.6	1.00
Hermit	58.3	53.6	65.3	2.06

Table 5: Supervised evaluation: supervised recall, 80% mapping and 20% evaluation

Overall our system performs better than other systems with respect to supervised recall. When a system has higher V-Measure and paired F-Score on nouns than another system, it achieves a higher supervised recall on nouns too. However, this behavior is not observed on verbs. For example, NB has higher V-Measure and paired F-Score on verbs than NMF_{lib} but NB attains a lower supervised recall on verbs than NMF_{lib}. It is difficult to see which verbs clusters are better than some other clusters.

6 Conclusion

Of the four SemEval-2010 evaluation metrics, and restricting ourselves to systems obeying the evaluation conditions for that competition, our new model achieves new best results on three. The exception is paired F-Score. As we note earlier, this metric tends to assign very high scores when every word receives only one sense, and our model is bested by the baseline system that does exactly that.

[2] 60-40 split is omitted here due to almost identical result.

If we loosen possible comparison systems, the LDA/HDP model of Lau et al. (2012) achieves superior numbers to ours for the two supervised metrics, but at the expense of requiring LDA type processing on the test data, something that the SemEval organizers ruled out, presumably with the reasonable idea that such processing would not be feasible in the real world. More generally, their system assigns many senses (about 10) to each word, and thus no-doubt does poorly on the paired F-Score (they do not report results on V-Measure and paired F-Score).

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 7–12. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 103–111, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Jason Eisner and Damianos Karakos. 2005. Bootstrapping without the boot. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 395–402, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Jurgens and Keith Stevens. 2010. Hermit: Flexible clustering for the semeval-2 wsi task. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 359–362. Association for Computational Linguistics.

Ioannis Korkontzelos and Suresh Manandhar. 2010. Uoy: Graphs of unambiguous vertices for word sense induction and disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 355–358. Association for Computational Linguistics.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601. Association for Computational Linguistics.

Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68. Association for Computational Linguistics.

Ted Pedersen. 2010. Duluth-wsi: Senseclusters applied to the sense induction task of semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 363–366. Association for Computational Linguistics.

Tim Van de Cruys, Marianna Apidianaki, et al. 2011. Latent semantic word sense induction and disambiguation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT)*, pages 1476–1485.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

The VerbCorner Project: Toward an Empirically-Based Semantic Decomposition of Verbs

Joshua K. Hartshorne
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139, USA
jkhartshorne@gmail.com

Claire Bonial, Martha Palmer
Department of Linguistics
University of Colorado at Boulder
Hellems 290, 295 UCB
Boulder, CO 80309, USA
{CBonial, MPalmer}@colorado.edu

Abstract

This research describes efforts to use crowdsourcing to improve the validity of the semantic predicates in VerbNet, a lexicon of about 6300 English verbs. The current semantic predicates can be thought of semantic primitives, into which the concepts denoted by a verb can be decomposed. For example, the verb *spray* (of the Spray class), involves the predicates MOTION, NOT, and LOCATION, where the event can be decomposed into an AGENT causing a THEME that was originally not in a particular location to now be in that location. Although VerbNet's predicates are theoretically well-motivated, systematic empirical data is scarce. This paper describes a recently-launched attempt to address this issue with a series of human judgment tasks, posed to subjects in the form of games.

1 Introduction

One key application of Natural Language Processing (NLP) is meaning extraction. Of particular importance is propositional meaning: To understand "Jessica sprayed paint on the wall," it is not enough to know who Jessica is, what paint is, and where the wall is, but that, by the end of the event, some quantity of paint that was not previously on the wall now is. One must extract not only meanings for individual words but also the relations between them.

One option is to learn these relations in a largely bottom-up, data-driven fashion (Chklovski and Pantel, 2004; Poon and Domingos, 2009). For instance, Poon and Domingos (2009) first extracts dependency trees, converts those into quasi-logical form, recursively induces lambda expressions from them, and uses clustering to derive progressively abstract knowledge.

An alternative is to take a human-inspired approach, mapping the linguistic input onto the kinds of representations that linguistic and psychological research suggests are the representations employed by humans. While the exact characterization of meaning (and by extension, thought) remains an area of active research in the cognitive sciences (Margolis and Laurence, 1999), decades of research in linguistics and psychology suggests that much of the meaning of a sentence – as well as its syntactic structure – can be accounted for by invoking a small number of highly abstract semantic features (usually represented as predicates), such as causation, agency, basic topological relations, and directed motion (Ambridge et al., 2013; Croft, 2012; Jackendoff, 1990; Levin and Rappaport Hovav, 2005; Pesetsky, 1995; Pinker, 1989). For instance, a given verb can appear in some syntactic frames (*Sally broke the vase. Sally broke the vase with the hammer. The vase broke.*) and not others (**Sally broke the vase to the floor. *Sally broke John the vase.*). When verbs are classified according to the syntactic frames they can appear in, most if not all the verbs in a class involve the same set of abstract semantic features.[1]

Interestingly, roughly these same features (causation, etc.) have been singled out by developmental psychologists as part of "core knowledge" – a set of early-learned or perhaps innate concepts upon which

[1] Whether all verbs in a class share the same abstract predicates or merely most is an area of active research (Levin and Rappaport Hovav, 2005).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1438–1442,
Seattle, Washington, USA, 18-21 October 2013.

the rest of cognition is built (Spelke and Kinzler, 2007). Thus these semantic features/predicates may be not only crucial to describing linguistic meaning but may be central organizing principles for a human's (reasonably successful) thinking about and conceptualization of the world. As such, they provide a potentially rewarding target for NLP.

2 VerbNet

2.1 Overview and Structure

Perhaps the most comprehensive implementation of this approach appears in VerbNet (Kipper et al., 2008; based on Levin, 1993). VerbNet classifies verbs based on the syntactic frames they can appear in, providing a semantic description of each frame for each class. An example entry is shown below:

Syntactic Frame NP V NP PP.DESTINATION
Example Jessica sprayed the wall.
Syntax AGENT V THEME {+LOC|+DEST_CONF} DESTINATION
Semantics MOTION(DURING(E), THEME)
NOT(PREP(START(E), THEME, DESTINATION))
PREP(END(E), THEME, DESTINATION)
CAUSE(AGENT, E)

The "Syntactic Frame" provides a flat syntactic parse. "Syntax" provides semantic role labels for each of the NPs and PPs, which are invoked in "Semantics". VerbNet decomposes the semantics of this sentence into four separate predicates: 1) the THEME (the paint) moves doing the event E; 2) at the start of the event E, the THEME (the paint) is not at the DESTINATION (on the wall), whereas 3) at the end of the event E, the THEME (the paint) is at the DESTINATION (on the wall), and; 4) the event is caused by the AGENT (Sally). Note that this captures only the core aspects of semantics shared by all verbs in the class; differences between verbs in the same class (e.g., *spray* vs. *splash*) are omitted.

Importantly, the semantics of the sentence is dependent on both the matrix verb (paint) and the syntactic frame. Famously, when inserted in the slightly different frame NP V NP.DESTINATION PP.THEME – "Sally sprayed the wall with paint" – "spray" entails that destination (the wall) is now fully painted, an entailment that does not follow in the example above (Pinker, 1989).

2.2 Uses and Limitations

VerbNet has been used in a variety of NLP applications, such as semantic role labeling (Swier and Stevenson, 2004), inferencing (Zaenen et al., 2008), verb classification (Joanis et al., 2008), and information extraction (Maynard, Funk, and Peters, 2009).

While such applications have been successful thus far, an important constraint on how well VerbNet-based NLP applications can be expected to perform is the accuracy of the semantics encoded in VerbNet. Here, several issues arise. Leaving aside miscategorized verbs and other inaccuracies, as noted above VerbNet assumes that all verbs in the same class share the same core predicates, which may or may not be empirically justified. Given the number of semantic predicates (146),[2] verb entries (6580), and unique verb lemmas (6284) it is not feasible for a single research team to check, particularly since after a certain number of verbs, intuitions become less clear. In any case, it may not be ideal to rely solely on the intuitions of invested researchers, whose intuitions about subtle judgments may be clouded by theoretical commitments (Gibson and Federenko, 2013); the only way to ensure this is not the case is through independent validation. Unfortunately, of the 280 verb classes in VerbNet, this has been done for only a few (cf Ambridge et al., 2013).

3 VerbCorner

The VerbCorner project was designed to address these issues by crowd-sourcing the semantic judgments online (gameswithwords.org/VerbCorner/). Several previous projects have successfully crowd-sourced linguistic annotations, such as Phrase Detectives, where volunteers have contributed 2.5 million judgments on anaphoric relations (Poesio et al., 2012). Below, we outline the VerbCorner project and describe one specific annotation task in detail.

3.1 Developing Semantic Annotation Tasks

Collecting accurate judgments on subtle questions from naive participants with limited metalinguistic

[2]Note that these vary in applicability from those specific to a small number of verbs (CHARACTERIZE, CONSPIRE) to those frequently invoked (BEGIN, EXIST).

skills is difficult. Rare is the non-linguist who can immediately answer the question, "Does the verb 'throw,' when used transitively, entail a change of location on the part of its THEME?" Thus, we began by developing tasks that isolate semantic features in a way accessible to untrained annotators.

We converted the metalinguistic judgments ("Does this verb entail this abstract predicate?") into real-world problems, which previous research suggests should be easier (Cosmides and Tooby, 1992). Each judgment tasks involved a fanciful backstory. For instance, in "Simon Says Freeze", a task designed to elicit judgments about movement, the Galactic Overlord (Simon) decrees "Galactic Stay Where You Are Day," during which nobody is allowed to move from their current location. Participants read descriptions of events and decide whether anyone violated the rule. In "Explode on Contact", designed to elicit judgments about physical contact, objects and people explode when they touch one another. The participant reads descriptions of events and decides whether anything has exploded.[3]

Each task was piloted until inter-coder reliability was acceptably high and the modal response nearly always corresponded with researcher intuitions. As such, these tasks cannot be used to establish whether researcher intuitions for the pilot stimuli are correct (this would be circular); however, there is no guarantee that agreement with the researcher will generalize to new items (the pilot stimuli cover a trivial proportion of all verbs in VerbNet).

3.2 Crowd-sourcing Semantic Judgments

The pilot experiments showed that it is possible to elicit reliable semantic judgments corresponding to VerbNet predicates from naive participants (see section 3.3). At the project website, volunteers choose one of the tasks from a list and begin tagging sentences. The sentences are sampled smartly, avoiding sentences already tagged by that volunteer and biased in favor of of the sentences with the fewest judgments so far. Rather than assessing annotator quality through gold standard trials with known answers (which wastes data – the answers to these trials are known), approximately 150 sentences were chosen to be "over-sampled." As the volunteer tags sentences, approximately one out of every five are from this over-sampled set until that volunteer has tagged all of them. This guarantees that any given volunteer will have tried some sentences targeted by many other volunteers, allowing inter-annotator agreement to be used to assess annotator quality.

Following the example of Zooniverse (zooniverse.org), a popular "Citizen Science" platform, volunteers are encouraged but required to register (requiring registration prior to seeing the tasks was found to be a significant barrier to entry). Registration allows collecting linguistic and educational background from the volunteer, and also makes it possible to track the same volunteer across sessions.

Multiple gamification elements were incorporated into VerbCorner in order to recruit and motivate volunteers. Each task has a leaderboard, where the volunteer can see his/her rank out of all volunteers in terms of number of contributions made. In addition, there is a general leaderboard, which sums across tasks. Volunteers can earn badges, displayed on their homepage, for answering certain numbers of questions in each task. Finally, at random intervals bonus points are awarded, with the explanation for the bonus points tailored to the task's backstory.

VerbCorner was launched on May 21, 2013. After six weeks, 555 volunteers had provided at least one annotation, for a total of 39,274 annotations, demonstrating the feasibility of collecting large numbers of annotations through this method.

3.3 Case Study: Equilibrium

"Equilibrium" was designed to elicit judgments about application of force, frequently argued to be a core semantic feature in the sense discussed above (Pinker, 1989). The backstory involves the "Zen Dimension," in which nobody is allowed to exert force on anything else. The participant reads descriptions of events (*Sally sprayed paint onto the wall*) and decides whether they would be allowable in the Zen Dimension – and, in particular, which participants in the event are illegally applying force.

In order to minimize unwanted effects of world

[3]Note that each task is designed to elicit judgments about entailments – things that must be true rather than are merely likely to be true. If John greeted Bill, they might have come into contact (e.g., by shaking hands), but perhaps they did not. Previous work suggests that it is entailments that matter, particularly for explaining the syntactic behavior of verbs (Levin and Rappaport Hovav, 2005)

knowledge, the verb's arguments are replaced with nonsense words or randomly chosen proper names (*Sally sprayed the dax onto the blicket*). In the context of the story, this is explained as necessary anonymization: You are a government official determining whether certain activities are allowable, and ensuring anonymity is an important safeguard against favoritism and corruption. An alternative wouod be to use multiple different content words, randomly chosen for each annotator. However, this greatly increases the number of annotators needed and quickly becomes infeasible.

3.3.1 Pilot Results

The task was piloted on 138 sentences, which comprised all possible syntactic frames for three verbs from each of five verb classes in VerbNet. After two rounds of piloting (between the first and second, wording in the backstory was adjusted for clarity based on pilot subject feedback and results), Kripp's alpha reached .76 for 8 annotators, which represents a reasonably high level of inter-annotator agreement. Importantly, the modal response matched the intuitions of the researchers in 137 of 138 cases.[4]

3.3.2 Preliminary VerbCorner Results

"Equillibrium" was one of the first tasks posted on VerbCorner, with data currently being collected on 12 of the 280 VerbNet classes, for a total of 5,171 sentences. As of writing, 414 users have submitted 14,294 judgments. Individual annotators annotated anywhere from 1 to 195 sentences (mean=8, median=4). While most sentences have relatively few judgments, each of the 194 over-sampled sentences has between 15 and 20 judgments.[5]

Comparing the modal response with the researchers' intuitions resulted in a match for 184 of 194 sentences. In general, where the modal response did not match researcher intuitions, the modal response was itself not popular, comprising an average of 53% of responses, compared with an average of 77% where the modal response matched researcher intuitions. Thus, these appear to be cases of disagreement, either because the correct intuition requires more work to obtain or because of differences across idiolects (at the moment, there is no obvious pattern as to which sentences caused difficulty, but the sample size is small). Thus, follow-up investigation of sentences with little inter-coder agreement may be warranted.

4 Conclusion and Future Work

Data-collection is ongoing. VerbNet identifies approximately 150 different semantic predicates. Annotating every verb in each of its syntactic frames for each semantic predicate would take many millions of judgments. However, most of the semantic predicates employed in VerbNet are very narrow in scope and only apply to a few classes. Thus, we have begun with broad predicates that are thought to apply to many verbs and are adding progressively narrower predicates as work progresses. At the current rate, we should complete annotation for the half-dozen most frequent semantic predicates in the space of a year.

Future work will explore using an individual annotator's history across trials to weight that user's contributions, something that VerbCorner was specifically designed to allow (see above). How to assess annotator quality without gold standard data is an active area of research (Passonneau and Carpenter, 2013; Rzhetsky, Shatkay and Wilbur, 2009; Whitehill et al., 2009). For instance, Whitehill and colleagues (2009) provide an algorithm for jointly estimating both annotator quality and annotation difficulty (including the latter is important because some annotators will have low agreement with others due to their poor luck in being assigned difficult-to-annotate sentences). This algorithm is shown to outperform using the modal response.

Note that this necessarily biases against annotators with few responses. In our case study above, excluding annotators who contributed small numbers of annotations led to progressively worse match to researcher intuition, suggesting that the loss in data

[4]The remaining case was "The crose smashed sondily." for which four pilot subjects thought involved the crose applying force – matching researcher intuition – and four thought did not involve any application of force, perhaps interpreting the sentence was a passive.

[5]These are the same 15 verbs used in the piloting. The number of sentences is larger in order to test a wider range of possible arguments. In particular, wherever appropriate, separate sentences were constructed using animate and inanimate arguments. Compare *Sally sprayed the dax onto Mary* and *Sally sprayed the dax onto the blicket*.

caused by excluding these annotations may not be worth the increased confidence in annotation quality. Future research will be needed to assess this trade-off.

The above work shows the feasibility of crowd-sourcing VerbNet semantic entailments, as has been shown for a handful of other linguistic judgments (Artignan, Hascoet and Lafourcade, 2009; Poesio et al., 2012; Venhuizen et al., 2013). There are many domains in which gold standard human judgments are scarce; crowd-sourcing has considerable potential at addressing this need.

References

B. Ambridge, J. M. Pine, C. F. Rowland, F. Chang, and A. Bidgood. 2013. The retreat from overgeneralization in child language acquisition: Word learning, morphology, and verb argument structure. *Wiley Interdisciplinary Reviews: Cognitive Science*. 4:47-62.

G. Artignan, M. Hascoet, and M. Lafourcade. 2009. Mutliscale visual analysis of lexical networks. *Proceedings of the 13th International Conference on Information Visualisation*. Barcelona, Spain.

T. Chklovski and P. Pantel. 2004. VerbOcean: Mining the Web for fine-grained semantic relations. *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*. Barcelona, Spain.

L. Cosmides and J. Tooby. 1992. Cognitive adaptations for social exchange. in *The Adapted Mind*. (J. Barkow, L. Cosmides, and J. Tooby, Eds.) Oxford University Press, Oxford, UK.

W. Croft. 2012. *Verbs: Aspect and Argument Structure*. Oxford University Press, Oxford, UK.

D. R. Dowty. 1991. Thematic proto-roles and argument selection. *Language*. 67:547-619.

E. Gibson and E. Fedorenko. 2013. The need for quantitative methods in syntax and semantics research. *Language and Cognitive Processes*. 28(1-2):88–124.

R. Jackendoff. 1990. *Semantic Structures*. The MIT Press, Cambridge, MA.

E. Joanis, S. Stevenson, and D. James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*. 14(3):337-367.

K. Kipper, A. Korhonen, N. Ryant and M. Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation Journal*, 42:21–40

E. Margolis and S. Laurence 1999. *Concepts: Core Readings*. The MIT Press, Cambridge, MA.

B. Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago.

B. Levin and M. Rappaport Hovav. 2005. *Argument Realization*. Cambridge University Press, Cambridge, UK.

D. Maynard, A. Funk, and W. Peters. 2009. Using lexico-syntactic ontology design patterns for ontology creation and population. *Proceedings of Workshop on Ontology Patterns (WOP 2009)*. Washington, DC

R. J. Passonneau and B. Carpenter 2013. The benefits of a model of annotation. 7th Linguistic Annotation Workshop and Interoperability with Discourse. Sofia, Bulgaria.

D. Pesetsky. 1995. *Zero Syntax: Experiencers and Cascades*. The MIT Press, Cambridge, MA.

S. Pinker. 1989. *Learnability and Cognition*. The MIT Press, Cambridge, MA.

M. Poesio, J. Camberlain, U. Kruschwitz, L. Robaldo, and L. Ducceschi. 2012. The Phrase Detective Multilingual Corpus, Release 0.1. *Proceedings of the Collaborative Resource Development and Delivery Workshop*. Istanbul, Turkey

H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore.

A. Rzhetsky, H. Shatkay, and W. J. Wilbur. 2009. How to get the most out of your curation effort. *PLoS Computational Biology*, 5(5):1–13.

E. S. Spelke and K. D. Kinzler. 2007. Core knowledge. *Developmental Science*, 10(1):89–96.

R. Swier and S. Stevenson. 2004. Unsupervised semantic role labeling. *Proceedings of the Generative Lexicon Conference, GenLex-09*. Pisa, Italy.

N. Venhuizen, V. Basile, K. Evang, and J. Bos. 2013. Gamification for word sense labeling. *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*. Potsdam, Germany

J. Whitehill, P. Ruvolo, T. F. Wu, J. Bergsma. and J. Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems, 22*. Vancouver, Canada

A. Zaenen, C. Condoravdi, and D G. Bobrow. 2008. The encoding of lexical implications in VN. *Proceedings of LREC 2008*. Morocco

Where *Not* to Eat? Improving Public Policy by Predicting Hygiene Inspections Using Online Reviews

Jun Seok Kang† **Polina Kuznetsova**† **Michael Luca**‡ **Yejin Choi**†
†Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
{junkang,pkuznetsova,ychoi}
@cs.stonybrook.edu

‡Harvard Business School
Soldiers Field Road
Boston, MA 02163
mluca@hbs.edu

Abstract

This paper offers an approach for governments to harness the information contained in social media in order to make public inspections and disclosure more efficient. As a case study, we turn to restaurant hygiene inspections – which are done for restaurants throughout the United States and in most of the world and are a frequently cited example of public inspections and disclosure. We present the first empirical study that shows the viability of statistical models that learn the mapping between textual signals in restaurant reviews and the hygiene inspection records from the Department of Public Health. The learned model achieves over 82% accuracy in discriminating severe offenders from places with no violation, and provides insights into salient cues in reviews that are indicative of the restaurant's sanitary conditions. Our study suggests that public disclosure policy can be improved by mining public opinions from social media to target inspections and to provide alternative forms of disclosure to customers.

1 Introduction

Public health inspection records help customers to be wary of restaurants that have violated health codes. In some counties and cities, e.g., LA, NYC, it is required for restaurants to post their inspection grades at their premises, which have shown to affect the revenue of the business substantially (e.g., Jin and Leslie (2005), Henson et al. (2006)), thereby motivating restaurants to improve their sanitary practice. Other studies have reported correlation between the frequency of unannounced inspections per year, and the average violation scores, confirming the regulatory role of inspections in improving the hygiene quality of the restaurants and decreasing food-borne illness risks (e.g., Jin and Leslie (2003), Jin and Leslie (2009), Filion and Powell (2009), NYC-DoHMH (2012)).

However, one practical challenge in the current inspection system is that the department of health has only limited resources to dispatch inspectors, leaving out a large number of restaurants with unknown hygiene grades. We postulate that online reviews written by the very citizens who have visited those restaurants can serve as a proxy for predicting the likely outcome of the health inspection of any given restaurant. Such a prediction model can complement the current inspection system by enlightening the department of health to make a more informed decision when allocating inspectors, and by guiding customers when choosing restaurants.

Our work shares the spirit of recently emerging studies that explores social media analysis for public health surveillance, in particular, monitoring influenza or food-poisoning outbreaks from microblogs (e.g., Aramaki et al. (2011), Sadilek et al. (2012b), Sadilek et al. (2012a), Sadilek et al. (2013), Lamb et al. (2013), Dredze et al. (2013), von Etter et al. (2010)). However, no prior work has examined the utility of review analysis as a predictive tool for accessing hygiene of restaurants, perhaps because the connection is not entirely conspicuous: after all, customers are neither familiar with inspection codes, nor have the full access to the kitchen, nor have been asked to report on the hygiene aspects of their expe-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1443–1448,
Seattle, Washington, USA, 18-21 October 2013.

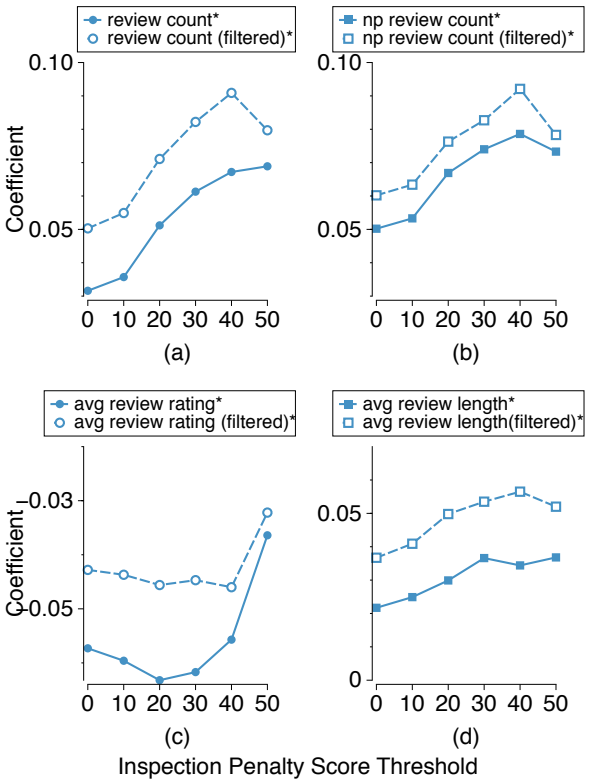


Figure 1: Spearman's coefficients of factors & inspection penalty scores. '*': statistically significant ($p \leq 0.05$)

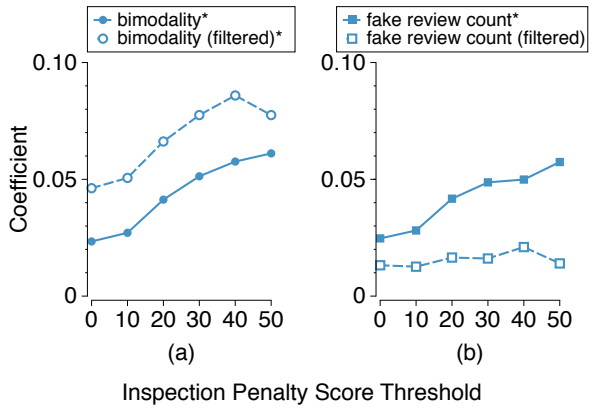


Figure 2: Spearman's coefficients of factors & inspection penalty scores. '*': statistically significant ($p \leq 0.05$)

rience.

In this work, we report the first empirical study demonstrating the utility of review analysis for predicting health inspections, achieving over 82% accuracy in discriminating severe offenders from places with no violation, and find predictive cues in reviews that correlate with the inspection results.

2 Data

We scraped entire reviews written for restaurants in Seattle from Yelp over the period of 2006 to 2013.[1] The inspection records of Seattle is publicly available at `www.datakc.org`. More than 50% of the restaurants listed under Yelp did not have inspection records, implying the limited coverage of inspections. We converted street addresses into canonical forms when matching restaurants between Yelp and inspection database. After integrating reviews with inspection records, we obtained about 13k inspections over 1,756 restaurants with 152k reviews. For each restaurant, there are typically several inspection records. We defined an "inspection period" of each inspection record as the period of time starting from the day after the previous inspection to the day of the current inspection. If there is no previous inspection, then the period stretches to the past 6 months in time. Each inspection period corresponds to an instance in the training or test set. We merge all reviews within an inspection period into one document when creating the feature vector.

Note that non-zero penalty scores may not necessarily indicate alarming hygiene issues. For example, violating codes such as *"proper labeling"* or *"proper consumer advisory posted for raw or undercooked foods"* seem relatively minor, and unlikely to be noted and mentioned by reviewers. Therefore, we focus on restaurants with severe violations, as they are exactly the set of restaurants that inspectors and customers need to pay the most attention to. To define restaurants with "severe violations" we experiment with a varying threshold t, such that restaurants with score $\geq t$ are labeled as *"unhygienic"*.[2]

3 Correlates of Inspection Penalty Scores

We examine correlation between penalty scores and several statistics of reviews:

I. Volume of Reviews:

[1] Available at `http://www.cs.stonybrook.edu/~junkang/hygiene/`

[2] For restaurants with *"hygienic"* labels, we only consider those without violation, as there are enough number of such restaurants to keep balanced distribution between two classes.

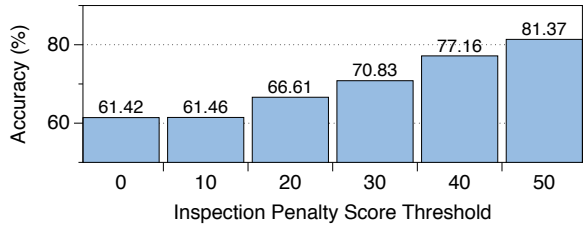


Figure 3: Trend of penalty score thresholds & accuracies.

- *count of all reviews*
- *average length of all reviews*

II. Sentiment of Reviews: We examine whether the overall sentiment of the customers correlates with the hygiene of the restaurants based on following measures:

- *average review rating*
- *count of negative (≤ 3) reviews*

III. Deceptiveness of Reviews: Restaurants with bad hygiene status are more likely to attract negative reviews, which would then motivate the restaurants to solicit fake reviews. But it is also possible that some of the most assiduous restaurants that abide by health codes strictly are also diligent in soliciting fake positive reviews. We therefore examine the correlation between hygiene violations and the degree of deception as follows.

- *bimodal distribution of review ratings*

  The work of Feng et al. (2012) has shown that the shape of the distribution of opinions, overtly skewed bimodal distributions in particular, can be a telltale sign of deceptive reviewing activities. We approximately measure this by computing the variance of review ratings.

- *volume of deceptive reviews based on linguistic patterns*

  We also explore the use of deception classifiers based on linguistic patterns (Ott et al., 2011) to measure the degree of deception. Since no deception corpus is available in the restaurant domain, we collected a set of fake reviews and truthful reviews (250 reviews for each class), following Ott et al. (2011).[3]

[3] 10 fold cross validation on this dataset yields 79.2% accuracy based on unigram and bigram features.

Features	Acc.	MSE	SCC
-	*50.00	0.500	-
review count	*50.00	0.489	0.0005
np review count	*52.94	0.522	0.0017
cuisine	*66.18	0.227	0.1530
zip code	*67.32	0.209	0.1669
avrg. rating	*57.52	0.248	0.0091
inspection history	*72.22	0.202	0.1961
unigram	78.43	0.461	0.1027
bigram	*76.63	0.476	0.0523
unigram + bigram	82.68	0.442	0.0979
all	81.37	0.190	0.2642

Table 1: Feature Compositions & Respective Accuracies, Respective Mean Squared Errors(MSE) & Squared Correlation Coefficients (SCC), np=non-positive

Filtering Reviews: When computing above statistics over the set of reviews corresponding to each restaurant, we also consider removing a subset of reviews that might be dubious or just noise. In particular, we remove reviews that are too far away (delta ≥ 2) from the average review rating. Another filtering rule can be removing all reviews that are classified as deceptive by the deception classifier explained above. For brevity, we only show results based on the first filtering rule, as we did not find notable differences in different filtering strategies.

Results: Fig 1 and 2 show Spearman's rank correlation coefficient with respect to the statistics listed above, with and without filtering, computed at different threshold cutoffs $\in \{0, 10, 20, 30, 40, 50\}$ of inspection scores. Although coefficients are not strong,[4] they are mostly statistically significant with $p \leq 0.05$ (marked with '*'), and show interesting contrastive trends as highlighted below.

In Fig 1, as expected, average review rating is negatively correlated with the inspection penalty scores. Interestingly, all three statistics corresponding to the volume of customer reviews are positively correlated with inspection penalty. What is more interesting is that if potentially deceptive reviews are filtered, then the correlation gets stronger, which suggests the existence of deceptive reviews covering up unhappy customers. Also notice that correlation is

[4] Spearman's coefficient assumes monotonic correlation. We suspect that the actual correlation of these factors and inspection scores are not entirely monotonic.

Hygienic gross, mess, sticky, smell, restroom, dirty
Basic Ingredients: beef, pork, noodle, egg, soy, ramen, pho,
Cuisines Vietnamese, Dim Sum, Thai, Mexican, Japanese, Chinese, American, Pizza, Sushi, Indian, Italian, Asian
Sentiment: cheap, never,
Service & Atmosphere cash, worth, district, delivery, think, really, thing, parking, always, usually, definitely - door: "The wait is always out the *door* when I actually want to go there", - sticker: "I had *sticker* shock when I saw the prices.", - student: "heap, large portions and tasty = the perfect *student* food!", - the size: "i was pretty astonished at *the size* of all the plates for the money.", - was dry: "The beef *was dry*, the sweet soy and anise-like sauce was TOO salty (almost inedible).", - pool: "There are *pool* tables, TV airing soccer games from around the globe and of course - great drinks!"

Table 2: Lexical Cues & Examples - Unhygienic (dirty)

Hygienic:
Cooking Method & Garnish: brew, frosting, grill, crush, crust, taco, burrito, toast **Healthy or Fancier Ingredients**: celery, calamity, wine, broccoli, salad, flatbread, olive, pesto
Cuisines : Breakfast, Fish & Chips, Fast Food, German, Diner, Belgian, European, Sandwiches, Vegetarian
Whom & When: date, weekend, our, husband, evening, night
Sentiment: lovely, yummy, generous, friendly, great, nice
Service & Atmosphere: selection, attitude, atmosphere, ambiance, pretentious

Table 3: Lexical Cues & Examples - Hygienic (clean)

generally stronger when higher cutoffs are used (x-axis), as expected. Fig 2 looks at the relation between the deception level and the inspection scores more directly. As suspected, restaurants with high penalty scores show increased level of deceptive reviews.

Although various correlates of hygiene scores examined so far are insightful, these alone are not informative enough to be used as a predictive tool, hence we explore content-based classification next.

4 Content-based Prediction

We examine the utility of the following features:

Features based on customers' opinion:

1. Aggregated opinion: average review rating
2. Content of the reviews: unigram, bigram

Features based on restaurant's metadata:

3. Cuisine: e.g., Thai, Italian, as listed under Yelp
4. Location: first 5 digits of zip code
5. Inspection History: a boolean feature ("hygienic" or "unhygienic"), a numerical feature (previous penalty score rescaled $\in [0, 1]$), a numeric feature (average penalty score over all previous inspections)
6. Review Count
7. Non-positive Review Count

Classification Results We use liblinear's SVM (Fan et al., 2008) with L1 regularization and 10 fold cross validation. We filter reviews that are farther than 2 from the average rating. We also run Support Vector Regression (SVR) using liblinear. Fig 3 shows the results. As we increase the threshold, the accuracy also goes up in most cases. Table 1 shows feature ablation at threshold $t = 50$, and '*' denotes statistically significant ($p \leq 0.05$) difference over the performance with all features based on student t-test.

We find that metadata information of restaurants such as location and cuisine alone show good predictive power, both above 66%, which are significantly higher than the expected accuracy of random guessing (50%).

Somewhat unexpected outcome is aggregated opinion, which is the average review rating during the corresponding inspection period, as it performs not much better than chance (57.52%). This result suggest that the task of hygiene prediction from reviews differs from the task of sentiment classification of reviews.

Interestingly, the inspection history feature alone is highly informative, reaching accuracy upto 72%, suggesting that the past performance is a good predictor of the future performance.

Textual content of the reviews (unigram+bigram) turns out to be the most effective features, reaching upto 82.68% accuracy. Lastly, when all the features

are combined together, the performance decreases slightly to 81.37%, perhaps because n-gram features perform drastically better than all others.

4.1 Insightful Cues

Table 2 and 3 shows representative lexical cues for each class with example sentences excerpted from actual reviews when context can be helpful.

Hygiene: Interestingly, hygiene related words are overwhelmingly negative, e.g., *"gross", "mess", "sticky"*. What this suggests is that reviewers do complain when the restaurants are noticeably dirty, but do not seem to feel the need to complement on cleanliness as often. Instead, they seem to focus on other positive aspects of their experience, e.g., details of food, atmosphere, and their social occasions.

Service and Atmosphere: Discriminative features reveal that it is not just the hygiene related words that are predictive of the inspection results of restaurants. It turns out that there are other qualities of restaurants, such as service and atmosphere, that also correlate with the likely outcome of inspections. For example, when reviewers feel the need to talk about *"door", "student", "sticker"*, or *"the size"* (see Table 2 and 3), one can extrapolate that the overall experience probably was not glorious. In contrast, words such as *"selection", "atmosphere", "ambiance"* are predictive of hygienic restaurants, even including those with slightly negative connotation such as *"attitude"* or *"pretentious"*.

Whom and When: If reviewers talk about details of their social occasions such as *"date", "husband"*, it seems to be a good sign.

The way food items are described: Another interesting aspect of discriminative words are the way food items are described by reviewers. In general, mentions of basic ingredients of dishes, e.g., *"noodle", "egg", "soy"* do not seem like a good sign. In contrast, words that help describing the way dish is prepared or decorated, e.g., *"grill", "toast", "frosting", "bento box" "sugar"* (as in "sugar coated") are good signs of satisfied customers.

Cuisines: Finally, cuisines have clear correlations with inspection outcome, as shown in Table 2 and 3.

5 Related Work

There have been several recent studies that probe the viability of public health surveillance by measuring relevant textual signals in social media, in particular, micro-blogs (e.g., Aramaki et al. (2011), Sadilek et al. (2012b), Sadilek et al. (2012a), Sadilek et al. (2013), Lamb et al. (2013), Dredze et al. (2013), von Etter et al. (2010)). Our work joins this line of research but differs in two distinct ways. First, most prior work aims to monitor a specific illness, e.g., influenza or food-poisoning by paying attention to a relatively small set of keywords that are directly relevant to the corresponding sickness. In contrast, we examine all words people use in online reviews, and draw insights on correlating terms and concepts that may not seem immediately relevant to the hygiene status of restaurants, but nonetheless are predictive of the outcome of the inspections. Second, our work is the first to examine online reviews in the context of improving public policy, suggesting additional source of information for public policy makers to pay attention to.

Our work draws from the rich body of research that studies online reviews for sentiment analysis (e.g., Pang and Lee (2008)) and deception detection (e.g., Mihalcea and Strapparava (2009), Ott et al. (2011), Feng et al. (2012)), while introducing the new task of public hygiene prediction. We expect that previous studies for aspect-based sentiment analysis (e.g., Titov and McDonald (2008), Brody and Elhadad (2010), Wang et al. (2010)) would be a fruitful venue for further investigation.

6 Conclusion

We have reported the first empirical study demonstrating the promise of review analysis for predicting health inspections, introducing a task that has potentially significant societal benefits, while being relevant to much research in NLP for opinion analysis based on customer reviews.

Acknowledgments

This research was supported in part by the Stony Brook University Office of the Vice President for Research, and in part by gift from Google. We thank anonymous reviewers and Adam Sadilek for helpful comments and suggestions.

References

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: Detecting influenza epidemics using twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1576, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 804–812, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark Dredze, Michael J. Paul, Shane Bergsma, and Hieu Tran. 2013. Carmen: A twitter geolocation system with applications to public health. In *AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI)*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012. Distributional footprints of deceptive product reviews. In *ICWSM*.

Katie Filion and Douglas A Powell. 2009. The use of restaurant inspection disclosure systems as a means of communicating food safety information. *Journal of Foodservice*, 20(6):287–297.

Spencer Henson, Shannon Majowicz, Oliver Masakure, Paul Sockett, Anria Johnes, Robert Hart, Debora Carr, and Lewinda Knowles. 2006. Consumer assessment of the safety of restaurants: The role of inspection notices and other information cues. *Journal of Food Safety*, 26(4):275–301.

Ginger Zhe Jin and Phillip Leslie. 2003. The effect of information on product quality: Evidence from restaurant hygiene grade cards. *The Quarterly Journal of Economics*, 118(2):409–451.

Ginger Zhe Jin and Phillip Leslie. 2005. The case in support of restaurant hygiene grade cards.

Ginger Zhe Jin and Phillip Leslie. 2009. Reputational incentives for restaurant hygiene. *American Economic Journal: Microeconomics*, pages 237–267.

Alex Lamb, Michael J. Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on twitter. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312, Suntec, Singapore, August. Association for Computational Linguistics.

NYC-DoHMH. 2012. Restaurant grading in new york city at 18 months. *New York City Department of Health and Mental Hygiene*.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA, June. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Adam Sadilek, Henry Kautz, and Vincent Silenzio. 2012a. Predicting disease transmission from geotagged micro-blog data. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Adam Sadilek, Henry A. Kautz, and Vincent Silenzio. 2012b. Modeling spread of disease from social interactions. In John G. Breslin, Nicole B. Ellison, James G. Shanahan, and Zeynep Tufekci, editors, *ICWSM*. The AAAI Press.

Adam Sadilek, Sean Brennan, Henry Kautz, and Vincent Silenzio. 2013. nemesis: Which restaurants should you avoid today? *First AAAI Conference on Human Computation and Crowdsourcing*.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, pages 308–316, Columbus, Ohio, June. Association for Computational Linguistics.

Peter von Etter, Silja Huttunen, Arto Vihavainen, Matti Vuorinen, and Roman Yangarber. 2010. Assessment of utility in web mining for the domain of public health. In *Proceedings of the NAACL HLT 2010 Second Louhi Workshop on Text and Data Mining of Health Documents*, pages 29–37, Los Angeles, California, USA, June. Association for Computational Linguistics.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACM.

Automatically Identifying Pseudepigraphic Texts

Moshe Koppel
Bar Ilan University
Ramat-Gan, 52900, Israel
moishk@gmail.com

Shachar Seidman
Bar Ilan University
Ramat-Gan, 52900, Israel
shachar9@gmail.com

Abstract

The identification of pseudepigraphic texts – texts not written by the authors to which they are attributed – has important historical, forensic and commercial applications. We introduce an unsupervised technique for identifying pseudepigrapha. The idea is to identify textual outliers in a corpus based on the pairwise similarities of all documents in the corpus. The crucial point is that document similarity not be measured in any of the standard ways but rather be based on the output of a recently introduced algorithm for authorship verification. The proposed method strongly outperforms existing techniques in systematic experiments on a blog corpus.

1 Introduction

The Shakespeare attribution problem is centuries old and shows no signs of abating. Some scholars argue that some, or even all, of Shakespeare's works were not actually written by him. The most mainstream theory – and the one that interests us here – is that most of the works were written by Shakespeare, but that several of them were not. Could modern methods of computational authorship attribution be used to detect which, if any, of the works attributed to Shakespeare were not written by him?

More generally, this paper deals with the unsupervised problem of detecting pseudepigrapha: documents in a supposedly single-author corpus that were not actually written by the corpus's presumed author. Studies as early as Mendenhall (1887), have observed that texts by a single author tend to be somewhat homogeneous in style. If this is indeed the case, we would expect that pseudepigrapha would be detectable as outliers.

Identifying such outlier texts is, of course, a special case of general outlier identification, one of the central tasks of statistics. We will thus consider the pseudepigrapha problem in the context of the more general outlier detection problem.

Typically, research on textual outliers assumes that we have a corpus of known authentic documents and are asked to decide if a specified other document is authentic or not (Juola and Stamatatos, 2013). One crucial aspect of our problem is that we do not assume that any specific text in a corpus is known a priori to be authentic or pseudepigraphic; we can assume only that most of the documents in the corpus are authentic.

The method we introduce in this paper builds on the approach of Koppel and Winter (2013) for determining if two documents are by the same author. We apply that method to every pair of documents in a corpus and use properties of the resulting adjacency graph to identify outliers. In the following section, we briefly outline previous work. In Section 3 we provide a framework for outlier detection and in Section 4 we describe our method. In Section 5 we describe the experimental setting and give results and in Section 6 we present results for the plays of Shakespeare.

2 Related Work

Identifying outlier texts consists of two main stages: first, representing each text as a numerical vector representing relevant linguistic features of the text and second, using generic methods to identify outlier vectors.

There is a vast literature on generic methods for outlier detection, summarized in Hodge & Austin (2004) and Chandola et al. (2009). Since our prob-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1449–1454,
Seattle, Washington, USA, 18-21 October 2013.

lem setup does not entail obtaining any labeled examples of authentic or outlier documents, supervised and semi-supervised methods are inapplicable. The available methods are unsupervised, principally probabilistic or proximity-based methods. A classical variant of such methods for univariate normally distributed data uses the the z-score (Grubbs, 1969). Such simple univariate outlier detectors are, however, inappropriate for identifying outliers in a high-dimensional textual corpus. Subsequent work, such as the *Stahel-Donoho Estimator* (Stahel, 1981; Donoho, 1982), *PCout* (Filzmoser et al., 2008), *LOF* (Breunig and Kriegel, 2000) and *ABOD* (Kriegel et al., 2008) have generalized univariate methods to high-dimensional data points.

In his comprehensive review of outlier detection methods in textual data, Guthrie (2008) compares a variety of vectorization methods along with a variety of generic outlier methods. The vectorization methods employ a variety of lexical and syntactic stylistic features, while the outlier detection methods use a variety of similarity/distance measures such as cosine and Euclidean distance. Similar methods have also been used in the field of intrinsic plagiarism detection, which involves segmenting a text and then identifying outlier segments (Stamatatos, 2009; Stein et al., 2010).

3 Proximity Methods

Formally, the problem we wish to solve is defined as follows: Given a set of documents $\mathbf{D} = \{\mathbf{d_1},\ldots,\mathbf{d_n}\}$, all or most of which were written by author A, which, if any, documents in D were not written by A?

We begin by considering the kinds of proximity methods for textual outlier detection considered by Guthrie (2008) and in the work on intrinsic plagiarism detection; these will serve as baseline methods for our approach. The idea is simple: mark as an outlier any document that is too far from the rest of the documents in the corpus.

We briefly sketch the key steps:

1. *Represent a document as a numerical vector.*
   The kinds of measurable features that can be used to represent a document include frequencies of word unigrams, function words, parts-of-speech and character n-grams, as well as complexity measures such as type/token ratio, sentence and word length and so on.
2. *Measure the similarity of two document vectors.*
   We can use either inverses of distance measures such as Euclidean distance or Manhattan distance, or else direct similarity measures such as cosine or min-max.
3. *Use an aggregation method to measure the similarity of a document to a set of documents.*
   One approach is to simply measure the distance from a document to the centroid of all the other documents (*centroid* method). Another approach is to first measure the similarity of a document to each other document and then to aggregate the results by averaging all the obtained values (*mean* method):
   $$mean_{\boldsymbol{d_t}\in\boldsymbol{D}}\big(sim(\boldsymbol{d_i},\boldsymbol{d_t})\big)$$
   Alternatively, we can average the values only for the k nearest neighbors (*k-NN* method):
   $$mean_{\boldsymbol{d_t}\in\boldsymbol{D_k}}\big(sim(\boldsymbol{d_i},\boldsymbol{d_t})\big)$$
   (where $\mathbf{D_k}$ = k nearest neighbors of $\mathbf{d_i}$).
   Yet another method is to use median distance (*median* method).
   $$median_{\boldsymbol{d_t}\in\boldsymbol{D}}\big(sim(\boldsymbol{d_i},\boldsymbol{d_t})\big)$$
   We note that the centroid method and mean method suffer from the *masking effect* (Bendre and Kale, 1987; Rousseeuw and Leroy, 2003): the presence of some outliers in the data can greatly distort the estimator's results regarding the presence of other outliers. The k-NN method and the median method are both much more robust.
4. *Choose some threshold beyond which a document is marked as an outlier.*
   Choosing the threshold is one of the central issues in statistical approaches. For our purposes, however, the choice of threshold is simply a parameter trading off recall and precision.

4 Second-Order Similarity

Our approach is to use an entirely different kind of similarity measure in Step 2. Rather than use a first-order similarity measure, as is customary, we employ a second-order similarity measure that is the output of an algorithm used for the *authorship verification problem* (Koppel et al. 2011), in which we need to determine if two, possibly short, documents were written by the same author.

That algorithm, known as the "impostors method" (*IM*), works as follows. Given two docu-

ments, $\mathbf{d_1}$ and $\mathbf{d_2}$, generate an appropriate set of impostor documents, $\mathbf{p_1},\ldots,\mathbf{p_m}$ and represent each of the documents in terms of some large feature set (for example, the frequencies of various words or character n-grams in the document). For some random subset of the feature set, measure the similarity of $\mathbf{d_1}$ to $\mathbf{d_2}$ as well as to each of the documents $\mathbf{p_1},\ldots,\mathbf{p_m}$ and note if $\mathbf{d_1}$ is closer to $\mathbf{d_2}$ than to any of the impostors. Repeat this k times, choosing a different random subset of the features in each iteration. If $\mathbf{d_1}$ is closer to $\mathbf{d_2}$ than to any of the impostors (and likewise switching the roles of $\mathbf{d_1}$ and $\mathbf{d_2}$) for at least θ% of iterations, then output that $\mathbf{d_2}$ and $\mathbf{d_1}$ are the same author. (The parameter θ is used to trade-off recall and precision.)

Adapting that method for our purposes, we use the proportion of iterations for which $\mathbf{d_1}$ is closer to $\mathbf{d_2}$ than to any of the impostors as our similarity measure (adding a small twist to make the measure symmetric over $\mathbf{d_1}$ and $\mathbf{d_2}$, as can be seen in line 2.2.2 of the algorithm). More precisely, we do the following:

Given: Corpus $\mathbf{D=\{d_1,\ldots,d_n\}}$

1. Choose a feature set **FS** for representing documents, a first-order similarity measure *sim*, and an impostor set $\{\mathbf{p_1},\ldots,\mathbf{p_m}\}$.
2. For each pair of documents $<\mathbf{d_i}, \mathbf{d_j}>$ in set **D**:
   - 2.1. Let $sim2(\mathbf{d_i}, \mathbf{d_j}) := 0$
   - 2.2. Iterate *K* times:
     - 2.2.1. Randomly choose 40% of features in **FS**
     - 2.2.2. If $sim(\mathbf{d_i}, \mathbf{d_j})^2 > max_{u\in\{1,..,m\}} sim(\mathbf{d_i}, \mathbf{p_u}) * max_{u\in\{1,..,m\}} sim(\mathbf{d_j}, \mathbf{p_u})$, then $sim2(\mathbf{d_i}, \mathbf{d_j}) := sim2(\mathbf{d_i}, \mathbf{d_j}) + 1/K$
3. For each document $\mathbf{d_i}$ in set **D**:
   - 3.1. Compute $sim2(\mathbf{d_i}, \mathbf{D}) = agg_{w\in\{1,..,n\}}[sim2(\mathbf{d_i}, \mathbf{d_w})]$ where *agg* is some aggregation function
   - 3.2. If $sim2(\mathbf{d_i}, \mathbf{D}) < \theta$ (where θ is a parameter), then mark $\mathbf{d_i}$ as *outlier*.

The method for choosing the impostor set is corpus-dependent, but quite straightforward: we simply choose random impostors from the same genre and language as the documents in question. The choice of feature set **FS,** first-order similarity measure *sim*, and aggregation function *agg* can be varied. For **FS**, we simply use bag-of-words (BOW). As for *sim* and *agg*, we show below results of experiments comparing the effectiveness of various choices for these parameters.

Using second-order similarity has several surface advantages over standard first-order measures. First, it is decisive: for most pairs, second-order similarity will be close to 0 or close to 1. Second, it is self-normalizing: scaling doesn't depend on the size of the underlying feature sets or the lengths of the documents. As we will see, it is also simply much more effective for identifying outliers.

5 Experiments

We begin by assembling a corpus consisting of 3540 blog posts written by 156 different bloggers. The blogs are taken from the blog corpus assembled by Schler et al. (2006) for use in authorship attribution tasks. Each of the blogs was written in English by a single author in 2004 and each post consists of 1000 words (excess is truncated).

For our initial experiments, each trial consists of 10 blog posts, all but *p* of which are by a single blogger. The number of pseudepigraphic documents, *p*, is chosen from a uniform distribution over the set {0,1,2,3}. Our task is to identify which, if any, documents in the set are not by the main author of the set. The pseudepigraphic documents might be written by a single author or by multiple authors.

To measure the performance of a given similarity measure *sim*, we do the following in each trial:

1. Represent each document in the trial set **D** in terms of BOW.
2. Measure the similarity of each pair of documents in the trial set using the similarity measure *sim*.
3. Using some aggregation function *agg*, compute for each document $\mathbf{d_i}$: $sim(\mathbf{d_i}, \mathbf{D}) = agg_{w\in\{1,..,n\}}[sim(\mathbf{d_i}, \mathbf{d_w})]$.
4. If $sim(\mathbf{d_i}, \mathbf{D}) < \theta$, mark $\mathbf{d_i}$ as an outlier (where θ is a parameter).

Our objective is to show that results using second-order similarity are stronger than those using first-order similarity. Before we do this, we need to determine the best aggregation function to use in our experiments. In Figure 1, we show recall-precision breakeven values (for the *outlier* class) over 250 independent trials, for each of our four first-order similarity measures (inverse Euclidean, inverse Manhattan, cosine, min-max) used in conjunction with each of four aggregation functions (centroid, mean, k-NN mean, median). As is evident, k-NN is the best aggregation function in each case. We will give these baseline methods an advantage by using k-NN as our aggregation function in all our subsequent experiments.

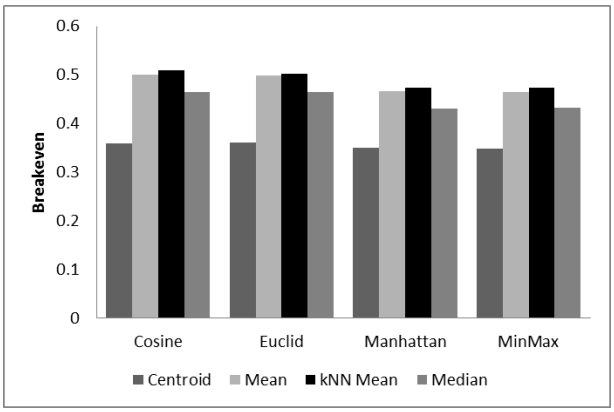


Figure 1. Breakeven values on first-order similarity measures with various aggregation functions.

We are now ready to perform our main experiment. We use BOW as our feature set and k-NN as our aggregation function. We use 500 random blog posts as our impostor set. In Figure 2, we show recall-precision curves for outlier documents over 250 independent trials, as just described, using four first-order similarity measures as well our second-order similarity measure using each of the four as a base measure. As can be seen, even the worst second-order similarity measure significantly outperforms all the standard first-order measures. In Figure 3, we show the breakeven values for each measure, pairing each first-order measure with the second-order measure that uses it as a base. Clearly, the mere use of a second-order method improves results, regardless of the base measure.

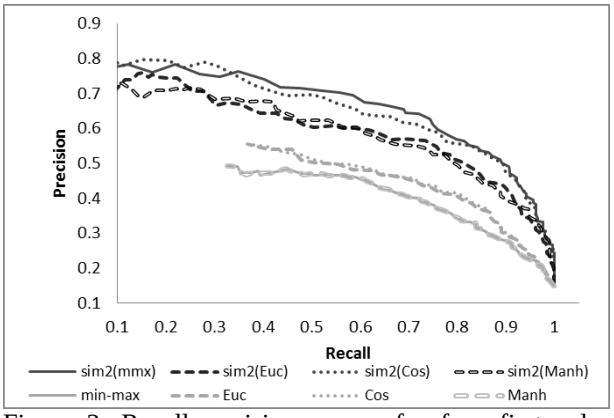


Figure 2. Recall-precision curves for four first-order similarity measures and four second-order similarity measures, based on 250 trials of 10 documents each.

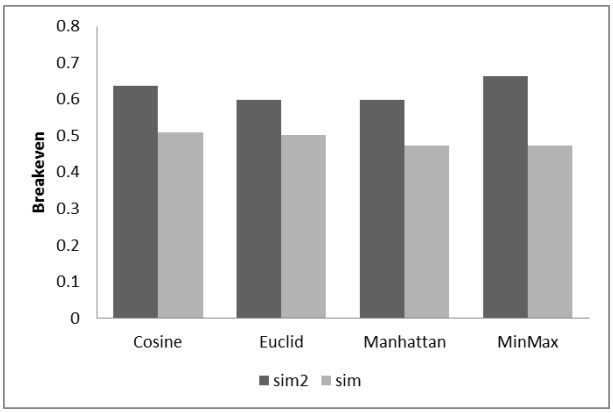


Figure 3. Breakeven values for first-order measures and corresponding second-order measures.

Thus far we have considered authorial corpora consisting of only ten documents. In Figures 4 and 5, we repeat the experiment described in Figures 2 and 3 above, but with each trial consisting of 50 documents including any number of pseudepigraphic documents in the range 0 to 15. The same phenomenon is apparent: second-order similarity strongly improves results over the corresponding first-order base similarity measure.

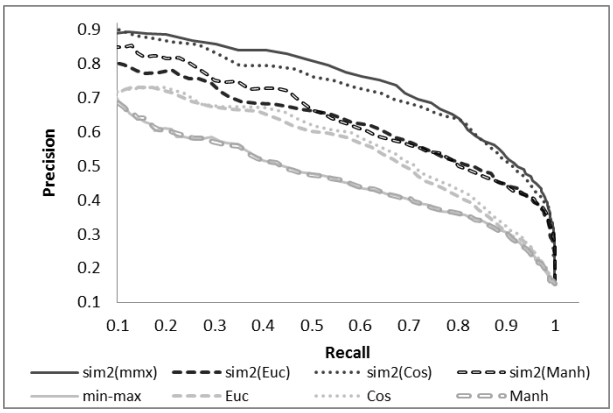


Figure 4. Recall-precision curves for four first-order similarity measures and four second-order similarity measures, based on 250 trials of 50 documents each.

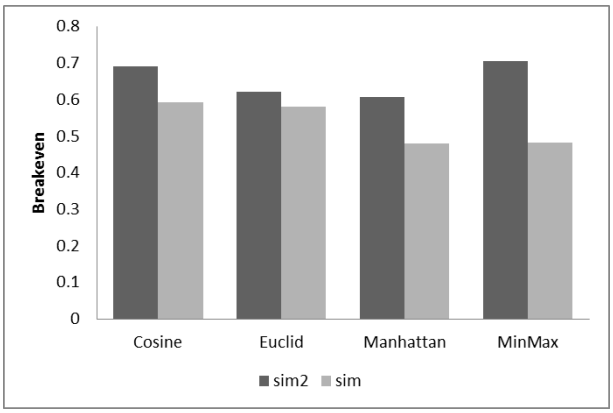


Figure 5. Breakeven values for first-order measures and corresponding second-order measures

6 Results on Shakespeare

We applied our methods to the texts of 42 plays by Shakespeare (taken from Project Gutenberg). We included two plays by Thomas Kyd as sanity checks. In addition, we included three plays occasionally attributed to Shakespeare, but generally regarded by authorities as pseudepigrapha (*A Yorkshire Tragedy*, *The Life of Sir John Oldcastle* and *Pericles Prince of Tyre*). We also included *King Edward III* and *King Henry VI* (Part 1), both of which are subjects of dispute among Shakespeare scholars. As impostors we used 39 works by contemporaries of Shakespeare, including Christopher Marlowe, Ben Jonson and John Fletcher.

We found that the two plays by Thomas Kyd and the three pseudepigraphic plays were all among the seven furthest outliers, as one would expect. In addition, *King Edward III* was 9th furthest. *King Henry VI* (Part 1) was not found to be an outlier at all. Curiously, however, three undisputed plays by Shakespeare were found to be greater outliers than *King Edward III*. These are *The Merry Wives of Windsor*, *The Comedy of Errors* and *The Tragedy of Julius Caesar*. *The Merry Wives of Windsor* is a particularly distant outlier, even further out than *Oldcastle* and *Pericles*. We leave it to Shakespeare scholars to explain the reasons for these anomalies.

7 Conclusion

In this paper we defined the problem of unsupervised outlier detection in the authorship verification domain. Our method combines standard outlier detection methods with a novel inter-document similarity measure. This similarity measure is the output of the impostors method recently developed for solving the authorship verification problem. We have found that use of the kNN method for outlier detection in conjunction with this second-order similarity measure strongly outperforms methods based on any outlier detection method used in conjunction with any standard first-order similarity measures. This improvement proves to be robust, holding for various corpus sizes and various underlying base similarity measures used in the second-order similarity measure.

The method can be used to resolve historical conundrums regarding the authenticity of works in questioned corpora, such as the Shakespeare corpus briefly considered here. This is currently the subject of our ongoing research.

References

S. M. Bendre and B. K. Kale. 1987. *Masking effect on tests for outliers in normal samples*, Biometrika, 74(4):891-896.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jörg Sander. 2000. *LOF: Identifying Density-Based Local Outliers*, ACM SIGMOD Conference Proceedings.

Varun Chandola, Arindam Banerjee and Vipin Kumar. 2009. *Anomaly detection: a survey*. ACM Computing Surveys 41, 3, Article 15.

David L. Donoho. 1982. *Breakdown properties of multivariate location estimators*. Ph.D. qualifying paper, Harvard University.

Peter Filzmoser, Ricardo Maronna and Mark Werner. 2008. *Outlier identification in high dimensions*. Computational Statistics and Data Analysis, 52:1694-1711.

David Guthrie. 2008. *Unsupervised Detection of Anomalous Text*. PhD Thesis, University of Sheffield.

Frank E. Grubbs. 1969. *Procedures for detecting outlying observations in samples*, Technometrics.

V.J. Hodge and J. Austin. 2004. A survey of outlier detection methodologies. Artificial. Intelligence Review, 22 (2). pp. 85-126.

Patrick Juola and Efstathios Stamatatos. 2013. *Overview of the Author Identification Task at PAN 2013*. P. Forner, R. Navigli, and D. Tufis (eds) CLEF 2013 Evaluation Labs and Workshop –Working Notes Papers.

Moshe Koppel and Jonathan Schler 2004. *Authorship verification as a one-class classification problem*. In ICML '04: Twenty-first International Conference on Machine Learning, New York, NY, USA.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. *Authorship attribution in the wild*. Language Resources and Evaluation, 45(1): 83–94.

Moshe Koppel M. and Yaron Winter. 2013. *Determining If Two Documents Are by the Same Author*. J. Am. Soc. Inf. Sci. Technol.

Frederick Mosteller and David L. Wallace. 1964. *Inference and Disputed Authorship: The Federalist*. Reading, Mass. Addison Wesley.

Hans-Peter Kriegel, Matthias S. Schubert and Arthur Zimek. 2008. *Angle-based outlier detection in high dimensional data*. Proc. KDD.

Thomas C. Mendenhall. 1887. *The characteristic curves of composition*, Science 9, 237-259.

Sridhar Ramaswamy, Rajeev Rastogi and Kyuseok Shim. 2000. *Efficient Algorithms for Mining Outliers from Large Data Sets*. Proc. ACM SIDMOD Int. Conf. on Management of Data.

Peter J. Rousseeuw. 1984. *Least median of squares regression*. Journal of the American Statistical Association, 79(388):87-880.

Peter J. Rousseeuw and Annick M. Leroy. 2003. *Robust Regression and Outlier Detection*. John Wiley & Sons.

J. Schler, M. Koppel, S. Argamon and J. Pennebaker. 2006. *Effects of Age and Gender on Blogging*. in Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs.

Werner A. Stahel. 1981. *Breakdown of covariance estimators*. Research Report 31, Fachgruppe f¨ur Statistik, Swiss Federal Institute of Technology (ETH), Zurich.

Efstathios Stamatatos. 2009. *Intrinsic plagiarism detection using character n-gram profiles*. Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 38–46.

Benno Stein B, Nedim Lipka and Peter Prettenhofer. 2010. *Intrinsic Plagiarism Analysis*. Language Resources and Evaluation, 1–20. 2010.

Benno Stein B, Nedim Lipka and Peter Prettenhofer. 2010. *Intrinsic Plagiarism Analysis*. Language Resources and Evaluation, 1–20. 2010.

Dynamic Feature Selection for Dependency Parsing

He He **Hal Daumé III**
Department of Computer Science
University of Maryland
College Park, MD 20740
{hhe,hal}@cs.umd.edu

Jason Eisner
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
jason@cs.jhu.edu

Abstract

Feature computation and exhaustive search have significantly restricted the speed of graph-based dependency parsing. We propose a faster framework of *dynamic feature selection*, where features are added sequentially as needed, edges are pruned early, and decisions are made online for each sentence. We model this as a sequential decision-making problem and solve it by imitation learning techniques. We test our method on 7 languages. Our dynamic parser can achieve accuracies comparable or even superior to parsers using a full set of features, while computing fewer than 30% of the feature templates.

1 Introduction

Graph-based dependency parsing usually consists of two stages. In the scoring stage, we score all possible edges (or other small substructures) using a learned function; in the decoding stage, we use combinatorial optimization to find the dependency tree with the highest *total* score.

Generally linear edge-scoring functions are used for speed. But they use a large set of features, derived from feature templates that consider different conjunctions of the edge's attributes. As a result, parsing time is dominated by the scoring stage—computing edge attributes, using them to instantiate feature templates, and looking up the weights of the resulting features in a hash table. For example, McDonald et al. (2005a) used on average about 120 first-order feature templates on each edge, built from attributes such as the edge direction and length, the two words connected by the edge, and the parts of speech of these and nearby words.

We therefore ask the question: can we use fewer features to score the edges, while maintaining the effect that the true dependency tree still gets a higher score? Motivated by recent progress on dynamic feature selection (Benbouzid et al., 2012; He et al., 2012), we propose to add features one group at a time to the dependency graph, and to use these features together with interactions among edges (as determined by intermediate parsing results) to make hard decisions on some edges before all their features have been seen. Our approach has a similar flavor to cascaded classifiers (Viola and Jones, 2004; Weiss and Taskar, 2010) in that we make decisions for each edge at every stage. However, in place of relatively simple heuristics such as a global relative pruning threshold, we learn a featurized decision-making policy of a more complex form. Since each decision can affect later stages, or later decisions in the same stage, we model this problem as a sequential decision-making process and solve it by Dataset Aggregation (DAgger) (Ross et al., 2011), a recent iterative imitation learning technique for structured prediction.

Previous work has made much progress on the complementary problem: speeding up the decoding stage by pruning the search space of tree structures. In Roark and Hollingshead (2008) and Bergsma and Cherry (2010), pruning decisions are made locally as a preprocessing step. In the recent vine pruning approach (Rush and Petrov, 2012), significant speedup is gained by leveraging structured information via a coarse-to-fine projective parsing cas-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1455–1464,
Seattle, Washington, USA, 18-21 October 2013.

cade (Charniak et al., 2006). These approaches do not directly tackle the feature selection problem. Although pruned edges do not require further feature computation, the pruning step must itself compute similar high-dimensional features just to decide which edges to prune. For this reason, Rush and Petrov (2012) restrict the pruning models to a smaller feature set for time efficiency. We aim to do feature selection and edge pruning dynamically, balancing speed and accuracy by using only as many features as needed.

In this paper, we first explore standard static feature selection methods for dependency parsing, and show that even a few feature templates can give decent accuracy (Section 3.2). We then propose a novel way to dynamically select features for each edge while keeping the overhead of decision making low (Section 4). Our present experiments use the Maximum Spanning Tree (MST) parsing algorithm (McDonald et al., 2005a; McDonald and Pereira, 2006). However, our approach applies to other graph-based dependency parsers as well—including non-projective parsing, higher-order parsing, or approximations to higher-order parsing that use stacking (Martins et al., 2008), belief propagation (Smith and Eisner, 2008), or structured boosting (Wang et al., 2007).

2 Graph-based Dependency Parsing

In graph-based dependency parsing of an n-word input sentence, we must construct a tree $\boldsymbol{y}$ whose vertices $0, 1, \ldots n$ correspond to the root node (namely 0) and the ordered words of the sentence. Each directed edge of this tree points from a head (parent) to one of its modifiers (child).

Following a common approach to structured prediction problems, the score of a tree $\boldsymbol{y}$ is defined as a sum of local scores. That is, $s_{\boldsymbol{\theta}}(\boldsymbol{y}) = \boldsymbol{\theta} \cdot \sum_{E \in \boldsymbol{y}} \phi(E) = \sum_{E \in \boldsymbol{y}} \boldsymbol{\theta} \cdot \phi(E)$, where E ranges over small connected subgraphs of $\boldsymbol{y}$ that can be scored individually. Here $\phi(E)$ extracts a high-dimensional feature vector from E together with the input sentence, and $\boldsymbol{\theta}$ denotes a weight vector that has typically been learned from data.

The first-order model decomposes the tree into edges E of the form $\langle h, m \rangle$, where $h \in [0, n]$ and $m \in [1, n]$ (with $h \neq m$) are a head token and one of its modifiers. Finding the best tree requires first computing $\boldsymbol{\theta} \cdot \phi(E)$ for each of the n^2 possible edges.

Since scoring the edges independently in this way restricts the parser to a local view of the dependency structure, higher-order models can achieve better accuracy. For example, in the second-order model of McDonald and Pereira (2006), each local subgraph E is a triple that includes the head and two modifiers of the head, which are adjacent to each other. Other methods that use triples include grandparent-parent-child triples (Koo and Collins, 2010), or non-adjacent siblings (Carreras, 2007). Third-order models (Koo and Collins, 2010) use quadruples, employing grand-sibling and tri-sibling information.

The usual inference problem is to find the highest scoring tree for the input sentence. Note that in a valid tree, each token $1, \ldots, n$ must be attached to exactly one parent (either another token or the root 0). We can further require the tree to be projective, meaning that edges are not allowed to cross each other. It is well known that dynamic programming can be used to find the best projective dependency tree in $O(n^3)$ time, much as in CKY, for first-order models and some higher-order models (Eisner, 1996; McDonald and Pereira, 2006).[1] When the projectivity restriction is lifted, McDonald et al. (2005b) pointed out that the best tree can be found in $O(n^2)$ time using a minimum directed spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967; Tarjan, 1977), though only for first-order models.[2] We will make use of this fast non-projective algorithm as a subroutine in early stages of our system.

3 Dynamic Feature Selection

Unlike typical feature selection methods that fix a subset of selected features and use it throughout testing, in dynamic feature selection we choose features adaptively for each instance. We briefly introduce this framework below and motivate our algorithm from empirical results on MST dependency parsing.

[1] Although the third-order model of Koo and Collins (2010), for example, takes $O(n^4)$ time.

[2] The non-projective parsing problem becomes NP-hard for higher-order models. One approximate solution (McDonald and Pereira, 2006) works by doing projective parsing and then rearranging edges.

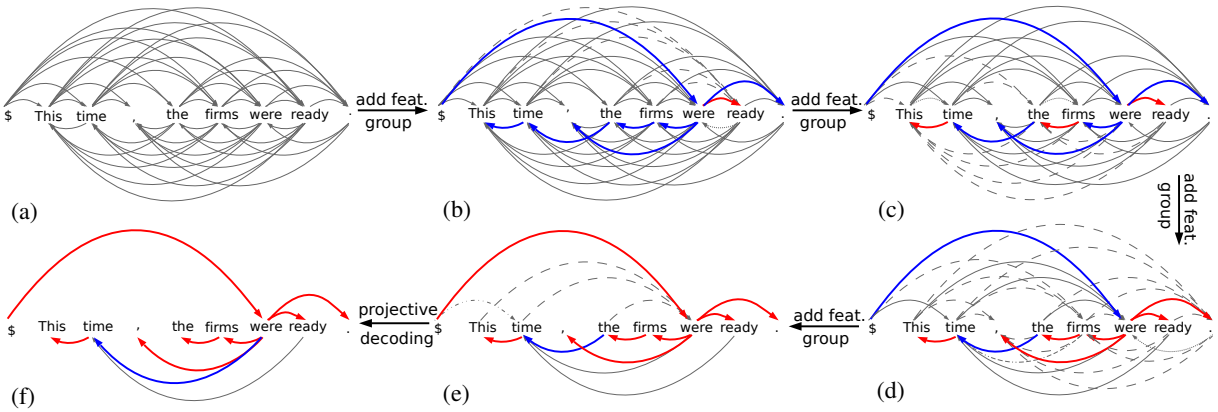


Figure 1: Dynamic feature selection for dependency parsing. (a) Start with all possible edges except those filtered by the length dictionary. (b) – (e) Add the next group of feature templates and parse using the non-projective parser. Predicted trees are shown as blue and red edges, where red indicates the edges that we then decide to lock. Dashed edges are pruned because of having the same child as a locked edge; 2-dot-3-dash edges are pruned because of crossing with a locked edge; fine-dashed edges are pruned because of forming a cycle with a locked edge; and 2-dot-1-dash edges are pruned since the root has already been locked with one child. (f) Final projective parsing.

3.1 Sequential Decision Making

Our work is motivated by recent progress on dynamic feature selection (Benbouzid et al., 2012; He et al., 2012; Grubb and Bagnell, 2012), where features are added sequentially to a test instance based on previously acquired features and intermediate prediction results. This requires sequential decision making. Abstractly, when the system is in some state $s \in S$, it chooses an action $a = \pi(s)$ from the action set A using its policy π, and transitions to a new state s', inducing some cost. In the specific case of dynamic feature selection, when the system is in a given state, it decides whether to add some more features or to stop and make a prediction based on the features added so far. Usually the sequential decision making problem is solved by reinforcement learning (Sutton and Barto, 1998) or imitation learning (Abbeel and Ng, 2004; Ratliff et al., 2004).

The dynamic feature selection framework has been successfully applied to supervised classification and ranking problems (Benbouzid et al., 2012; He et al., 2012; Gao and Koller, 2010). Below, we design a version that avoids overhead in our *structured prediction* setting. As there are n^2 possible edges on a sentence of length n, we wish to avoid the overhead of making many individual decisions about specific features on specific edges, with each decision considering the current scores of all other edges. Instead we will batch the work of dynamic feature selection into a smaller number of coarse-grained steps.

3.2 Strategy

To speed up graph-based dependency parsing, we first investigate time usage in the parsing process on our development set, section 22 of the Penn Treebank (PTB) (Marcus et al., 1993). In Figure 2, we observe that (a) feature computation took more than 80% of the total time; (b) even though non-projective decoding time grows quadratically in terms of the sentence length, in practice it is almost negligible compared to the projective decoding time, with an average of 0.23 ms; (c) the second-order projective model is significantly slower due to higher asymptotic complexity in both the scoring and decoding stages.

At each stage of our algorithm, we need to decide whether to use additional features to refine the edge scores. As making this decision separately for each of the n^2 possible edges is expensive, we instead propose a version that reduces the number of decisions needed. We show the process for one short sentence in Figure 1. The first step is to parse using the current features. We use the fast first-order non-projective parser for this purpose, since given observations (b) and (c), we cannot afford to run projective parsing multiple times. The single resulting tree (blue and red edges in Figure 1) has only

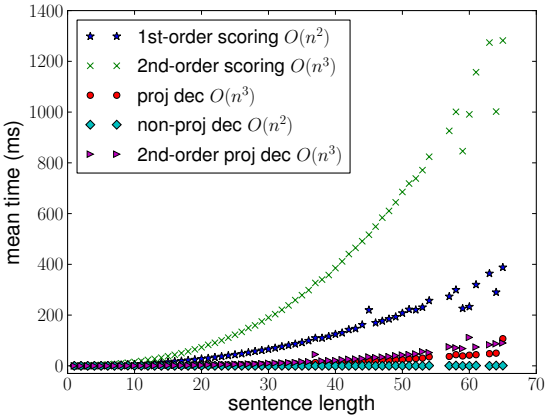


Figure 2: Time comparison of scoring time and decoding time on English PTB section 22.

n edges, and we use a classifier to decide which of these edges are reliable enough that we should "lock" them—i.e., commit to including them in the final tree. This is the only decision that our policy π must make. Locked (red) edges are definitely in the final tree. We also do constraint propagation: we rule out all edges that conflict with the locked edges, barring them from appearing in the final tree.[3] Conflicts are defined as violation of the projective parsing constraints:

- Each word has exactly one parent
- Edges cannot cross each other[4]
- The directed graph is non-cyclic
- Only one word is attached to the root

For example, in Figure 1(d), the dashed edges are removed because they have the same child as one of the locked (red) edges. The 2-dot-3-dash edge *time* ← *firms* is removed because it crosses the locked edge *(comma)* ← *were* (whereas we ultimately seek a projective parse). The fine dashed edge *were* ← *(period)* is removed because it forms a cycle with *were* → *(period)*. In Figure 1(e), the 2-dot-1-dash edge *(root)* → *time* is removed since we allow the root to have only one modifier.

[3]Constraint propagation also automatically locks an edge when all other edges with the same child have been ruled out.

[4]A reviewer asks about the cost of finding edges that cross a locked edge. Naively this is $O(n^2)$. But at most n edges will be locked during the entire algorithm, for a total $O(n^3)$ runtime—the same as *one* call to projective parsing, and far faster in practice. With cleverness this can even be reduced to $O(n^2 \log n)$.

Once constraint propagation has finished, we visit all edges (gray) whose fate is still unknown, and update their scores in parallel by adding the next group of features.

As a result, most edges will be locked in or ruled out without needing to look up all of their features. Some edges may still remain uncertain even after including all features. If so, a final iteration (Figure 1 (f)) uses the slower projective parser to resolve the status of these maximally uncertain edges. In our example, the parser does not figure out the correct parent of *time* until this final step. This final, accurate parser can use its own set of weighted features, including higher-order features, as well as the projectivity constraint. But since it only needs to resolve the few uncertain edges, both scoring and decoding are fast.

If we wanted our parser to be able to produce non-projective trees, then we would skip this final step or have it use a higher-order non-projective parser. Also, at earlier steps we would not prune edges crossing the locked edges.

4 Methods

Our goal is to produce a faster dependency parser by reducing the feature computation time. We assume that we are given three increasingly accurate but increasingly slow parsers that can be called as subroutines: a first-order non-projective parser, a first-order projective parser, and a second-order projective parser. In all cases, their feature weights have already been trained using the *full* set of features, and we will not change these weights. In general we will return the output of one of the projective parsers. But at early iterations, the non-projective parser helps us rapidly consider interactions among edges that may be relevant to our dynamic decisions.

4.1 Feature Template Ranking

We first rank the 268 first-order feature templates by forward selection. We start with an empty list of feature templates, and at each step we greedily add the one whose addition most improves the parsing accuracy on a development set. Since some features may be slower than others (for example, the "between" feature templates require checking all tokens in-between the head and the modifier), we could in-

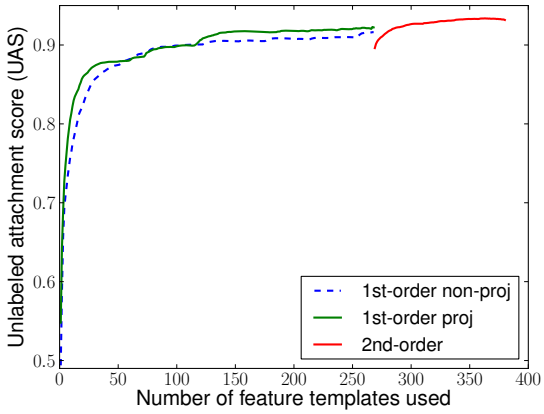


Figure 3: Forward feature selection result using the non-projective model on English PTB section 22.

stead select the feature template with the highest ratio of accuracy improvement to runtime. However, for simplicity we do not consider this: after grouping (see below), minor changes of the ranks within a group have no effect. The accuracy is evaluated by running the first-order non-projective parser, since we will use it to make most of the decisions. The 112 second-order feature templates are then ranked by adding them in a similar greedy fashion (given that all first-order features have already been added), evaluating with the second-order projective parser.

We then divide this ordered list of feature templates into K groups: $\{T_1, T_2, \ldots, T_K\}$. Our parser adds an entire group of feature templates at each step, since adding one template at a time would require too many decisions and obviate speedups. The simplest grouping method would be to put an equal number of feature templates in each group. From Figure 3 we can see that the accuracy increases significantly with the first few templates and gradually levels off as we add less valuable templates. Thus, a more cost-efficient method is to split the ranked list into several groups so that the accuracy increases by roughly the same amount after each group is added. In this case, earlier stages are fast because they tend to have many fewer feature templates than later stages. For example, for English, we use 7 groups of first-order feature templates and 4 groups of second-order feature templates. The sequence of group sizes is 1, 4, 10, 12, 47, 33, 161 and 35, 29, 31, 17 for first- and second-order parsing respectively.

4.2 Sequential Feature Selection

Similar to the length dictionary filter of Rush and Petrov (2012), for each test sentence, we first deterministically remove edges longer than the maximum length of edges in the training set that have the same head POS tag, modifier POS tag, and direction. This simple step prunes around 40% of the non-gold edges in our Penn Treebank development set (Section 6.1) at a cost of less than 0.1% in accuracy.

Given a test sentence of length n, we start with a complete directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{E} = \{\langle h, m\rangle : h \in [0, n], m \in [1, n]\}$. After the length dictionary pruning step, we compute T_1 for all remaining edges to obtain a pruned weighted directed graph. We predict a parse tree using the features so far (other features are treated as absent, with value 0). Then for each edge in this intermediate tree, we use a binary linear classifier to choose between two actions: $A = \{lock, add\}$. The *lock* action ensures that $\langle h, m\rangle$ appears in the final parse tree by pruning edges that conflict with $\langle h, m\rangle$.[5] If the classifier is not confident enough about the parent of m, it decides to *add* to gather more information. The *add* action computes the next group of features for $\langle h, m\rangle$ and all other competing edges with child m.

(Since we classify the edges one at a time, decisions on one edge may affect later edges. To improve efficiency and reduce cascaded error, we sort the edges in the predicted tree and process them as above in descending order of their scores.)

Now we can continue with the second iteration of parsing. Overall, our method runs up to $K = K_1 + K_2$ iterations on a given sentence, where we have K_1 groups of first-order features and K_2 groups of second-order features. We run $K_1 - 1$ iterations of non-projective first-order parsing (adding groups $T_1, \ldots, T_{K_1-1}$), then 1 iteration of *projective* first-order parsing (adding group T_{K_1}), and finally K_2 iterations of projective second-order parsing (adding groups $T_{K_1+1}, \ldots T_K$).

Before each iteration, we use the result of the previous iteration (as explained above) to prune some edges and add a new group of features to the rest. We

[5] If the conflicting edge is in the current predicted parse tree (which can happen because of non-projectivity), we forbid the model to prune it. Otherwise in rare cases the non-projective parser at the next stage may fail to find a tree.

then run the relevant parser. Each of the three parsers has a different set of feature weights, so when we switch parsers on rounds K_1 and $K_1 + 1$, we must also *change* the weights of the previously added features to those specified by the new parsing model.

In practice, we can stop as soon as the fate of all edges is known. Also, if no projective parse tree can be constructed at round K_1 using the available unpruned edges, then we immediately fall back to returning the non-projective parse tree from round $K_1 - 1$. This FAIL case rarely occurs in our experiments (fewer than 1% of sentences).

We report results both for a first-order system where $K_2 = 0$ (shown in Figure 1 and Algorithm 1) and for a second-order system where $K_2 > 0$.

Algorithm 1 DynFS($\mathcal{G}(\mathcal{V}, \mathcal{E}), \pi$)

$\mathcal{E} \leftarrow \{\langle h, m\rangle : |h - m| \leq \text{lenDict}(h, m)\}$
Add T_1 to all edges in $\mathcal{E}$
$\hat{\boldsymbol{y}} \leftarrow$ non-projective decoding
for $i = 2$ to K **do**
  $\mathcal{E}_{\text{sort}} \leftarrow$ sort unlocked edges $\{E : E \in \hat{\boldsymbol{y}}\}$ in descending order of their scores
  for $\langle h, m\rangle \in \mathcal{E}_{\text{sort}}$ **do**
    if $\pi(\psi(\langle h, m\rangle)) == \textit{lock}$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \setminus \{\{\langle h', m\rangle \in \mathcal{E} : h' \neq h\} \bigcup$
      $\{\langle h', m'\rangle \in \mathcal{E} :$ crosses $\langle h, m\rangle\} \bigcup$
      $\{\langle h', m'\rangle \in \mathcal{E} :$ cycle with $\langle h, m\rangle\}\}$
      if $h == 0$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \setminus \{\langle 0, m'\rangle \in \mathcal{E} : m' \neq m\}$
      end if
    else
      Add T_i to $\{\langle h', m'\rangle \in \mathcal{E} : m' == m\}$
    end if
  end for
  if $i == K$ **then**
    $\hat{\boldsymbol{y}} \leftarrow$ projective decoding
  else if $i \neq K$ **or** FAIL **then**
    $\hat{\boldsymbol{y}} \leftarrow$ non-projective decoding
  end if
end for
return $\hat{\boldsymbol{y}}$

5 Policy Training

We cast this problem as an imitation learning task and use Dataset Aggregation (DAgger) Ross et al. (2011) to train the policy iteratively.

5.1 Imitation Learning

In imitation learning (also called apprenticeship learning) (Abbeel and Ng, 2004; Ratliff et al., 2004), instead of exploring the environment directed by its feedback (reward) as in typical reinforcement learning problems, the learner observes expert demonstrations and aims to mimic the expert's behavior. The expert demonstration can be represented as trajectories of state-action pairs, $\{(s_t, a_t)\}$ where t is the time step. A typical approach to imitation learning is to collect supervised data from the expert's trajectories to learn a policy (multiclass classifier), where the input is $\psi(s)$, a feature representation of the current state (we call these *policy features* to avoid confusion with the *parsing features*), and the output is the predicted action (label) for that state.

In the sequential feature selection framework, it is hard to directly apply standard reinforcement learning algorithms, as we cannot assign credit to certain features until the policy decides to stop and let us evaluate the prediction result. On the other hand, knowing the gold parse tree makes it easy to obtain expert demonstrations, which enables imitation learning.

5.2 DAgger

Since the above approach collects training data only from the expert's trajectories, it ignores the fact that the distribution of states at training time and that at test time are different. If the learned policy cannot mimic the expert perfectly, one wrong step may lead to states never visited by the expert due to cumulative errors. This problem of insufficient exploration can be alleviated by iteratively learning a policy trained under states visited by both the expert and the learner (Ross et al., 2011; Daumé III et al., 2009; Kääriäinen, 2006).

Ross et al. (2011) proposed to train the policy iteratively and aggregate data collected from the previous learned policy. Let π^* denote the expert's policy and s_{π_i} denote states visited by executing π_i. In its simplest parameter-free form, in each iteration, we first run the most recently learned policy π_i; then for each state s_{π_i} on the trajectory, we collect a training example $(\psi(s_{\pi_i}), \pi^*(s_{\pi_i}))$ by labeling the state with the expert's action. Intuitively, this step intends to correct the learner's mistakes and pull it back to the

expert's trajectory. Thus we can obtain a policy that performs well under its own induced state distribution.

5.3 DAgger for Feature Selection

In our case, the expert's decision is rather straightforward. Replace the policy π in Algorithm 1 by an expert. If the edge under consideration is a gold edge, it executes *lock*; otherwise, it executes *add*. Basically the expert "cheats" by knowing the true tree and always making the right decision. On our PTB dev set, it can get 96.47% accuracy[6] with only 2.9% of the first-order features. This is an upper bound on our performance.

We present the training procedure in Algorithm 2. We begin by partitioning the training set into N folds. To simulate parsing results at test time, when collecting examples on $\mathcal{T}^i$, similar to cross-validation, we use parsers trained on $\bar{\mathcal{T}}^i = \mathcal{T} \setminus \mathcal{T}^i$. Also note that we show only one pass over training sentences in Algorithm 2; however, multiple passes are possible in practice, especially when the training data is limited.

Algorithm 2 DAgger($\mathcal{T}$, π^*)

```
Split the training sentences T into N folds T^1, T^2, ..., T^N
Initialize D ← ∅, π_1 ← π*
for i = 1 to N do
  for G(V, E) ∈ T^i do
    Sample trajectories {(s_{π_i}, π_i(s_{π_i}))} by DynFS(G(V, E), π_i)
    D ← D ⋃ {(ψ(s), π*(s))}
  end for
end for
Train policy π_{i+1} on D
return Best π_i evaluated on development set
```

5.4 Policy Features

Our linear edge classifier uses a feature vector ψ that concatenates all previously acquired parsing features together with "meta-features" that reflect confidence in the edge. The classifier's weights are fixed across iterations, but ψ(edge) changes per iteration.

We standardize the edge scores by a sigmoid function. Let $\dot{s}$ denote the normalized score, defined by $\dot{s}_{\boldsymbol{\theta}}(\langle h, m\rangle) = 1/(1 + \exp\{-s_{\boldsymbol{\theta}}(\langle h, m\rangle)\})$. Our meta-features for $\langle h, m\rangle$ include

- current normalized score, and normalized score before adding the current feature group
- margin to the highest scoring competing edges, i.e., $\dot{s}(\boldsymbol{w}, \langle h, m\rangle) - \max_{h'} \dot{s}(\boldsymbol{w}, \langle h', m\rangle)$ where $h' \in [0, n]$ and $h' \neq h$
- index of the next feature group to be added

We also tried more complex meta-features, for example, mean and variance of the scores of competing edges, and structured features such as whether the head of e is locked and how many locked children it currently has. It turns out that *given all the parsing features*, the margin is the most discriminative meta-feature. When it is present, other meta-features we added do not help much, Thus we do not include them in our experiments due to overhead.

6 Experiment

6.1 Setup

We generate dependency structures from the PTB constituency trees using the head rules of Yamada and Matsumoto (2003). Following convention, we use sections 02–21 for training, section 22 for development and section 23 for testing. We also report results on six languages from the CoNLL-X shared task (Buchholz and Marsi, 2006) as suggested in (Rush and Petrov, 2012), which cover a variety of language families. We follow the standard training/test split specified in the CoNLL-X data and tune parameters by cross validation when training the classifiers (policies). The PTB test data is tagged by a Stanford part-of-speech (POS) tagger (Toutanova et al., 2003) trained on sections 02–21. We use the provided gold POS tags for the CoNLL test data. All results are evaluated by the unlabeled attachment score (UAS). For fair comparison with previous work, punctuation is included when computing parsing accuracy of all CoNLL-X languages but not English (PTB).

For policy training, we train a linear SVM classifier using Liblinear (Fan et al., 2008). For all languages, we run DAgger for 20 iterations and se-

[6]The imperfect performance is because the accuracy is measured with respect to the gold parse trees. The expert only makes optimal pruning decisions but the performance depends on the pre-trained parser as well.

Language	Method	First-order				Second-order			
		Speedup	Cost(%)	UAS(D)	UAS(F)	Speedup	Cost(%)	UAS(D)	UAS(F)
Bulgarian	DynFS	3.44	34.6	91.1	91.3	4.73	16.3	91.6	92.0
	VineP	3.25	-	90.5	90.7	7.91	-	91.6	92.0
Chinese	DynFS	2.12	42.7	91.0	91.3	2.36	31.6	91.6	91.9
	VineP	1.02	-	89.3	89.5	2.03	-	90.3	90.5
English	DynFS	5.58	24.8	91.7	91.9	5.27	49.1	92.5	92.7
	VineP	5.23	-	91.0	91.2	11.88	-	92.2	92.4
German	DynFS	4.71	21.0	89.2	89.3	6.02	36.6	89.7	89.9
	VineP	3.37	-	89.0	89.2	7.38	-	90.1	90.3
Japanese	DynFS	4.80	15.6	93.7	93.6	8.49	7.53	93.9	93.9
	VineP	4.60	-	91.7	92.0	14.90	-	92.1	92.0
Portuguese	DynFS	4.36	32.9	87.3	87.1	6.84	40.4	88.0	88.2
	VineP	4.47	-	90.0	90.1	12.32	-	90.9	91.2
Swedish	DynFS	3.60	37.8	88.8	89.0	5.04	22.1	89.5	89.8
	VineP	4.64	-	88.3	88.5	13.89	-	89.4	89.7

Table 1: Comparison of speedup and accuracy with the vine pruning cascade approach for six languages. In the setup, DynFS means our dynamic feature selection model, VineP means the vine pruning cascade model, UAS(D) and UAS(F) refer to the unlabeled attachment score of the dynamic model (D) and the full-feature model (F) respectively. For each language, the speedup is relative to its corresponding first- or second-order model using the full set of features. Results for the vine pruning cascade model are taken from Rush and Petrov (2012). The cost is the percentage of feature templates used per sentence on edges that are *not pruned by the dictionary filter*.

lect the best policy evaluated on the development set among the 20 policies obtained from each iteration.

6.2 Baseline Models

We use the publicly available implementation of MSTParser[7] (with modifications to the feature computation) and its default settings, so the feature weights of the projective and non-projective parsers are trained by the MIRA algorithm (Crammer and Singer, 2003; Crammer et al., 2006).

Our feature set contains most features proposed in the literature (McDonald et al., 2005a; Koo and Collins, 2010). The basic feature components include lexical features (token, prefix, suffix), POS features (coarse and fine), edge length and direction. The feature templates consists of different conjunctions of these components. Other than features on the head word and the child word, we include features on in-between words and surrounding words as well. For PTB, our first-order model has 268 feature templates and 76,287,848 features; the second-order model has 380 feature templates and 95,796,140 features. The accuracy of our full-feature models is comparable or superior to previous results.

[7] http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html

6.3 Results

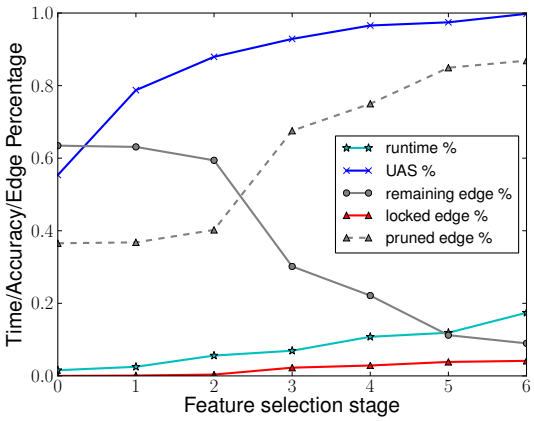


Figure 4: System dynamics on English PTB section 23. Time and accuracy are relative to those of the baseline model using full features. Red (locked), gray (undecided), dashed gray (pruned) lines correspond to edges shown in Figure 1.

In Table 1, we compare the dynamic parsing models with the full-feature models and the vine pruning cascade models for first-order and second-order

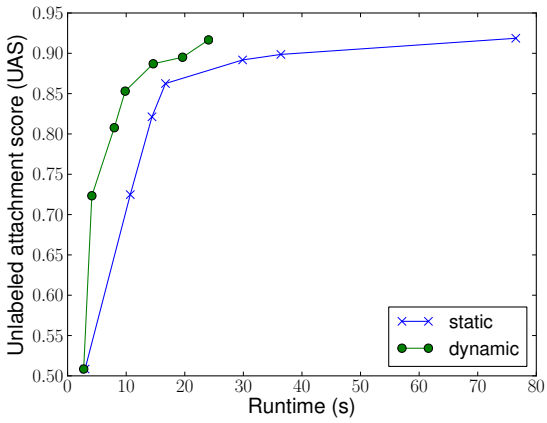


Figure 5: Pareto curves for the dynamic and static approaches on English PTB section 23.

parsing. The *speedup* for each language is defined as the speed relative to its full-feature baseline model. We take results reported by Rush and Petrov (2012) for the vine pruning model. As speed comparison for parsing largely relies on implementation, we also report the percentage of feature templates chosen for each sentence. The *cost* column shows the average number of feature templates computed for each sentence, expressed as a percentage of the number of feature templates if we had only pruned using the length dictionary filter.

From the table we notice that our first-order model's performance is comparable or superior to the vine pruning model, both in terms of speedup and accuracy. In some cases, the model with fewer features even achieves higher accuracy than the model with full features. The second-order model, however, does not work as well. In our experiments, the second-order model is more sensitive to false negatives, i.e. pruning of gold edges, due to larger error propagation than the first-order model. Therefore, to maintain parsing accuracy, the policy must make high-precision pruning decisions and becomes conservative. We could mitigate this by training the original parsing feature weights in conjunction with our policy feature weights. In addition, there is larger overhead during when checking non-projective edges and cycles.

We demonstrate the dynamics of our system in Figure 4 on PTB section 23. We show how the runtime, accuracy, number of locked edges and undecided edges change over the iterations in our first-order dynamic projective parsing. From iterations 1 to 6, we obtain parsing results from the non-projective parser; in iteration 7, we run the projective parser. The plot shows relative numbers (percentage) to the baseline model with full features. The number of remaining edges drops quickly after the second iteration. From Figure 3, however, we notice that even with the first feature group which only contains one feature template, the non-projective parser can almost achieve 50% accuracy. Thus, ideally, our policy should have locked that many edges after the first iteration. The learned policy does not imitate the expert perfectly, either because our policy features are not discriminative enough, or because a linear classifier is not powerful enough for this task.

Finally, to show the advantage of making dynamic decisions that consider the interaction among edges on the given input sentence, we compare our results with a static feature selection approach on PTB section 23. The static algorithm does no pruning except by the length dictionary at the start. In each iteration, instead of running a fast parser and making decisions online, it simply adds the next group of feature templates to all edges. By forcing both algorithms to stop after each stage, we get the Pareto curves shown in Figure 5. For a given level of high accuracy, our dynamic approach (black) is much faster than its static counterpart (blue).

7 Conclusion

In this paper we present a dynamic feature selection algorithm for graph-based dependency parsing. We show that choosing feature templates adaptively for each edge in the dependency graph greatly reduces feature computation time and in some cases improves parsing accuracy. Our model also makes it practical to use an even larger feature set, since features are computed only when needed. In future, we are interested in training parsers favoring the dynamic feature selection setting, for example, parsers that are robust to missing features, or parsers optimized for different stages.

Acknowledgements

This work was supported by the National Science Foundation under Grant No. 0964681. We thank the anonymous reviewers for very helpful comments.

References

P. Abbeel and A. Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of ICML*.

D. Benbouzid, R. Busa-Fekete, and B. Kégl. 2012. Fast classification using space decision DAGs. In *Proceedings of ICML*.

S. Bergsma and C. Cherry. 2010. Fast and accurate arc filtering for dependency parsing. In *Proceedings of COLING*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of ACL*.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, (71B):233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of COLING*.

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

Tianshi Gao and Daphne Koller. 2010. Active classification based on value of classifier. In *Proceedings of NIPS*.

Alexander Grubb and J. Andrew Bagnell. 2012. SpeedBoost: Anytime prediction with uniform near-optimality. In *AISTATS*.

He He, Hal Daumé III, and Jason Eisner. 2012. Cost-sensitive dynamic feature selection. In *ICML Inferning Workshop*.

Matti Kääriäinen. 2006. Lower bounds for reductions. Talk at the Atomic Learning Workshop (TTI-C), March.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, K. Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of EMNLP*.

N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt. 2004. Boosting structured prediction for imitation learning. In *Proceedings of ICML*.

B. Roark and K. Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of COLING*.

Stéphane. Ross, Geoffrey J. Gordon, and J. Andrew. Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of AISTATS*.

Alexander Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of NAACL*.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning : An Introduction*. MIT Press.

R. E. Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

Paul Viola and Michael Jones. 2004. Robust feal-time face detection. *International Journal of Computer Vision*, 57:137–154.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI*.

David Weiss and Ben Taskar. 2010. Structured prediction cascades. In *Proceedings of AISTATS*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines. In *Proceedings of IWPT*.

Semi-Supervised Representation Learning for Cross-Lingual Text Classification

Min Xiao and **Yuhong Guo**
Department of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
{minxiao,yuhong}@temple.edu

Abstract

Cross-lingual adaptation aims to learn a prediction model in a label-scarce target language by exploiting labeled data from a label-rich source language. An effective cross-lingual adaptation system can substantially reduce the manual annotation effort required in many natural language processing tasks. In this paper, we propose a new cross-lingual adaptation approach for document classification based on learning cross-lingual discriminative distributed representations of words. Specifically, we propose to maximize the log-likelihood of the documents from both language domains under a cross-lingual log-bilinear document model, while minimizing the prediction log-losses of labeled documents. We conduct extensive experiments on cross-lingual sentiment classification tasks of Amazon product reviews. Our experimental results demonstrate the efficacy of the proposed cross-lingual adaptation approach.

1 Introduction

With the rapid development of linguistic resources in different languages, developing cross-lingual natural language processing (NLP) systems becomes increasingly important (Bel et al., 2003; Shanahan et al., 2004). Recently, cross-lingual adaptation methods have been studied to exploit labeled information from an existing *source* language domain where labeled training data is abundant for use in a *target* language domain where annotated training data is scarce (Prettenhofer and Stein, 2010). Previous work has shown that cross-lingual adaptation can greatly reduce labeling effort for a variety of cross language NLP tasks such as document categorization (Bel et al., 2003; Amini et al., 2009), genre classification (Petrenz and Webber, 2012), and sentiment classification (Shanahan et al., 2004; Wei and Pal, 2010; Prettenhofer and Stein, 2010).

The fundamental challenge of cross-lingual adaptation stems from a lack of overlap between the feature space of the source language data and that of the target language data. To address this challenge, previous work in the literature mainly relies on automatic machine translation tools. They first translate all the text data from one language domain into the other and then apply techniques such as domain adaptation (Wan et al., 2011; Rigutini and Maggini, 2005; Ling et al., 2008) and multi-view learning (Amini et al., 2009; Guo and Xiao, 2012b; Wan, 2009) to achieve cross-lingual adaptation. However, machine translation tools may not be freely available for all languages. Moreover, translating all the text data in one language into the other language is too time-consuming in reality. As an economic alternative solution, cross-lingual representation learning has recently been used in the literature to learn language-independent representations of the data for cross language text classification (Prettenhofer and Stein, 2010; Petrenz and Webber, 2012).

In this paper, we propose to tackle cross language text classification by inducing cross-lingual predictive data representations with both labeled and unlabeled documents from the two language domains. Specifically, we propose a cross-lingual log-bilinear document model to learn distributed representations of words, which can capture both the semantic sim-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1465–1475,
Seattle, Washington, USA, 18-21 October 2013.

ilarities of words across languages and the predictive information with respect to the target classification task. We conduct the representation learning by maximizing the log-likelihood of all documents from both language domains under the cross-lingual log-bilinear document model and minimizing the prediction log-losses of labeled documents. We formulate the learning problem as a joint non-convex minimization problem and solve it using a local optimization algorithm. To evaluate the effectiveness of the proposed approach, we conduct experiments on the task of cross language sentiment classification of Amazon product reviews. The empirical results show the proposed approach is very effective for cross-lingual document classification, and outperforms other comparison methods.

2 Related Work

Much work in the literature proposes to construct cross-lingual representations by using aligned parallel data. Basically, they first employ machine translation tools to translate documents from one language domain to the other one and then induce low dimensional latent representations as interlingual representations (Littman et al., 1998; Vinokourov et al., 2002; Platt et al., 2010; Pan et al., 2011; Guo and Xiao, 2012a). Littman et al. (1998) proposed a cross-language latent semantic indexing method to induce interlingual representations by performing latent semantic indexing over a dual-language document-term matrix, where each dual-language document contains its original words and the corresponding translation text. Vinokourov et al. (2002) proposed a cross-lingual kernel canonical correlation analysis method, which learns two projections (one for each language) by conducting kernel canonical correlation analysis over a paired bilingual corpus and then uses the two projections to project documents from language-specific feature spaces to the shared multilingual semantic feature space. Platt et al. (2010) employed oriented principal component analysis (Diamantaras and Kung, 1996) over concatenated parallel documents, which learns a multilingual projection by simultaneously minimizing the projected distance between parallel documents and maximizing the projected covariance of documents across different languages. Pan et al. (2011) proposed a bi-view non-negative matrix tri-factorization method for cross-lingual sentiment classification on the parallel training and test data. Guo and Xiao (2012a) developed a transductive subspace representation learning method for cross-lingual text classification based on non-negative matrix factorization. Some other works exploited parallel data by using multilingual topic models to extract cross-language latent topics as interlingual representations (Mimno et al., 2009; Ni et al., 2011; Platt et al., 2010; Smet et al., 2011) and using neural probabilistic language modes to learn word embeddings as cross-lingual distributed representations (Klementiev et al., 2012). Most of them were developed by applying the latent Dirichlet allocation (LDA) model (Blei et al., 2003) in a multilingual setting, including the polylingual topic model (Mimno et al., 2009), the bilingual LDA model (Smet et al., 2011), and the multilingual LDA model (Ni et al., 2011). Platt et al. (2010) extended the probabilistic latent semantic analysis (PLSA) model (Hofmann, 1999) and presented two variants of multilingual topic models: the joint PLSA model and the coupled PLSA model. Recently, Klementiev et al. (2012) extended the neural probabilistic language model (Bengio et al., 2000) to induce cross-lingual word distributed representations on a set of word-level aligned parallel sentences. The applicability of these approaches however is limited by the availability of parallel corpus. Translating the whole set of documents to produce parallel corpus is too time-consuming, expensive and even practically impossible for some language pairs. We thus do not evaluate those approaches in our empirical study.

Another group of works propose to use bilingual dictionaries to learn interlingual representations (Gliozzo, 2006; Prettenhofer and Stein, 2010). Gliozzo (2006) first translated each term from one language to the other using a bilingual dictionary and used the translated terms to augment original documents. Then they conducted latent semantic analysis (LSA) over the document-term matrix with concatenated vocabularies to obtain interlingual representations. Prettenhofer and Stein (2010) proposed a cross-language structural correspondence learning (CL-SCL) method to induce language-independent features by using word translation oracles. They first selected a subset of source

language features, which have the highest mutual information with respect to the class labels in the labeled documents from the source language domain, to translate them into the target language domain, and then used these pivot pairs to induce cross-lingual representations by modeling the correlations between pivot features and non-pivot features. Our proposed approach shares a similarity with the CL-SCL method in (Prettenhofer and Stein, 2010) on only requiring a small amount of word translations. But our approach performs representation learning in a semi-supervised manner by directly incorporating discriminative information with respect to the target prediction task, while CL-SCL only exploits labels when selecting pivot features and the structural correspondence learning process is conducted in a fully unsupervised fashion.

Some other bilingual resources, such as multilingual WordNet (Fellbaum, 1998) and universal part-of-speech (POS) tags (Petrov et al., 2012), have also been exploited in the literature for interlingual learning. Gliozzo (2006) proposed to use MultiWordNet to map words from different languages to a common synset-id as language-sharing terms. A similar work was proposed in A.R. et al. (2012), which transformed words from different languages to WordNet synset identifiers as interlingual sense-based representations. However, multilingual WordNet resources are not always available for different language pairs. Recently, Petrenz and Webber (2012) used language-specific POS taggers to tag each word and then mapped those language-specific POS tags to twelve universal POS tags as interlingual features for cross language fine-grained genre classification. This approach requires a POS tagger for each language and it may be adversely affected by the POS tagging accuracy.

3 Semi-Supervised Representation Learning for Cross-Lingual Text Classification

In this section, we introduce a semi-supervised cross-lingual representation learning method and then use it for cross language text classification.

Assume we have ℓ_s labeled and u_s unlabeled documents in the source language domain $\mathcal{S}$ and ℓ_t labeled and u_t unlabeled documents in the target language domain $\mathcal{T}$. We assume all the documents are independent and identically distributed in each language domain, and each document $\mathbf{x}_i$ is represented as a bag of words, $\mathbf{x}_i = \{w_{i1}, w_{i2}, \ldots, w_{iN_i}\}$. We use $(\mathbf{x}_i^\ell, y_i)$ to denote the i-th labeled document and its label, and consider exploiting the labeled documents in the source domain $\mathcal{S}$ for learning classifiers in the target domain $\mathcal{T}$.

To build connections between the two language domains, we first construct a set of critical bilingual word pairs $M = \{(w_i^s, w_j^t)\}_{i=1}^m$, where w_i^s is a critical word in the source language domain, w_j^t is its translation in the target language domain, and m is the number of word pairs. Here being critical means the word should be discriminative for the prediction task and occur frequently in both language domains. Following the work (Prettenhofer and Stein, 2010), we select bilingual word pairs in a heuristic way. First we select a subset of words from the source language domain, which have the highest mutual information with the class labels in labeled source documents. The mutual information is computed based on the empirical distributions of words and labels in the labeled source documents. Then we translate the selected words into the target language using a translation tool to produce word pairs. Finally we produce the M set by eliminating any candidate pair (w^s, w^t), if either w^s occurs less than a predefined threshold value ϕ in all source language documents or w^t occurs less than ϕ in all target language documents. Given the constructed bilingual word pair set M, the words appearing in the source language documents but not in M can be put together to form a source specific vocabulary set $V_s = \{w_1^s, \ldots, w_{v_s}^s\}$. Similarly, the words appearing in the target language documents but not in M can be put together to form a target specific vocabulary set $V_t = \{w_1^t, \ldots, w_{v_t}^t\}$. An overall cross-lingual vocabulary set can then be constructed as $V = V_s \cup V_t \cup M$, which has a total of $v = v_s + v_t + m$ entries. This cross-lingual vocabulary set covers all words appearing in both domains, while mapping each bilingual pair in M into the same entry.

To tackle cross language text classification, we then propose a cross-lingual log-bilinear document model to learn a predictive cross-lingual representation of words, which maps each entry in the vocabulary set V to one row vector in a word embed-

ding matrix $R \in \mathbb{R}^{v \times k}$. Similar to the log-bilinear language model (Mnih and Hinton, 2007) and the log-bilinear document model (Maas et al., 2011), our proposed model learns a dense feature vector for each word to capture semantic similarities between the vocabulary entries. But unlike the previous two models which only work with a monolingual language, our model also captures semantic similarities across different languages. Moreover, we explicitly incorporate the label information into our proposed approach, rendering the induced word embeddings more discriminative to the target prediction task.

3.1 Cross-Lingual Word Embeddings

As mentioned above, we assume a unified embedding matrix R which contains the distributed vector representations of words in the two language domains. However, even in a unified representation space, the distribution of words in the two domains will be different. To capture the distribution divergence of the two domains and facilitate cross-lingual learning, we split the word embedding matrix into three parts: source language specific part $R_s \in \mathbb{R}^{v \times k_s}$, common part $R_c \in \mathbb{R}^{v \times k_c}$ and target language specific part $R_t \in \mathbb{R}^{v \times k_t}$, such that $k = k_s + k_c + k_t$. Intuitively, we assume that source language words contain no target language specific representations and target language words contain no source language specific representations. Thus for words in the two language domains, we retrieve their distributed vector representations from the embedding matrix R using two mapping functions, $\Phi_{\mathcal{S}}$ and $\Phi_{\mathcal{T}}$, one for each language domain. The two mapping functions are defined as

$$\Phi_{\mathcal{S}}(w) = [R_s(w), R_c(w), \mathbf{0}_t]^T \quad (1)$$

$$\Phi_{\mathcal{T}}(w) = [\mathbf{0}_s, R_c(w), R_t(w)]^T \quad (2)$$

where $\mathbf{0}_t$ is a k_t-dimensional row vector of zeros, $\mathbf{0}_s$ is a k_s-dimensional row vector of zeros, $R_s(w)$ denotes the row vector of R_s matrix corresponding to the word w, $R_c(w)$ denotes the row vector of R_c matrix corresponding to the word w, and $R_t(w)$ denotes the row vector of R_t matrix corresponding to the word w. It is easy to see that each pair of words in M will share the same vector from R_c. To encode more information into the common part of representation for better knowledge transfer from the source language domain to the target language domain, we assume $k_c \geq k_s$ and $k_c \geq k_t$. The form of three part feature representations has been exploited in previous work of domain adaptation with heterogeneous feature spaces (Duan et al., 2012). However, their approach simply duplicates the original features as language-specific representations, while we will automatically learn those three part latent representations in our approach.

3.2 Semi-Supervised Cross-Lingual Representation Learning

Given the word representation scheme above, we conduct cross-lingual representation learning by simultaneously maximizing the log-likelihood of all documents and the conditional likelihood of labeled documents from the two language domains

$$\max_{\theta} \sum_{\mathcal{L} \in \{\mathcal{S}, \mathcal{T}\}} \sum_{\mathbf{x}_i \in \mathcal{L}} \sum_{j=1}^{N_i} \log P_{\mathcal{L}}(w_{ij}|\theta) + \alpha \sum_{\mathcal{L} \in \{\mathcal{S}, \mathcal{T}\}} \sum_{\mathbf{x}_i^{\ell} \in \mathcal{L}} \log P_{\mathcal{L}}(y_i|\mathbf{x}_i^{\ell}, \theta) \quad (3)$$

where θ denotes the model parameters and α is a trade-off parameter. The first part of the objective function captures the likelihood of the documents being generated with the learned representation R. $P_{\mathcal{L}}(w_{ij}|\theta)$ is the probability of word w_{ij} appearing in the document $\mathbf{x}_i$ from the language domain $\mathcal{L}$, and is defined as

$$P_{\mathcal{L}}(w_{ij}|\theta) = \frac{\exp\left(-E_{\mathcal{L}}(w_{ij}, \theta)\right)}{\sum_{w' \in V} \exp\left(-E_{\mathcal{L}}(w', \theta)\right)} \quad (4)$$

The term $E_{\mathcal{L}}(w_{ij}, \theta)$ is a log-bilinear energy function, defined as

$$E_{\mathcal{L}}(w_{ij}, \theta) = -\mathbf{d}_i^T \Phi_{\mathcal{L}}(w_{ij}) - b_{w_{ij}} \quad (5)$$

where $\mathbf{d}_i$ is a k-dimensional weight vector for document $\mathbf{x}_i$ and $b_{w_{ij}}$ is the bias for word w_{ij}. Below we will use $\mathbf{b}$ to denote a v-dimensional vector containing all words' biases.

The second part of the objective function in (3) takes the label information into account and aims to render the latent word representations more task-predictive. We use a logistic regression model to

compute the conditional probability of the class label given the document with the induced word representations, such that

$$P_{\mathcal{L}}(y_i|\mathbf{x}_i^{\ell},\theta) = \frac{1}{1+\exp\left(-y_i\left(\mathbf{w}^T\Psi_{\mathcal{L}}(\mathbf{x}_i^{\ell})+q\right)\right)} \tag{6}$$

where $\mathbf{w}, q$ are model parameters of the logistic regression model, $\Psi_{\mathcal{L}}(\mathbf{x}_i)$ is the k-dimensional vector representation of the document $\mathbf{x}_i$ in the language domain $\mathcal{L}$. We compute $\Psi_{\mathcal{L}}(\mathbf{x}_i)$ by taking average over all words in the document $\mathbf{x}_i$ such as

$$\Psi_{\mathcal{L}}(\mathbf{x}_i) = \frac{1}{N_i}\sum_{j=1}^{N_i}\Phi_{\mathcal{L}}(w_{ij}) \tag{7}$$

By summing over all descriptions above, we can see that the proposed semi-supervised representation learning has a set of model parameters, $\theta = \{R, \{\mathbf{d}_i\}, \mathbf{b}, \mathbf{w}, q\}$. In order to avoid overfitting, we add regularization terms for the parameters $R, \{\mathbf{d}_i\}$ and $\mathbf{w}$, which leads to the final optimization problem below

$$\begin{aligned}\max_{\theta} & \sum_{\mathcal{L}\in\{\mathcal{S},\mathcal{T}\}}\sum_{\mathbf{x}_i\in\mathcal{L}}\Big(\sum_{j=1}^{N_i}\log P_{\mathcal{L}}(w_{ij}|\theta) - \gamma\|\mathbf{d}_i\|_2^2\Big) \\ & + \alpha\sum_{\mathcal{L}\in\{\mathcal{S},\mathcal{T}\}}\sum_{\mathbf{x}_i^{\ell}\in\mathcal{L}}\log P_{\mathcal{L}}(y_i|\mathbf{x}_i^{\ell},\theta) \\ & - \beta\|R\|_F^2 - \eta\|\mathbf{w}\|_2^2 \end{aligned} \tag{8}$$

where β, γ, η are trade-off parameters, $\|\cdot\|_F$ denote the Frobenius norm and $\|\cdot\|_2$ denote the Euclidean-norm. This objective function is not jointly convex in all model parameters. We develop a gradient-based iterative optimization procedure to seek a local optimal solution. We first randomly initialize the model parameters $\{\mathbf{d}_i\}, R, \mathbf{w}$ and set $\mathbf{b}$ and q to zeros. Then we iteratively make gradient-based updates over the model parameters until reach a local optimal solution.

3.3 Cross-Lingual Document Classification

After solving (8), we obtain a word embedding matrix R. The distributed vector representation of any given document can then be computed using Eq. (7) based on Eq. (1) or Eq. (2). Under the distributed vector representations of the documents in both language domains, we perform cross-lingual document classification by training a supervised classification model using labeled data from both language domains and then applying it to classify test documents in the target language domain .

4 Experiments

We empirically evaluate the proposed approach using the cross language sentiment classification tasks of Amazon product reviews in four languages. In this section, we report our experimental results.

4.1 Dataset

We used the multilingual sentiment classification dataset[1] provided by Prettenhofer and Stein (2010), which contains Amazon product reviews in four different languages, English (E), French (F), German (G) and Japanese (J). The English product reviews were sampled from previous cross-domain sentiment classification datasets (Blitzer et al., 2007), while the other three language product reviews were crawled from Amazon by the authors in November 2009. In the dataset, each language contains three categories of product reviews, Books (B), DVD (D) and Music (M). Each language-category pair contains a balanced training set and test set, each of which consists of 1000 positive reviews and 1000 negative reviews. Each review is represented as a unigram bag-of-word feature vector with term-frequency values. Following the work (Prettenhofer and Stein, 2010), we used the original English reviews as the source language while treating the other three languages as target languages. Thus, we construct nine cross language sentiment classification tasks (GB, GD, GM, FB, FD, FM, JB, JD, JM), one for each target language-category pair. For example, the task *GB* means that the target language is *German* and the training and test data are samples from *Books* reviews.

4.2 Approaches

We compare our proposed semi-supervised cross-lingual representation learning (**CL-RL**) approach to the following approaches for cross-lingual document classification.

[1] http://www.webis.de/research/corpora/

Table 1: Average classification accuracies and standard deviations for the 9 cross-lingual sentiment classification tasks. The bold format indicates that the difference between the results of *CL-RL* and *MT* is significant with $p < 0.05$ under a McNemar paired test for labeling disagreements.

Task	TB	CL-Dict	CLD-LSA	CL-SCL	MT	CL-RL
GB	66.25±0.64	69.40±0.61	70.30±0.44	73.78±0.32	78.05±0.64	**79.89**±0.30
GD	63.16±0.66	66.37±0.63	66.85±0.46	71.99±0.25	75.75±0.58	**77.14**±0.16
GM	65.42±0.77	68.81±0.51	68.93±0.58	71.58±0.35	74.85±0.62	**77.27**±0.16
FB	65.98±0.51	69.35±0.48	69.98±0.51	73.89±0.16	78.00±0.49	78.25±0.32
FD	63.76±0.37	67.96±0.60	68.88±0.43	73.79±0.28	75.75±0.71	74.83±0.30
FM	65.94±0.56	67.98±0.69	68.42±0.60	71.20±0.28	74.85±0.49	**78.71**±0.32
JB	63.86±0.80	59.40±0.29	62.62±0.62	62.49±0.23	67.20±0.80	**71.11**±0.21
JD	63.59±0.74	62.13±0.26	63.87±0.72	65.54±0.29	67.70±0.57	**73.12**±0.23
JM	65.84±0.90	63.01±0.46	65.67±0.72	65.49±0.36	68.30±0.61	**74.38**±0.40

- **TB:** This is a target baseline method, which trains a supervised monolingual classifier on the labeled training data from the target language domain without representation learning.
- **CL-Dict:** This is a simple baseline comparison method, which uses the bilingual word pairs directly to align features from different language domains into a unified feature dictionary and then trains a supervised classifier on this aligned feature space with labeled training data from both language domains.
- **CLD-LSA:** This is the cross-lingual representation learning method developed in (Gliozzo, 2006), which first translates each document from one language into the other language via a bilingual dictionary to produce augmenting features, and then performs latent semantic analysis (LSA) over the augmented bilingual document-term matrix.
- **CL-SCL:** This is the cross language structural correspondence learning method developed in (Prettenhofer and Stein, 2010).
- **MT:** This is a machine translation based comparison method, which first uses an existing machine translation tool (google translation) to translate the target language documents into the source language and then trains a monolingual classifier with labeled training data from both domains in the source language.

In all experiments, we used a linear support vector machine (SVM) for sentiment classification. For implementation, we used the liblinear package (Fan et al., 2008) with all of its default parameters. For the *CL-SCL* method, we used the same parameter setting as suggested in the paper (Prettenhofer and Stein, 2010): the number of pivot features is set as 450, the threshold value for selecting pivot features is 30, and the reduced dimensionality after singular value decomposition is 100. For the *CLD-LSA* method, we set the dimensionality of latent representation as 1000. Similarly, for our proposed approach, we built the cross-lingual vocabulary M by setting $m = 450$ and $\phi = 30$. For our representation learning, we set $\alpha = 1$, $\beta = \gamma = \eta = 1e^{-4}$, and set k_s, k_c, k_t to be 25, 50, 25, respectively. The values of α, β, γ and η are selected using the first cross language classification task GB. We selected the α value from the set $\{0.01, 0.1, 1, 10, 100\}$ and selected β, γ, η values from the set $\{1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-5}\}$ by repeating the experiment three times with random data partitions and choosing the parameter values that led to the best average classification accuracy.

4.3 Classification Accuracy

For each of the nine cross language sentiment classification tasks with different target language-category pairs, we used the training set in the source language domain (English) as labeled data while treating the test set in the source language domain as unlabeled.

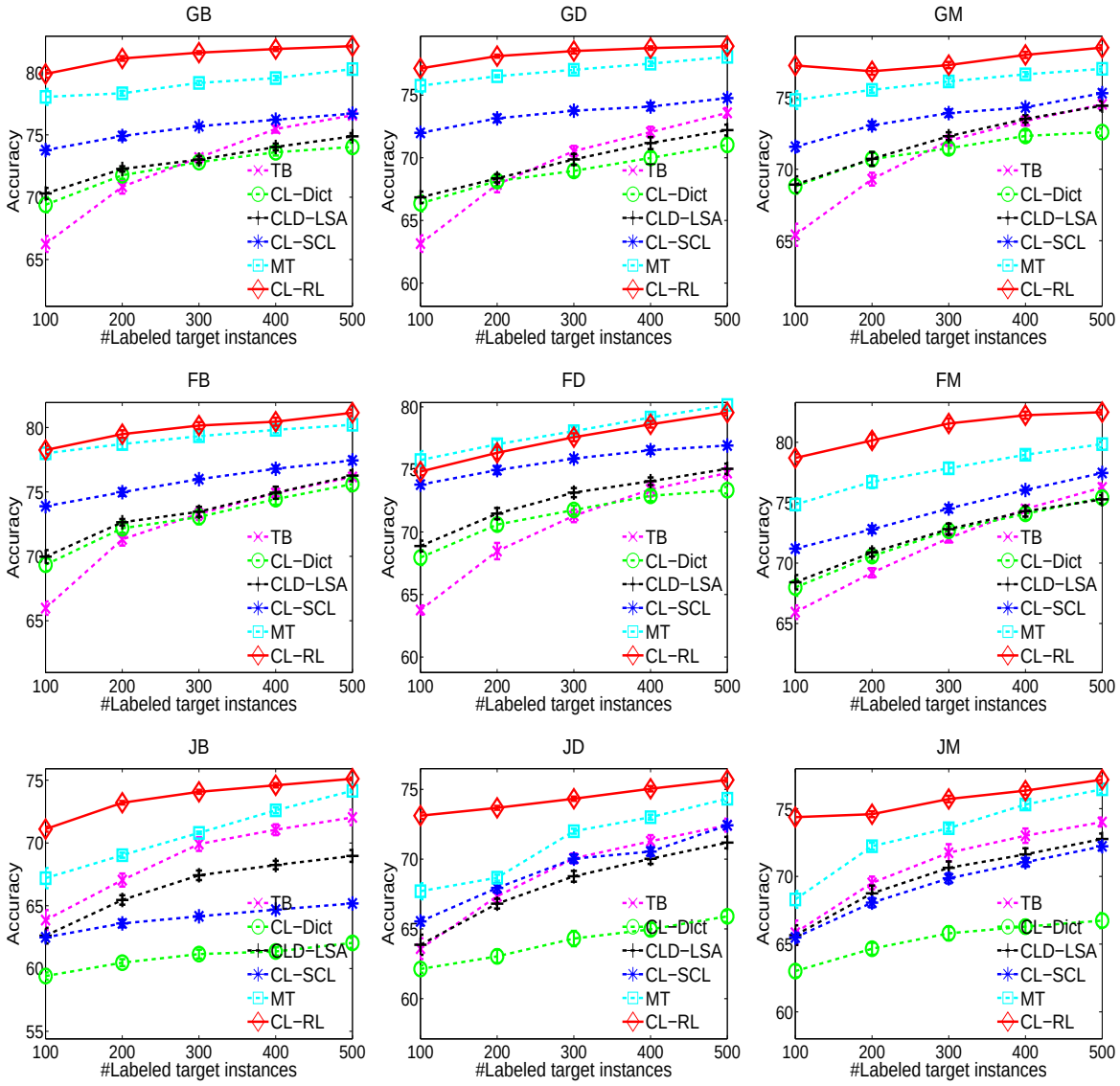


Figure 1: Average classification accuracies and standard deviations for 10 runs with respect to different numbers of labeled training documents in the target language domain.

For target language domain, we used the test set as test data while randomly selecting 100 documents from the training set as labeled data and treating the rest as unlabeled data. Thus, for each task, we have 2000 labeled documents and 2000 unlabeled documents from the source language domain, and 100 labeled and 1900 unlabeled documents from the target language domain for training. We have 2000 test documents from the target language domain as testing data. In each experiment, a classifier is produced by each approach with the training data and tested on the testing data. We repeated each experiment 10 times with different random selections of 100 labeled training documents from the target language domain. The average classification accuracies and standard deviations are reported in Table 1.

From Table 1, we can see that the proposed semi-supervised cross-lingual representation learning approach, CL-RL, clearly outperforms all other comparison methods on eight out of the nine tasks. The target baseline *TB* performs poorly on all the nine tasks, which suggests that 100 labeled instances from the target language is far from enough to obtain an accurate sentiment classifier in the target lan-

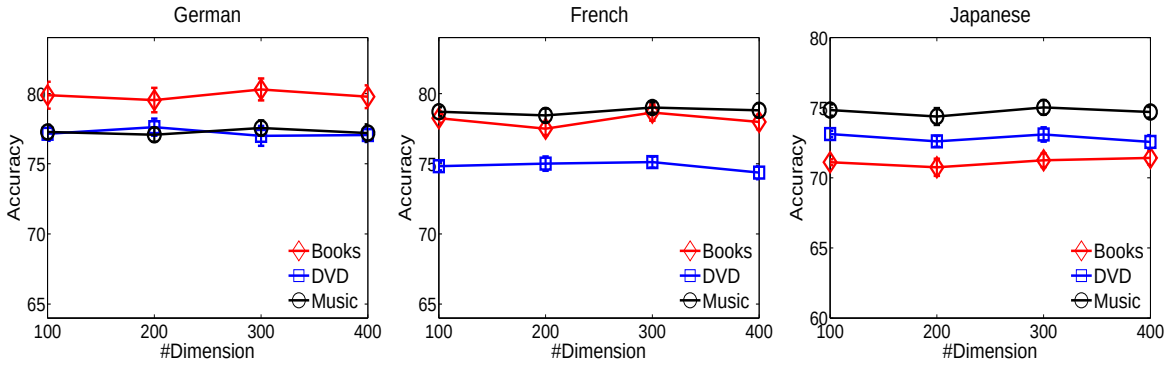


Figure 2: Average classification accuracy and standard deviation results for the proposed approach over 10 runs with respect to different dimensionality for the induced cross-lingual representations.

guage domain. By exploiting the large amount of labeled training data from the source language domain, even the simple cross-lingual adaptation approach, *CL-Dict*, produces effective improvements over *TB*. However, its performance is not consistent across the nine tasks. It has inferior performance than *TB* on the three tasks of adapting English to the Japanese language domain. This suggests the simple bilingual word-pair based feature space unification method is far from ideal for providing effective cross-lingual representations, especially when two languages (English, Japanese) are very different. With a better designed representation learning, *CLD-LSA* outperforms *CL-Dict* on all the nine tasks, but the improvements are very small on some tasks (e.g., GM). *CL-SCL* not only outperforms *CL-Dict* on all tasks, but also performs much better than *CLD-LSA* on most tasks. Its performance nevertheless is inferior to the method of *MT*. Though *MT* can greatly increase the test accuracies comparing to the other four methods, *TB*, *CL-Dict*, *CLD-LSA*, and *CL-SCL*, the benefit is obtained at the cost of whole document translations. In contrast, our proposed approach does not require whole document translations, but relies on the same simple word-pair translations used in *CL-Dict*. It however consistently and significantly outperforms *TB*, *CL-Dict*, *CLD-LSA*, and *CL-SCL* on all tasks, and outperforms *MT* on eight out of the nine tasks.

We also conduct significance tests for our proposed approach and *MT* using a McNemar paired test for labeling disagreements (Gillick and Cox, 1989). The results in bold format indicate that they are significant with $p < 0.05$. All these results demonstrate the efficacy of our cross-lingual representation learning method.

4.4 Classification Accuracy vs the Number of Labeled Target Documents

Next, we investigated the performance of the six approaches by varying the number of labeled training documents from the target language domain. We maintained the same experimental setting as before, but investigated a range of different values, $\ell_t = \{100, 200, 300, 400, 500\}$, as the number of labeled training documents from the target language domain. In each experiment, for a given value ℓ_t, we randomly selected ℓ_t documents from the training set of the target language domain as labeled data and used the rest as unlabeled data. We still performed prediction on the same 2000 test documents in the target language domain. We repeated each experiment 10 times based on different random selections of the labeled training data from the target language domain. The average classification accuracies and standard deviations across different ℓ_t values for all comparison methods on all the nine tasks are plotted in Figure 1.

We can see when the number of labeled target documents is small, *TB* performs poorly, especially for the first six tasks (GB, GD, GM, FB, FD, FM). By increasing the size of labeled target training data, *TB* can greatly increase its prediction accuracies and even outperform the *CL-Dict* method. The simple *CL-Dict* method has inconsistent performance across the nine tasks. Its performance is better than

TB when the labeled training data in the target language domain is very limited and is poor than *TB* when the labeled target data reaches 300 for the six tasks using German and French as target languages. Moreover, when adapting a system from English to a much more different target language (Japanese), *CL-Dict* produces much lower accuracies for all the three tasks comparing with *TB*. These results show that *CL-Dict* has very limited capacity on transferring labeled information from a related source language domain. Similar performance is observed for *CLD-LSA*. With a more sophisticated representation learning, the *CL-SCL* method consistently outperforms *CL-Dict*. However, it produces inferior performance than *CLD-LSA* on the tasks of *JB* and *JM*. By using more translation resources, the *MT* method outperforms *TB*, *CL-Dict*, *CLD-LSA*, *CL-SCL* in all the nine tasks across almost all scenarios. Our proposed method *CL-RL* significantly outperforms all the other five comparison methods across all experiments except on the task of FD, where *MT* produces similar performance. Moreover, it is especially important to notice that *CL-RL* achieves high test accuracies even when the number of labeled target instances is small. This is important for transferring knowledge from a source language to reduce the labeling effort in the target language.

4.5 Sensitivity Analysis

We also investigated the sensitivity of the proposed approach over the dimensionality of the induced cross-lingual representations. We used the same experimental setting as before, and conducted experiments with a set of different dimensionality values, $k = \{100, 200, 300, 400\}$. For each value k, we set $k_s = 0.25k, k_c = 0.5k, k_t = 0.25k$. We repeated each experiment for 10 times based on different random selections of labeled target training data and plotted the average prediction accuracies and standard deviations in Figure 2 for all the nine cross-lingual sentiment classification tasks. We can see the proposed approach produces stable accuracy results across the range of different k values. This suggests the proposed approach is not very sensitive to the dimensionality of the cross-lingual embedding features within the considered range of values, and with a small dimensionality of 100, the induced representation can already perform very well.

4.6 Cross-Lingual Word Representations

Finally, we used the first task *GB*, which adapts the *Books* reviews from English to German, to gain intuitive understandings over the learned cross-lingual word representations. Given an English word as seed word, we find its five closest neighboring English words and German words according to the Euclidean distances calculated in the induced cross-lingual representation space. We present a few results in Table 2. From Table 2, we can see that the retrieved words in both language domains are semantically close to the seed words, which indicates that our proposed method can capture semantic similarities of words not only in a monolingual setting but also in a multilingual setting.

5 Conclusion

In this paper, we proposed a semi-supervised cross-lingual representation learning approach to address cross-lingual text classification. The distributed word representation induced by the proposed approach can capture semantic similarities of words across languages while maintaining predictive information with respect to the target classification tasks. To evaluate the proposed approach, we conducted experiments on nine cross language sentiment classification tasks constructed from the Amazon product reviews in four languages, comparing to a number of comparison methods. The empirical results showed that the proposed approach can produce effective cross-lingual adaptation performance and significantly outperform other comparison methods.

References

M. Amini, N. Usunier, and C. Goutte. Learning from multiple partially observed views - an application to multilingual text categorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

B. A.R., A. Joshi, and P. Bhattacharyya. Cross-lingual sentiment analysis for indian languages using linked wordnets. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2012.

N. Bel, C. Koster, and M. Villegas. Cross-lingual

Table 2: Examples of source seed words together with five closest English words and five closest German words estimated using the Euclidean distance in the cross-lingual representation space on the task *GB*.

books		**absolutely**		**love**	
English	German	English	German	English	German
books	buch	absolutely	absolut	love	liebe
book	bücher	definitely	absolute	loved	lieben
text	text	completely	definitiv	like	wie
page	blatt	certainly	komplett	fond	wieder
words	wörter	totally	sicher	feel	fühlen
expensive		**good**		**not**	
English	German	English	German	English	German
expensive	teuer	good	gut	not	nicht
expense	höher	better	besser	no	nie
overpriced	höchsten	well	nett	cannot	nein
costly	hoch	nice	großartig	non	keine
price	preis	great	größten	never	keines

text categorization. In *Proceedings of European Conference on Digital Libraries (ECDL)*, 2003.

Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the Annual Meeting of the Asso. for Computational Linguistics (ACL)*, 2007.

K. Diamantaras and S. Kung. *Principal component neural networks: theory and applications*. Wiley-Interscience, 1996.

L. Duan, D. Xu, and I. Tsang. Learning with augmented features for heterogeneous domain adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874, 2008.

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

L. Gillick and S. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1989.

A. Gliozzo. Exploiting comparable corpora and bilingual dictionaries for cross-language text categorization. In *Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics (ICCL-ACL)*, 2006.

Y. Guo and M. Xiao. Transductive representation learning for cross-lingual text classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2012a.

Y. Guo and M. Xiao. Cross language text classification via subspace co-regularized multi-view learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012b.

T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 1999.

A. Klementiev, I. Titov, and B. Bhattarai. Inducing crosslingual distributed representations of words.

In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2012.

X. Ling, G. Xue, W. Dai, Y. Jiang, Q. Yang, and Y. Yu. Can chinese web pages be classified with english data source? In *Proceedings of the International Conference on World Wide Web (WWW)*, 2008.

M. Littman, S. Dumais, and T. Landauer. *Automatic Cross-Language Information Retrieval using Latent Semantic Indexing*, chapter 5, pages 51–62. Kluwer Academic Publishers, 1998.

A. Maas, R. Daly, P. Pham, D. Huang, A. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, 2011.

D. Mimno, H. Wallach, J. Naradowsky, D. Smith, and A. McCallum. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, 2009.

A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

X. Ni, J. Sun, J. Hu, and Z. Chen. Cross lingual text classification by mining multilingual topics from wikipedia. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2011.

J. Pan, G. Xue, Y. Yu, and Y. Wang. Cross-lingual sentiment classification via bi-view non-negative matrix tri-factorization. In *Proceedings of the Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD)*, 2011.

P. Petrenz and B. Webber. Label propagation for fine-grained cross-lingual genre classification. In *Proceedings of the NIPS xLiTe workshop*, 2012.

S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2012.

J. Platt, K. Toutanova, and W. Yih. Translingual document representations from discriminative projections. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2010.

P. Prettenhofer and B. Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010.

L. Rigutini and M. Maggini. An em based training algorithm for cross-language text categorization. In *Proceedings of the Web Intelligence Conference*, 2005.

J. Shanahan, G. Grefenstette, Y. Qu, and D. Evans. Mining multilingual opinions through classification and translation. In *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text*, 2004.

W. Smet, J. Tang, and M. Moens. Knowledge transfer across multilingual corpora via latent topics. In *Proceedings of the Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD)*, 2011.

A. Vinokourov, J. Shawe-taylor, and N. Cristianini. Inferring a semantic representation of text via cross-language correlation analysis. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

C. Wan, R. Pan, and J. Li. Bi-weighting domain adaptation for cross-language text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

X. Wan. Co-training for cross-lingual sentiment classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2009.

B. Wei and C. Pal. Cross lingual adaptation: An experiment on sentiment classifications. In *Proceedings of the Annual Meeting of the Asso. for Computational Linguistics (ACL)*, 2010.

Using crowdsourcing to get representations based on regular expressions

Anders Søgaard and Hector Martinez and Jakob Elming and Anders Johannsen
Center for Language Technology
University of Copenhagen
DK-2300 Copenhagen S
{soegaard|alonso|zmk867|ajohannsen}@hum.ku.dk

Abstract

Often the bottleneck in document classification is finding good representations that zoom in on the most important aspects of the documents. Most research uses n-gram representations, but relevant features often occur discontinuously, e.g., *not... good* in sentiment analysis. In this paper we present experiments getting experts to provide regular expressions, as well as crowdsourced annotation tasks from which regular expressions can be derived. Somewhat surprisingly, it turns out that these crowdsourced feature combinations outperform automatic feature combination methods, as well as expert features, by a very large margin and reduce error by 24-41% over n-gram representations.

1 Introduction

Finding good representations of classification problems is often glossed over in the literature. Several authors have emphasized the need to pay more attention to finding such representations (Wagstaff, 2012; Domingos, 2012), but in document classification most research still uses n-gram representations.

This paper considers two document classification problems where such representations seem inadequate. The problems are answer scoring (Burstein et al., 1998), on data from stackoverflow.com, and multi-attribute sentiment analysis (McAuley et al., 2012). We argue that in order to adequately represent such problems we need *discontinuous* features, i.e., regular expressions.

The problem with using regular expressions as features is of course that even with a finite vocabulary we can generate infinitely many regular expressions that match our documents. We suggest to use expert knowledge or crowdsourcing in the loop. In particular we present experiments where standard representations are augmented with features from a few hours of manual work, by machine learning experts or by turkers.

Somewhat surprisingly, we find that features derived from crowdsourced annotation tasks lead to the best results across the three datasets. While crowdsourcing of annotation tasks has become increasing popular in NLP, this is, to the best of our knowledge, the first attempt to crowdsource the problem of finding good representations.

1.1 Related work

Musat et al. (2012) design a collaborative two-player game for sentiment annotation and collecting a sentiment lexicon. One player guesses the sentiment of a text and picks a word from it that is representative of its sentiment. The other player also provides a guess observing only this word. If the two guesses agree, both players get a point. The idea of gamifying the problem of finding good representations goes beyond crowdsourcing, but is not considered here. Boyd-Graber et al. (2012) crowdsource the feature weighting problem, but using standard representations. The work most similar to ours is probably Tamuz et al. (2011), who learn a 'crowd kernel' by asking annotators to rate examples by similarity, providing an embedding that promotes feature combinations deemed relative when measuring similarity.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1476–1480,
Seattle, Washington, USA, 18-21 October 2013.

			BoW		*Exp*		*AMT*	
	n	$P(1)$	m	μ_x	m	μ_x	m	μ_x
STACKOVERFLOW	97,519	0.5013	30,716	0.00131	1,156	0.1380	172,691	0.00331
TASTE	152,390	0.5003	38,227	0.00095	666	0.10631	114,588	0.00285
APPEARANCE	152,331	0.5009	37,901	0.00097	650	0.14629	102,734	0.00289

Table 1: Characteristics of the $n \times m$ data sets

2 Experiments

Data The three datasets used in our experiments come from two sources, namely stackoverflow.com and ratebeer.com. The two beer review datasets (TASTE and APPEARANCE) are described in McAuley et al. (2012) and available for download.[1] Each input example is an unstructured review text, and the associated label is the score assigned to taste or appearance by the reviewer. We randomly sample about 152k data points, as well as 500 examples for experiments with experts and turks.

We extracted the STACKOVERFLOW dataset from a publicly available data dump,[2], and we briefly describe our sampling process here. We select pairs of answers, where one is ranked higher than the other by stackoverflow.com users. Obviously the answers submitted first have a better chance of being ranked highly, so we also require that the highest ranked answer was submitted last. From this set of answer pairs, we randomly sample 97,519 pairs, as well as 500 examples for our experiments with experts and turks.

Our experiments are classification experiments using the same learning algorithm in all experiments, namely L_1-regularized logistic regression. We don't set any parameters The only differences between our systems are in the feature sets. Results are from 5-fold cross-validation. The four feature sets are described below: *BoW*, *HI*, *Exp* and *AMT*.

For motivating using regular expressions, consider the following sentence from a review of John Harvard's Grand Cru:

(1) Could have been more flavorful.

The only word carrying direct sentiment in this sentence is *flavorful*, which is positive, but the sentence is a negative evaluation of the Grand Cru's taste. The trigram *been more flavorful* seems negative at first, but in the context of negation or in a comparative, it can become positive again. However, note that this trigram may occur discontinuously, e.g., in *been less watery and more flavorful*. In order to match such occurrences, we need simple regular expressions, e.g.,:

```
been.*more.*flavorful
```

This is exactly the kind of regular expressions we asked experts to submit, and that we derived from the crowdsourced annotation tasks. Note that the sentence says nothing about the beer's appearance, so this feature is only relevant in TASTE, not in APPEARANCE.

BoW **and** *BoW+HI* Our most simple baseline approach is a bag-of-words model of unigram features (*BoW*). We lower-case our data, but leave in stop words. We also introduce a semantically enriched unigram model *(BoW)+HI*, where in addition to representing what words occur in a text, we also represent what Harvard Inquirer (HI)[3] word classes occur in it. The HI classes are used to generate features from the crowdsourced annotation tasks, so the semantically enriched unigram model is an important baseline in our experiments below.

BoW+Exp In order to collect regular expressions from experts, we set up a web interface for querying held-out portions of the datasets with regular expressions that reports how occurrences of the submitted regular expressions correlate with class. We used the Python `re` syntax for regular expressions after augmenting word forms with POS and semantic classes from the HI. Few of the experts made use of the POS tags, but many regular expressions included references to HI classes.

[1] http://snap.stanford.edu/data/web-RateBeer.html

[2] http://www.clearbits.net/torrents/2076-aug-2012

[3] http://www.wjh.harvard.edu/ inquirer/homecat.htm

Regular expressions submitted by participants were visible to other participants during the experiment, and participants were allowed to work together. Participants had 15 minutes to familiarize themselves with the syntax used in the experiments. Each query was executed in 2-30 seconds.

Seven researchers and graduate students spent five effective hours querying the datasets with regular expressions. In particular, they spent three hours on the Stack Exchange dataset, and one hour on each of the two RateBeer datasets. One had to leave an hour early. So, in total, we spent 20 person hours on Stack Exchange, and seven person hours on each of the RateBeer datasets. In the five hours, we collected 1,156 regular expressions for the STACKOVERFLOW dataset, and about 650 regular expressions for each of the two RateBeer datasets. *Exp* refers to these sets of regular expressions. In our experiments below we concatenate these with the *BoW* features to form *BoW+Exp*.

BoW+AMT For each dataset, we also had 500 held-out examples annotated by three turkers each, using Amazon Mechanical Turk,[4] obtaining 1,500 HITs for each dataset. The annotators were presented with each text, a review or an answer, twice: once as running text, once word-by-word with bullets to tick off words. The annotators were instructed to tick off words or phrases that they found predictive of the text's sentiment or answer quality. They were not informed about the class of the text. We chose this annotation task, because it is relatively easy for annotators to mark spans of text with a particular attribute. This set-up has been used in other applications, including NER (Finin et al., 2010) and error detection (Dahlmeier et al., 2013). The annotators were constrained to tick off at least three words, including one closed class item (closed class items were colored differently). Finally, we only used annotators with a track record of providing high-quality annotations in previous tasks. It was clear from the average time spent by annotators that annotating STACKOVERFLOW was harder than annotating the Ratebeer datasets. The average time spent on a Ratebeer HIT was 44s, while for STACKOVERFLOW it was 3m:8s. The mean number of words ticked off was between 5.6 and 7, with more words ticked off in STACKOVERFLOW. The maximum number of words ticked off by an annotator was 41. We spent $292.5 on the annotations, including a trial round. This was supposed to match, roughly, the cost of the experts consulted for *BoW+Exp*.

[4] www.mturk.com

	BoW	*HI*	*Exp*	*AMT*
STACKOVERF	0.655	0.654	0.683	**0.739**
TASTE	0.798	0.797	0.798	**0.867**
APPEARANCE	0.758	0.760	0.761	**0.859**

Table 2: Results using all features

The features generated from the annotations were constructed as follows: We use a sliding window of size 3 to extract trigrams over the possibly discontinuous words ticked off by the annotators. These trigrams were converted into regular expressions by placing Kleene stars between the words. This gives us a manually selected subset of skip trigrams. For each skip trigram, we add copies with one or more words replaced by one of their HI classes.

Feature combinations This subsection introduces some harder baselines for our experiments, considered in Experiment #2. The simplest possible way of combining unigram features is by considering n-gram models. An n-gram extracts features from a sliding window (of size n) over the text. We call this model *BoW*$(N = n)$. Our *BoW*$(N = 1)$ model takes word forms as features, and there are obviously more advanced ways of automatically combining such features.

Kernel representations We experimented with applying an approximate feature map for the additive χ^2-kernel. We used two sample steps, resulting in $4N + 1$ features. See Vedaldi and Zimmerman (2011) for details.

Deep features We also ran denoising autoencoders (Pascal et al., 2008), previously applied to a wide range of NLP tasks (Ranganath et al., 2009; Socher et al., 2011; Chen et al., 2012), with $2N$ nodes in the middle layer to obtain a deep representation of our datasets from χ^2-*BoW* input. The network was trained for 15 epochs. We set the drop-out rate to 0.0 and 0.3.

Summary of feature sets The feature sets – *BoW*,

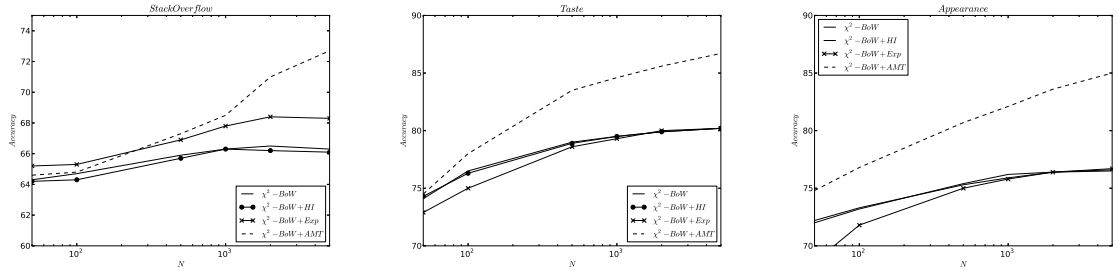


Figure 1: Results selecting N features using χ^2 (left to right): STACKOVERFLOW, TASTE, and APPEARANCE. The x-axis is logarithmic scale.

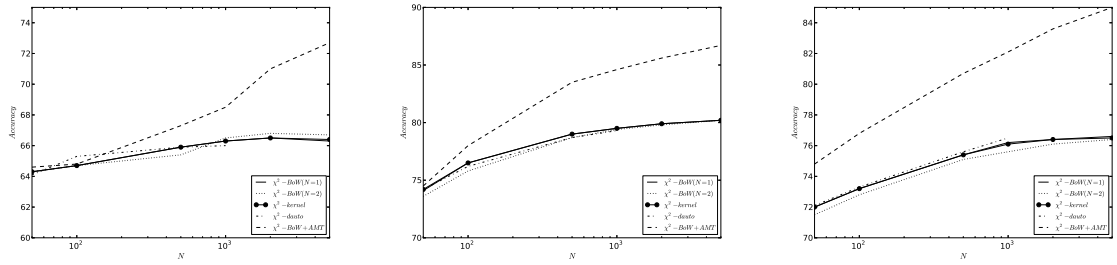


Figure 2: Results using different feature combination techniques (left to right): STACKOVERFLOW, TASTE, and APPEARANCE. The x-axis is logarithmic scale.

Exp and *AMT* – are very different. Their characteristics are presented in Table 1. $P(1)$ is the class distribution, e.g., the prior probability of positive class. n is the number of data points, m the number of features. Finally, μ_x is the average density of data points. One observation is of course that the expert feature set *Exp* is much smaller than *BoW* and *AMT*, but note also that the expert features fire about 150 times more often on average than the BoW features. *HI* is only a small set of additional features.

3 Results

Experiment #1: *BoW* **vs.** *Exp* **and** *AMT* We present results using all features, as well as results obtained after selecting k features as ranked by a simple χ^2 test. The results using all collected features are presented in Table 2. The error reduction on STACKOVERFLOW when adding crowdsourced features to our baseline model (*BoW+AMT*), is 24.3%. On TASTE, it is 34.2%. On APPEARANCE, it is 41.0%.

The *BoW+AMT* feature set is bigger than those of the other models. We therefore report results using the top-k features as ranked by a simple χ^2 test. The result curves are presented in the three plots in Fig. 1. With +500 features, *BoW+AMT* outperforms the other models by a large margin.

Experiment #2: *AMT* **vs. more baselines** The *BoW* baseline uses a standard representation that, while widely used, is usually thought of as a weak baseline. *BoW+HIT* did not provide a stronger baseline. We also show that bigram features, kernel-based decomposition and deep features do not provide much stronger baselines either. The result curves are presented in the three plots in Fig. 2. *BoW+AMT* is still significantly better than all other models with +500 features. Since autoencoders are consistently worse than denoising autoencoders (drop-out 0.3), we only plot denoising autoencoders.

4 Conclusion

We presented a new method for deriving feature representations from crowdsourced annotation tasks and showed how it leads to 24%-41% error reductions on answer scoring and multi-aspect sentiment analysis problems. We saw no significant improvements using features contributed by experts, kernel representations or learned deep representations.

References

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daume. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *NAACL*.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated scoring using a hybrid feature identification technique. In *ACL*.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *ICML*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English. In *Workshop on Innovative Use of NLP for Building Educational Applications, NAACL*.

Pedro Domingos. 2012. A few useful things to know about machine learning. In *CACM*.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In *NAACL Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *ICDM*.

Claudiu-Christian Musat, Alireza Ghasemi, and Boi Faltings. 2012. Sentiment analysis using a novel human computation game. In *Workshop on the People's Web Meets NLP, ACL*.

Vincent Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.

Rajesh Ranganath, Dan Jurafsky, and Dan McFarland. 2009. It's not you, it's me: detecting flirting and its misperception in speed-dates. In *NAACL*.

Richard Socher, Eric Huan, Jeffrey Pennington, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*.

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. 2011. Adaptively learning the crowd kernel. In *ICML*.

Andrea Vedaldi and Andrew Zisserman. 2011. Efficient additive kernels via explicit feature maps. In *CVPR*.

Kiri Wagstaff. 2012. Machine learning that matters. In *ICML*.

Overcoming the Lack of Parallel Data in Sentence Compression

Katja Filippova and **Yasemin Altun**
Google
Brandschenkestr. 110
Zürich, 8004 Switzerland
katjaf|altun@google.com

Abstract

A major challenge in supervised sentence compression is making use of rich feature representations because of very scarce parallel data. We address this problem and present a method to automatically build a compression corpus with hundreds of thousands of instances on which deletion-based algorithms can be trained. In our corpus, the syntactic trees of the compressions are subtrees of their uncompressed counterparts, and hence supervised systems which require a structural alignment between the input and output can be successfully trained. We also extend an existing unsupervised compression method with a learning module. The new system uses structured prediction to learn from lexical, syntactic and other features. An evaluation with human raters shows that the presented data harvesting method indeed produces a parallel corpus of high quality. Also, the supervised system trained on this corpus gets high scores both from human raters and in an automatic evaluation setting, significantly outperforming a strong baseline.

1 Introduction and related work

Sentence compression is a paraphrasing task where the goal is to generate sentences shorter than given while preserving the essential content. A robust compression system would be useful for mobile devices as well as a module in an extractive summarization system (Mani, 2001). Although a compression may differ lexically and structurally from the source sentence, to date most systems are extractive and proceed by deleting words from the input (Knight & Marcu, 2000; Dorr et al., 2003; Turner & Charniak, 2005; Clarke & Lapata, 2008; Berg-Kirkpatrick et al., 2011, inter alia). To decide which words, dependencies or phrases can be dropped, (i) rule-based approaches (Grefenstette, 1998; Jing & McKeown, 2000; Dorr et al., 2003; Zajic et al., 2007), (ii) supervised models trained on parallel data (Knight & Marcu, 2000; Turner & Charniak, 2005; McDonald, 2006; Gillick & Favre, 2009; Galanis & Androutsopoulos, 2010, inter alia) and (iii) unsupervised methods which make use of statistics collected from non-parallel data (Hori & Furui, 2004; Zajic et al., 2007; Clarke & Lapata, 2008; Filippova & Strube, 2008) have been investigated. Since it is infeasible to manually devise a set of accurate deletion rules with high coverage, recent research has been devoted to developing statistical methods and possibly augmenting them with a few linguistic rules to improve output readability (Clarke & Lapata, 2008; Nomoto, 2009).

Supervised models. A major problem for supervised deletion-based systems is very limited amount of parallel data. Many approaches make use of a small portion of the Ziff-Davis corpus which has about 1K sentence-compression pairs[1]. Other main sources of training data are the two manually crafted compression corpora from the University of Edinburgh ("written" and "spoken", each approx. 1.4K pairs). Galanis & Androutsopoulos (2011) attempt at getting more parallel data by applying a deletion-based compressor together with an automatic para-

[1]The method of Galley & McKeown (2007) could benefit from a larger number of sentences.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1481–1491,
Seattle, Washington, USA, 18-21 October 2013.

phraser and generating multiple alternative compressions. To our knowledge, this extended data set has not yet been used for successful training of compression systems.

Scarce parallel data makes it hard to go beyond a small set of features and explore lexicalization. For example, Knight & Marcu (2000) only induce non-lexicalized CFG rules, many of which occurred only once in the training data. The features of McDonald (2006) are formulated exclusively in terms of syntactic categories. Berg-Kirkpatrick et al. (2011) have as few as 13 features to decide whether a constituent can be dropped. Galanis & Androutsopoulos (2010) use many features when deciding which branches of the input dependency tree can be pruned but require a reranker to select most fluent compressions from a pool of candidates generated in the pruning phase, many of which are ungrammatical.

Even further data limitations exist for the algorithms which operate on syntactic trees and reformulate the compression task as a tree pruning one (Nomoto, 2008; Filippova & Strube, 2008; Cohn & Lapata, 2009; Galanis & Androutsopoulos, 2010, inter alia). These methods are sensitive to alignment errors, their performance degrades if the syntactic structure of the compression is very different from that of the input. For example, see Nomoto's 2009 analysis of the poor performance of the T3 system of Cohn & Lapata (2009) when retrained on a corpus of loosely similar RSS feeds and news.

Unsupervised models. Few approaches require no training data at all. The model of Hori & Furui (2004) combines scores estimated from monolingual corpora to generate compressions of transcribed speech. Adopting an integer linear programming (ILP) framework, Clarke & Lapata (2008) use hand-crafted syntactic constraints and an ngram language model, trained on uncompressed sentences, to find best compressions. The model of Filippova & Strube (2008) also uses ILP but the problem is formulated over dependencies and not ngrams. Conditional probabilities and word counts collected from a large treebank are combined in an ad hoc manner to assess grammatical importance and informativeness of dependencies. Similarly, Woodsend & Lapata (2010) formulate an ILP problem to generate news story highlights using precomputed scores. Again, an ad hoc combination of the scores learned independently of the task is used in the objective function.

Contributions of this paper. Our work is motivated by the obvious need for a large parallel corpus of sentences and compressions on which extractive systems can be trained. Furthermore, we want the compressions in the corpus to be structurally very close to the input. Ideally, in every pair, the compression should correspond to a subtree of the input. To this end, our contributions are three-fold:

- We describe an automatic procedure of constructing a parallel corpus of 250,000 sentence-compression pairs such that the dependency tree of the compression is a subtree of the source tree. An evaluation with human raters demonstrates high quality of the parallel data in terms of readability and informativeness.

- We successfully apply the acquired data to train a novel supervised compression system which produces readable and informative compressions without employing a separate reranker. In particular, we start with the unsupervised method of Filippova & Strube (2008) and replace the ad hoc edge weighting with a linear function over a rich feature representation. The parameter vector is learned from our corpus specifically for the compression task using structured prediction (Collins, 2002). The new system significantly outperforms the baseline and hence provides further evidence for the utility of the parallel data.

- We demonstrate that sparse lexical features are very useful for sentence compression, and that a large parallel corpus is a requirement for applying them successfully.

The compression framework we adopt and the unsupervised baseline are introduced in Section 2, the training algorithm for learning edge weights from parallel data is described in Section 3. In Section 4 we explain how to obtain the data and present an evaluation of its quality. In Section 5 we compare the baseline with our system and report the results of an experiment with humans as well as the results of an automatic evaluation.

2 Framework and baseline

We adopt the unsupervised compression framework of Filippova & Strube (2008) as our baseline and extend it to a supervised structured prediction problem. In the experiments reported by Filippova & Strube (2008), the system was evaluated on the Edinburgh corpora. It achieved an F-score (Riezler et al., 2003) higher than reported by other systems on the same data under an aggressive compression rate and thus presents a competitive baseline.

Tree pruning as optimization. In this framework, compressions are obtained by deleting edges of the source dependency structure so that (1) the retained edges form a valid syntactic tree, and (2) their total edge weight is maximized. The objective function is defined over set $X = \{x_e, e \in E\}$ of binary variables, corresponding to the set E of the source edges, subject to the structural and length constraints,

$$f(X) = \sum_{e \in E} x_e \times w(e) \quad (1)$$

Here, $w(e)$ denotes the weight of edge e. This constrained optimization problem is solved under the tree structure and length constraints using ILP. If x_e is resolved to 1, the respective edge is retained, otherwise it is deleted. The tree structure constraints enforce at most one parent for every node and structure connectivity (i.e., no disconnected subtrees). Given that $length(node(e))$ denotes the length of the node to which edge e points and α is the maximum permitted length for the compression, the length constraint is simply

$$\sum_{e \in E} x_e \times length(node(e)) \leq \alpha \quad (2)$$

Word limit is used in the original paper, whereas we use character length which is more appropriate for system comparisons (Napoles et al., 2011). If uniform weights are used in Eq. (1), the optimal solution would correspond to a subtree covering as many edges as possible while keeping the compression length under given limit.

The solution to the surface realization problem (Belz et al., 2011) is standard: the words in the compression subtree are put in the same order they are found in the source.

Due to space limitations, we refer the reader to (Filippova & Strube, 2008) for a detailed description on the method. Essential for the present discussion is that source dependency trees are transformed to dependency graphs in that (1) auxiliary, determiner, preposition, negation and possessive nodes are collapsed with their heads; (2) prepositions replace labels on the edges to their arguments; (3) the dummy root node is connected with every inflected verb. Figures 1(a)-1(b) illustrate most of the transformations. The transformations are deterministic and reversible, they can be implemented in a single top-down tree traversal[2].

The set E of edges in Eq. (1) is thus the set of edges of the *transformed dependency graph*, like in Fig. 1(b). A benefit of the transformations is that function words and negation appear in the compression if and only if their head words are present. Hence no separate constraints are required to ensure that negation or a determiner is preserved. The dummy root node makes constraint formulation easier and also allows for the generation of compressions from any finite clause of the source.

The described pruning optimization framework is used both for the unsupervised baseline and for our supervised system. The difference between the baseline and our system is in how edge weights, $w(e)$'s in Eq. (1), are instantiated.

Baseline edge weights. The precomputed edge weights reflect syntactic importance as well as informativeness of the nodes they point to. Given edge e from head node h to node n, the edge weight is the product of the syntactic and the informativeness weights,

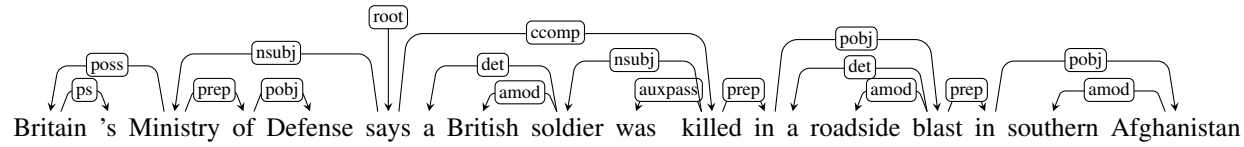
$$w(e) = w_{\text{synt}}(e) \times w_{\text{info}}(e) \quad (3)$$

The syntactic weight is defined as

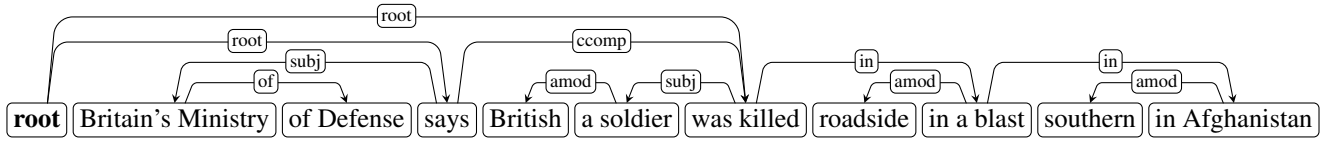
$$w_{\text{synt}}(e) = P(label(e)|lemma(h)) \quad (4)$$

For example, verb *kill* may have multiple arguments realized with dependency labels *subj*, *dobj*, *in*, etc. However, these argument labels are not equally likely, e.g., $P(\textit{subj}|\textit{kill}) > P(\textit{in}|\textit{kill})$. When forced to prune an edge, the system would prefer to keep

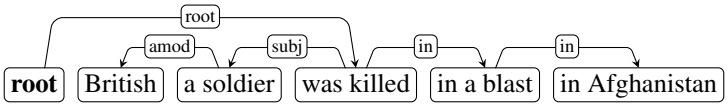
[2] Some of the transformations are comparable to what is implemented in the Stanford parser (de Marneffe et al., 2006).



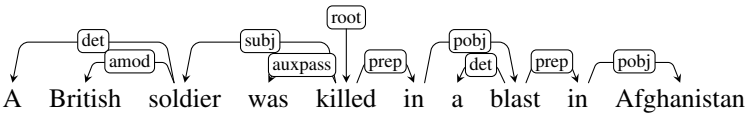
(a) Source dependency tree



(b) Transformed graph



(c) Tree of extracted headline *A British soldier was killed in a blast in Afghanistan*



(d) Tree of extracted headline with transformations undone

Figure 1: Source, transformed and extracted trees given headline *British soldier killed in Afghanistan*

the subject edge over the preposition-in edge since it contributes more weight to the objective function.

The informativeness score is inspired by Woodsend & Lapata (2012) and is defined as

$$w_{\text{info}}(e) = \frac{P_{\text{headline}}(lemma(n))}{P_{\text{article}}(lemma(n))} \quad (5)$$

This weight tells us how likely it is that a word from an article appears in the headline. For example, given two edges one of which points to verb *say* and another one to verb *kill*, the latter would be preferred over the former because *kill* is more "headliny" than *say*. When collecting counts for the syntactic and informativeness scores, we used 9M news articles crawled from the Internet, much more than Filippova & Strube (2008). As a result our estimates are probably more accurate than theirs.

Although both w_{synt} and w_{info} have a meaningful interpretation, there is no guarantee that product is the best way to combine the two when assigning edge weights. Also, it is unclear how to integrate other signals, such as distance to the root, node length or information about the siblings, which presumably all play a role in determining the overall edge importance.

3 Learning edge weights

Our supervised system differs from the unsupervised baseline in that instead of relying on precomputed scores, we define edge weight $w(e)$ in Eq. (1) with a linear function over a feature representation,

$$w(e) = \mathbf{w} \cdot \mathbf{f}(e) \quad (6)$$

Here $\mathbf{f}(e)$ is a vector of binary variables for every feature from the set of all possible but very infrequent features in the training set. $\mathbf{f}(e)$ has 1 for every feature extracted for edge *e* and zero otherwise.

Table 1 gives an overview of the feature types we use (edge *e* points from head *h* to node *n*). Note that syntactic, structural and semantic features are closed-class. For all the structural features but *char_length*, seven is used as maximum possible value; all possible character lengths are bucketed into six classes. All the features are local – for a given edge, contextual information is included about

syntactic	*label(e)*; for *e** to *h*, *label(e*)*; *pos(h)*; *pos(n)*
structural	*depth(n)*; *#children(n)*; *#children(h)*; *char_length(n)*; *#words_in(n)*
semantic	*NE_tag(h)*; *NE_tag(n)*; *is_negated(n)*
lexical	*lemma(n)*; *lemma(h)-label(e)*; for *e** to *n*'s siblings, *lemma(h)-label(e*)*

Table 1: Types of features extracted for edge *e* from *h* to *n*

the head and the target nodes, and the siblings as well as the children of the latter. The negation feature is only applicable to verb nodes which contain a negative particle, like *not*, after the tree transformations. Lexical features which combine lemmas and syntactic labels are inspired by the unsupervised baseline and are very sparse.

In what follows, our assumption is that we have a compression corpus at our disposal where for every input sentence there is a correct "oracle" compression such that its transformed parse tree matches a subtree of the transformed input graph. Given such a corpus, we can apply structured prediction methods to learn the parameter vector $\mathbf{w}$. In our study we employ an averaged variant of online structured perceptron (Collins, 2002). In the context of sentence fusion, a similar dependency structure pruning framework and a similar learning approach was adopted by Elsner & Santhanam (2011).

At every iteration, for every input graph, we find the optimal solution with ILP under the current parameter vector $\mathbf{w}$. The maximum permitted compression length is set to be the same as the length of the oracle compression. Since the oracle compression is a subtree of the input graph, it represents a feasible solution for ILP. The parameter vector is updated if there is a mismatch between the predicted and the oracle sets of edges for all the features with a non-zero net count. More formally, given an input graph with the set of edges E, oracle compression $C \subset E$ and compression $C_t \subseteq E$ predicted at iteration t , the parameter update vector at $t+1$ is given by

$$\mathbf{w_{t+1}} = \mathbf{w_t} + \sum_{e \in C \setminus C_t} \mathbf{f}(e) - \sum_{e \in C_t \setminus C} \mathbf{f}(e) \quad (7)$$

$\mathbf{w}$ is averaged over all the $\mathbf{w_t}$'s so that features whose weight fluctuated a lot during training are penalized (Freund & Shapire, 1999).

Of course, training a model with a large number of features, such as a lexicalized model, is only possible if there is a large compression corpus where the dependency tree of the compression is a subtree of the source sentence. In the next section we introduce our method of getting a sufficient amount of such data.

4 Acquiring parallel data automatically

In this section we explain how we obtained a parallel corpus of sentences and compressions. The underlying idea is to harvest news articles from the Internet where the headline appears to be similar to the first sentence and use it to find an extractive compression of the sentence.

Collecting headline-sentence pairs. Using a news crawler, we collected a corpus of news articles in English from the Internet. Similarly to previous work (Dolan et al., 2004; Wubben et al., 2009; Bejan & Harabagiu, 2010, inter alia), the Google News service[3] was used to identify news. From every article, the headline and the first sentence, which are known to be semantically similar (Dorr et al., 2003), were extracted. Predictably, very few headlines are extractive compressions of the first sentence, therefore simply looking for pairs where the headline is a subsequence of the words from the first sentence would not solve the problem of getting a large amount of parallel data. Importantly, headlines are syntactically quite different from "normal" sentences. For example, they may have no main verb, omit determiners and appear incomplete, making it hard for a supervised deletion-based system to learn useful rules. Moreover, we observed poor parsing accuracy for headlines which would make syntactic annotations for headlines hardly useful.

Thus, instead of taking the headline as it is, we use it to find a proper extractive compression of the sen-

[3] `http://news.google.com`, Jan-Dec 2012.

tence by matching lemmas of content words (*nouns, verbs, adjectives, adverbs*) and coreference IDs of entities from the headline with those of the sentence. The exact procedure is as follows (H, S and T stand for *headline*, *sentence* and *transformed graph* of the sentence):

PREPROCESSING H and S are preprocessed in a standard way: tokenized, lemmatized, PoS and NE tagged. Additionally, S is parsed with a dependency parser (Nivre, 2006) and transformed as described in Section 2 to obtain T. Finally, pronominal anaphora is resolved in S. Recall that S is the first sentence, so the antecedent must be located in a preceding, higher-level clause.

FILTERING To restrict the corpus to grammatical and informative headlines, we implemented a cascade of filters. Pair (H, S) is discarded if any of the questions in Table 2 is answered positively.

Is H a question?
Is H or S too short? (less than four word tokens)
Is H about as long as S? (min ratio: 1.5)
Does H lack a verb?
Does H begin with a verb?
Is there a *noun, verb, adj, adv* lemma from H not found in S?
Are the *noun, verb, adj, adv* lemmas from H found in S in a different order?

Table 2: Filters applied to candidate pair (H, S)

MATCHING Given the content words of H, a subset of nodes in T is selected based on lemma or coreference identity of the main (head) word in the nodes. For example, the main word of a collapsed node in T, which covers two words *was killed*, is *killed*; *was* is its child attached with label *aux* in the untransformed parse tree. This node is marked if H contains word *killed* or *killing* because of the lemma identity. In some cases there are multiple possible matches. For example, given S *Barack Obama said he will attend G20* and H mentioning *Obama*, both *Barack Obama* and *he* nodes are marked in T. Once all the nodes in T which match content words and entities from H are identified, a minimum subtree covering these nodes is found such that every word or entity from H occurs as many times in T as in H. So if *H* mentions *Obama* only once, then either *Barack Obama* or *he* must be covered by the subtree but not both. This minimum subtree corresponds to an **extractive headline**, H⋆, which we generate by ordering the surface forms of all the words in the subtree nodes by their offsets in S. Finally, the character length of H⋆ is compared with the length of H. If H⋆ is much longer than H, the pair (H, S) is discarded (max ratio 1.5).

As an illustration to the procedure, consider the example from Figure 1 with the extracted headline and its tree presented in Figure 1(c). Given the headline *British soldier killed in Afghanistan*, the extracted headline would be *A British soldier was killed in a blast in Afghanistan*. The lemmas *british, soldier, kill, afghanistan* from the headline match the nodes *British, a soldier, was killed, in Afghanistan* in the transformed graph. The node *in a blast* is added because it is on the path from *was killed* to *in Afghanistan*. Of course, it is possible to deterministically undo the transformations in order to obtain a standard dependency tree. In this case the extracted headline would still correspond to a subtree of the input (compare Fig. 1(d) with Fig. 1(a)). Also note that a similar procedure can be implemented for constituency parses.

The resulting corpus consists of 250K tuples (S, T, H, H⋆), Appendix provides more examples of source sentences, original headlines and extracted headlines. We did not attempt to tune the values for minimum/maximum length and ratio – lower thresholds may have produced comparable results.

Evaluating data quality. The described procedure produces a comparatively large compression corpus but how good are automatically constructed compressions? To answer this question, we randomly selected 50 tuples from the corpus and set up an experiment with human raters to validate and assess data quality in terms of *readability*[4] and *informativeness*[5] which are standard measures of compression quality (Clarke & Lapata, 2006). Raters were asked to read a sentence and a compression (original H or extracted H⋆ headline) and then rate the compression on two five-point scales. Three ratings were collected for every item. Table 3 gives

[4] Also called *grammaticality* and *fluency*.

[5] Also called *importance* and *representativeness*.

average ratings with standard deviation.

	AVG read	*AVG info*
ORIG. HEADLINE	4.36 (0.75)	3.86 (0.79)
EXTR. HEADLINE	4.26 (1.01)	3.70 (1.04)

Table 3: Results for two kinds of headlines

In terms of readability and informativeness the extracted headlines are comparable with human-written ones: at 95% confidence there is no statistically significant difference between the two.

Encouraged by the results of the validation experiment we proceeded to our next question: Can a supervised compression system be successfully trained on this corpus?

5 System evaluation and discussion

From the corpus of 250K tuples we used 100K to get pairs of *extracted* headlines and sentences for training (on the development set we did not observe much improvement from using more training data), 250 for development and the rest for testing. We ran the learning algorithm for 20 iterations, checking the performance on the development set. Features which applied to less than 20 edges were pruned, the size of the feature set is about 28K.

5.1 Evaluation with humans

50 pairs of *original* headlines and sentences (different from the data validation set in Sec. 4) were randomly selected for an evaluation with humans from the test data. As in the data quality validation experiment, we asked raters to assess the readability and informativeness of proposed compressions for the unsupervised system, our system and human-written headlines. The latter provide us with upper bounds on the evaluation criteria. Three ratings per item per parameter were collected. To get comparable results, the unsupervised and our systems used the same compression rate: for both, the requested maximum length was set to the length of the headline. Table 4 summarizes the results.

The results indicate that the trained model significantly outperforms the unsupervised system, getting particularly good marks for readability. The difference in readability between our system and *original* headlines is not statistically significant. Note that

	AVG read	*AVG info*
ORIG. HEADLINE	$4.66^{\dagger}$	$4.10^{\dagger\ddagger}$
OUR SYSTEM	$4.30^{\dagger}$	$3.52^{\dagger}$
UNSUP. SYSTEM	3.70	2.70

Table 4: Results for the systems and original headline: † and ‡ stand for *significantly better than Unsupervised* and *Our system at 95% confidence*, respectively

the unsupervised baseline is also capable of generating readable compressions but does a much poorer job in selecting most important information. Our trained model successfully learned to optimize both scores. We refer the reader to Appendix for input and compression examples. Note that the ratings for the human-written headlines in this experiment are slightly different from the ratings in the data validation experiment because a different data sample was used.

5.2 Automatic evaluation

Our automatic evaluation had the goal of explicitly addressing two relevant questions related to our claims about (1) the benefits of having a large parallel corpus and (2) employing a supervised approach with a rich feature representation.

1. Our primary motivation for collecting parallel data has been that having access to sparse lexical features, which considerably increase the feature space, would benefit compression systems. But is it really the case for sentence compression? Can a comparable performance be achieved with a closed, moderately sized set of dense, non-lexical features? If yes, then a large compression corpus is probably not needed. Furthermore, to demonstrate that a large corpus is not only sufficient but also necessary to learn weights for thousands of features, we need to compare the performance of the system when trained on the full data set and a small portion of it.

2. The syntactic and informativeness scores in Eq. (3) were calculated over millions of news articles and do provide us with meaninful statistics (see Sec. 2). Is there any benefit in replacing those scores with weights learned for

their feature counterparts? Recall that one of our feature types in Table 1 is the concatenation of *lemma(h)* (parent lemma) and *label(e)* which relies on the same information as $w_{\text{synt}} = P(label(e)|lemma(h))$. The feature counterpart of w_{info} defined in Eq. (5) is *lemma(n)*–the lemma of the node to which edge points. How would the supervised system perform against the unsupervised one, if it only extracted features of these two types?

To answer these questions, we sampled 1,000 tuples from the unused test data and measured F1 score (Riezler et al., 2003) by comparing the trees of the generated compression and the "correct", extracted headline. The systems we compared are the unsupervised baseline (UNSUP. SYSTEM) and the supervised model trained on three kinds of feature sets: (1) SYNT-INFO FEATURES, corresponding to the supervised training of the unsupervised baseline model (i.e., *lemma(h)-label(e)* and *lemma(n)*); (2) NON-LEX FEATURES, corresponding to a dense, non-lexical feature representation (i.e., all the feature types from Table 1 excluding the three involving *lemma*s); (3) ALL FEATURES (same as OUR SYSTEM). Additionally, we trained the system on 10% of the data–10K as opposed to 100K tuples, ALL FEATURES (10K)–for 20 iterations ignoring features which applied to less than three edges[6]. As before, the same compression rate was used for all the systems. The results are summarized in Table 5.

	F1 score	*#features*
UNSUP. SYSTEM	52.3	N.A.
SYNT-INFO FEATURES	75.0	12,490
NON-LEX FEATURES	79.6	330
ALL FEATURES	84.3	27,813
ALL FEATURES (10K)	81.4	22,529

Table 5: Results for the unsupervised baseline and the supervised system trained on three kinds of feature sets

Clearly, having more features, lexicalized and unlexicalized, is important: there is a significant improvement in going beyond the closed set of 330 non-lexical features to all, from 79.6 to 84.3 points. Moreover, successful training requires a large corpus since the performance of the system degrades if only 10K training instances are used. Note that this number already exceeds all the existing compression corpora taken together. Hence, sparse lexical features are useful for compression and a large parallel corpus is a requirement for successful supervised training.

Concerning our second question, learning feature weights from the data produces significantly better results than the hand-crafted way of making use of the same information, even if a much larger data set is used to collect statistics. We observed a dramatic increase from 52.3 to 75.0 points. Thus, we may conclude that training with dense and sparse features directly from data definitely improves the performance of the dependency pruning system.

5.3 Discussion

It is important to note that the data we used is challenging: first sentences in news articles tend to be long, in fact longer than other news sentences, which implies less reliable syntactic analysis and noisier input to the syntax-based systems. In the test set we used for the evaluation with humans, the mean sentence length is 165 characters. The average compression rate in characters is 0.46 ± 0.16 which is quite aggressive[7]. Recall that we used the very same framework for the unsupervised baseline and our system as well as the same compression rate. All the preprocessing errors affect both systems equally and the comparison of the two is fair. Predictably, wrong syntactic parses significantly increase chances of an ungrammatical compression, and parser errors seem to be a major source of readability deficiencies.

A property of the described compression framework is that a desired compression length is expected to be provided by the user. This can be seen both as a strength and as a weakness, depending on the application. In a scenario where mobile devices with a limited screen size are used, or in a summarization scenario where a total summary length is provided (see the DUC/TAC guidelines[8]), being able

[6]Recall from the beginning of the section that for the full (100K) training set the threshold was set to 20 with no tuning. For the 10K training set, we tried values of two, three, five and varied the number of iterations. The result we report is the highest we could get for 10K.

[7]We follow the standard terminology where smaller values imply *shorter* compressions.

[8]http://www.nist.gov/tac/

to specify a length is definitely an advantage. However, one can also think of other applications where the user does not have a strict length constraint but wants the text to be somewhat shorter. In this case, a reranker which compares compressions generated for a range of possible lengths can be employed to find a single compression (e.g., mean edge weight in the solution or a language model-based score).

6 Conclusions

We have addressed a major problem for supervised extractive compression models – the lack of a large parallel corpus. To this end, we presented a method to automatically build such a corpus from web documents available on the Internet. An evaluation with humans demonstrates that the quality of the corpus is high – the compressions are grammatical and informative. We also significantly improved a competitive unsupervised method achieving high readability and informativeness scores by incorporating thousands of features and learning the feature weights from our corpus. This result further confirms the practical utility of the automatically obtained data. We have shown that employing lexical features is important for sentence compression, and that our supervised module can successfully learn their weights from the corpus. To our knowledge, we are the first to empirically demonstrate that sparse features are useful for compression and that a large parallel corpus is a requirement for a successful learning of their weights. We believe that other supervised deletion-based systems can benefit from our work.

Acknowledgements: The authors are thankful to the EMNLP reviewers for their feedback and suggestions.

Appendix

The appendix presents examples of source sentences (S), original headlines (H), extracted headlines (H*), unsupervised baseline (U) and our system (O) compressions.

References

Bejan, C. & S. Harabagiu (2010). Unsupervised event coreference resolution with rich linguistic features. In *Proc. of ACL-10*, pp. 1412–1422.

Belz, A., M. White, D. Espinosa, E. Kow, D. Hogan & A. Stent (2011). The first surface realization shared task: Overview and evaluation results. In *Proc. of ENLG-11*, pp. 217–226.

Berg-Kirkpatrick, T., D. Gillick & D. Klein (2011). Jointly learning to extract and compress. In *Proc. of ACL-11*.

Clarke, J. & M. Lapata (2006). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proc. of COLING-ACL-06*, pp. 377–385.

Clarke, J. & M. Lapata (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Cohn, T. & M. Lapata (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

Collins, M. (2002). Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP-02*, pp. 1–8.

de Marneffe, M.-C., B. MacCartney & C. D. Manning (2006). Generating typed dependency parses from phrase structure parses. In *Proc. of LREC-06*, pp. 449–454.

Dolan, B., C. Quirk & C. Brokett (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics,* Geneva, Switzerland, 23–27 August 2004, pp. 350–356.

Dorr, B., D. Zajic & R. Schwartz (2003). Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the Text Summarization Workshop at HLT-NAACL-03,* Edmonton, Alberta, Canada, 2003, pp. 1–8.

S	Country star Sara Evans has married former University of Alabama quarterback Jay Barker.
H	Country star Sara Evans marries
H*	Country star Sara Evans has married
U	Sara Evans has married Jay Barker
O	Sara Evans has married Jay Barker
S	Intel would be building car batteries, expanding its business beyond its core strength, the company said in a statement
H	Intel to build car batteries
H*	Intel would be building car batteries
U	would be building the company said
O	Intel would be building car batteries
S	A New Orleans Saints team spokesman says tight end Jeremy Shockey was taken to a hospital but is doing fine.
H	Spokesman: Shockey taken to hospital, doing fine
H*	spokesman says Jeremy Shockey was taken to a hospital but is doing fine
U	A New Orleans Saints team spokesman says Jeremy Shockey was taken
O	tight end Jeremy Shockey was taken to a hospital but is doing fine
S	President Obama declared a major disaster exists in the State of Florida and ordered Federal aid to supplement State and local recovery efforts in the area struck by severe storms, flooding, tornadoes, and straight-line winds beginning on May 17, 2009, and continuing.
H	President Obama declares major disaster exists in the State of Florida
H*	President Obama declared a major disaster exists in the State of Florida
U	President Obama declared a major disaster exists and ordered Federal aid
O	President Obama declared a major disaster exists in the State of Florida
S	Regulators Friday shut down a small Florida bank, bringing to 119 the number of US bank failures this year amid mounting loan defaults.
H	Regulators shut down small Florida bank
H*	Regulators shut down a small Florida bank
U	shut down bringing the number of failures
O	Regulators shut down a small Florida bank
S	Three men were arrested Wednesday night and Dayton police said their arrests are in connection to a west Dayton bank robbery.
H	3 men arrested in connection with Bank robbery
H*	Three men were arrested are in connection to a bank robbery
U	were arrested and Dayton police said their arrests are
O	Three men were arrested and police said their arrests are
S	The government and the social partners will resume the talks on the introduction of the so-called crisis tax, which will be levied on all salaries, pensions and incomes over HRK 3,000.
H	Government, social partners to resume talks on introduction of “crisis” tax.
H*	The government and the social partners will resume the talks on the introduction of the crisis tax
U	The government will resume the talks on the introduction of the crisis tax which will be levied
O	The government and the social partners will resume the talks on the introduction of the crisis tax
S	England star David Beckham may have the chance to return to AC Milan after the Italian club’s coach said he was open to his move on Sunday.
H	Beckham has chance of returning to Milan
H*	David Beckham may have the chance to return to AC Milan
U	David Beckham may have the chance to return said star was
O	David Beckham may have the chance to return to AC Milan
S	Eastern Health and its insurance company have accepted liability for some patients involved in the breast cancer testing scandal, according to a statement released Friday afternoon.
H	Eastern Health accepts liability for some patients
H*	Eastern Health have accepted liability for some patients
U	Health have accepted liability according to a statement
O	Eastern Health have accepted liability for some patients
S	Frontier Communications Corp., a provider of phone, TV and Internet services, said Thursday it has started a cash tender offer to purchase up to $700 million of its notes.
H	Frontier Communications starts tender offer for up to $700 million of notes
H*	Frontier Communications has started a tender offer to purchase $700 million of its notes
U	Frontier Communications said Thursday a provider has started a tender offer
O	Frontier Communications has started a tender offer to purchase $700 million of its notes

Elsner, M. & D. Santhanam (2011). Learning to fuse disparate sentences. In *Proceedings of the Workshop on Monolingual Text-to-text Generation,* Prtland, OR, June 24 2011, pp. 54–63.

Filippova, K. & M. Strube (2008). Dependency tree based sentence compression. In *Proc. of INLG-08*, pp. 25–32.

Freund, Y. & R. E. Shapire (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

Galanis, D. & I. Androutsopoulos (2010). An extractive supervised two-stage method for sentence compression. In *Proc. of NAACL-HLT-10*, pp. 885–893.

Galanis, D. & I. Androutsopoulos (2011). A new sentence compression dataset and its use in an abstractive generate-and-rank sentence compressor. In *Proc. of UCNLG+Eval-11*, pp. 1–11.

Galley, M. & K. R. McKeown (2007). Lexicalized Markov grammars for sentence compression. In *Proc. of NAACL-HLT-07*, pp. 180–187.

Gillick, D. & B. Favre (2009). A scalable global model for summarization. In *ILP for NLP-09*, pp. 10–18.

Grefenstette, G. (1998). Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working Notes of the Workshop on Intelligent Text Summarization,* Palo Alto, Cal., 23 March 1998, pp. 111–117.

Hori, C. & S. Furui (2004). Speech summarization: An approach through word extraction and a method for evaluation. *IEEE Transactions on Information and Systems*, E87-D(1):15–25.

Jing, H. & K. McKeown (2000). Cut and paste based text summarization. In *Proc. of NAACL-00*, pp. 178–185.

Knight, K. & D. Marcu (2000). Statistics-based summarization – step one: Sentence compression. In *Proc. of AAAI-00*, pp. 703–711.

Mani, I. (2001). *Automatic Summarization.* Amsterdam, Philadelphia: John Benjamins.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic evidence. In *Proc. of EACL-06*, pp. 297–304.

Napoles, C., C. Callison-Burch, J. Ganitkevitch & B. Van Durme (2011). Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-to-text Generation,* Prtland, OR, June 24 2011, pp. 84–90.

Nivre, J. (2006). *Inductive Dependency Parsing.* Springer.

Nomoto, T. (2008). A generic sentence trimmer with CRFs. In *Proc. of ACL-HLT-08*, pp. 299–307.

Nomoto, T. (2009). A comparison of model free versus model intensive approaches to sentence compression. In *Proc. of EMNLP-09*, pp. 391–399.

Riezler, S., T. H. King, R. Crouch & A. Zaenen (2003). Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. In *Proc. of HLT-NAACL-03*, pp. 118–125.

Turner, J. & E. Charniak (2005). Supervised and unsupervised learning for sentence compression. In *Proc. of ACL-05*, pp. 290–297.

Woodsend, K. & M. Lapata (2010). Automatic generation of story highlights. In *Proc. of ACL-10*, pp. 565–574.

Woodsend, K. & M. Lapata (2012). Multiple aspect summarization using Integer Linear Programming. In *Proc. of EMNLP-12*, pp. 233–243.

Wubben, S., A. van den Bosch, E. Krahmer & E. Marsi (2009). Clustering and matching headlines for automatic paraphrase acquisition. In *Proc. of ENLG-09*, pp. 122–125.

Zajic, D., B. J. Dorr, J. Lin & R. Schwartz (2007). Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing & Management, Special Issue on Text Summarization*, 43(6):1549–1570.

Fast Joint Compression and Summarization via Graph Cuts

Xian Qian and **Yang Liu**
The University of Texas at Dallas
800 W. Campbell Rd., Richardson, TX, USA
{qx,yangl}@hlt.utdallas.edu

Abstract

Extractive summarization typically uses sentences as summarization units. In contrast, joint compression and summarization can use smaller units such as words and phrases, resulting in summaries containing more information. The goal of compressive summarization is to find a subset of words that maximize the total score of concepts and cutting dependency arcs under the grammar constraints and summary length constraint. We propose an efficient decoding algorithm for fast compressive summarization using graph cuts. Our approach first relaxes the length constraint using Lagrangian relaxation. Then we propose to bound the relaxed objective function by the supermodular binary quadratic programming problem, which can be solved efficiently using graph max-flow/min-cut. Since finding the tightest lower bound suffers from local optimality, we use convex relaxation for initialization. Experimental results on TAC2008 dataset demonstrate our method achieves competitive ROUGE score and has good readability, while is much faster than the integer linear programming (ILP) method.

1 Introduction

Automatic multi-document summarization helps readers get the most important information from large amounts of texts. Summarization techniques can be roughly divided into two categories: extractive and abstractive. Extractive summarization casts the summarization task as a sentence selection problem: identifying important summary sentences from one or multiple documents. Many methods have been developed in the past decades, including supervised approaches that use classifiers to predict summary sentences, graph based approaches to rank the sentences, and recent global optimization methods such as integer linear programming (Gillick et al., 2008) (ILP) and submodular maximization methods (Lin and Bilmes, 2011). Though extractive summarization is popular because of its simplicity and high readability, it has limitations in that it selects each sentence as a whole, and thus may miss informative partial sentences.

To improve the informativeness, joint compression and summarization was proposed (Berg-Kirkpatrick et al., 2011), which uses words as summarization units, unlike extractive summarization where each sentence is a basic undecomposable unit. To achieve better readability, manually defined grammar constraints or automatically learned models based on syntax trees are added during the summarization process. Up to now, the state of the art compressive systems are based on integer linear programming (ILP). Because ILP suffers from exponential complexity, word-based compression summarization is an order of magnitude slower than sentence-based extraction.

One common way to solve an ILP problem is to use its LP relaxation and round the results. However Berg-Kirkpatrick et al. (2011) found that LP relaxation gave poor results, finding unacceptably suboptimal solutions. For speedup, they proposed a two stage method where they performed some sentence selection in the first step to reduce the number of candidates. Despite their empirical success, such

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1492–1502,
Seattle, Washington, USA, 18-21 October 2013.

a pruning approach has its inherent problem in that it may eliminate correct sentences in the first step. Recently, Almeida and Martins (2013) proposed a fast joint decoding algorithm based on dual decomposition. For fast convergence, they added quadratic penalty terms to alleviate the learning rate problem.

In this paper, we propose an efficient decoding algorithm for fast ILP based compressive summarization using graph cuts. Our assumption is that all concepts are word n-grams and non-negatively scored. The rationale for the non-negativity assumption is straightforward: the score of a concept reflects its informativeness, hence should be non-negative. Given a set of documents, each word is associated with a binary variable, indicating whether the word is selected in the summary. Our idea is to approximate the ILP as a binary quadratic programming problem where coefficients of all quadratic terms are non-negative. It is well known that such binary quadratic function is supermodular, and its maximum can be solved efficiently using graph max-flow/min-cut. Hence the key is to find the coefficients of the supermodular binary quadratic function (SBQF) so that its maximum is close to the optimal ILP objective function. Our solution consists of 3 steps. First, we show that the subtree deletion model and grammar constraints can be eliminated by adding SBQFs to the objective function. Second, we relax the summary length constraint using Lagrangian relaxation. Third, we propose a family of SBQFs that are lower bounds of the ILP objective function. Since finding the tightest lower bound suffers from local optimality, we choose to use convex relaxation for initialization. To demonstrate our technique, we conduct experiments on Text Analysis Conference (TAC) datasets using the same train/test splits as previous work (Berg-Kirkpatrick et al., 2011). We compare our approach with the state-of-the-art ILP based approach in terms of summary quality (ROUGE scores and sentence quality) and speed. Experimental results show that our proposed method achieves competitive performance with ILP, while about 100 times faster.

2 Compressive Summarization

2.1 Extractive Summarization

As our method is an approximation of ILP based method, we first briefly review the ILP based extractive summarization and compressive summarization. Gillick and Favre (2009) introduced the concept-based ILP for summarization. A concept is a basic semantic unit. They used word bigrams as such language concepts. Their system achieved the highest ROUGE score on the TAC 2009 evaluation. This approach selects sentences so that the total score of language concepts appearing in the summary is maximized. The association between the language concepts and sentences serves as the constraints, in addition to the summary length constraint.

Formally, given a set of sentences $\mathcal{S} = \{s_n\}_{n=1}^{N}$, extractive summarization can be represented by a binary vector $\mathbf{y}$, where y_n indicates whether sentence s_n is selected. Let $\mathcal{C} = \{c_1, \dots c_J\}$ denote the set of concepts in $\mathcal{S}$, e.g., word bigrams (Gillick and Favre, 2009). Each concept c_j is associated with a given score w_j and a binary variable v_j indicating if c_j is selected in the summary. Let n_{jk} denote the index of the sentence containing the k^{th} occurrence of concept c_j, and l_n denote the length of sentence s_n. The ILP based extractive summarization system can be formulated as below:

$$\begin{aligned} \max_{\mathbf{y},\mathbf{v}} \quad & \sum_{j=1}^{J} w_j v_j \\ \text{s.t.} \quad & v_j = \bigcup_{k} y_{n_{jk}} \qquad 1 \le j \le J \qquad (1) \\ & \sum_{i=1}^{N} y_n l_n \le L \\ & \mathbf{v}, \mathbf{y} \text{ are binary} \end{aligned}$$

The first constraint is imposed by the relation between concept selection and sentence selection: selecting a sentence leads to the selection of all the concepts it contains, and selecting a concept only happens when it is present in at least one of the selected sentences. The second constraint is the summary length constraint.

As solving an ILP problem is generally NP-hard, pre-pruning of candidate concepts and sentences is necessary for efficient summarization. For exam-

ple, the ICSI system (Gillick et al., 2008) removed the sentences that are too short or have non-overlap with the queries, and concepts with document frequency less than 3, resulting in 95.8 sentences and about 80 concepts per topic on the TAC2009 dataset. Therefore the actual scale of ILP is rather small after pruning (e.g., 176 variables and 372 constraints per topic). Empirical studies showed that such small scale ILP can be solved within a few seconds (Gillick and Favre, 2009).

2.2 Compressive Summarization

The quality of sentence-based extractive summarization is limited by the informativeness of the original sentences and the summary length constraint. To remove the unimportant part from a long sentence, sentence compression is proposed to generate more informative summaries (Liu and Liu, 2009; Li et al., 2013a). Recent studies show that joint sentence compression and extraction, namely compressive summarization, outperforms pipeline systems that run extractive summarization on the compressed sentences or compress selected summary sentences (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Chali and Hasan, 2012). In Berg-Kirkpatrick et al. (2011), compressive summarization integrates the concept model for extractive summarization (Gillick and Favre, 2009) and subtree deletion model for sentence compression. The score of a compressive summary consists of two parts, scores of selected concepts, and scores of the broken arcs in the dependency parse trees. The selected words must satisfy the length constraint and grammar constraints that include subtree constraint and some manually defined hard constraints.

Formally, let $\mathbf{x} = x_1 \dots x_I$ denote the word sequence of documents, where $s_1 = x_1, \dots x_{l_1}$ corresponds to the first sentence, $s_2 = x_{l_1+1}, \dots, x_{l_1+l_2}$ corresponds to the second sentence, and so on. A compressive summary can be represented by a binary vector $\mathbf{z}$, where z_i indicates whether word x_i is selected in the summary. Let a_{hm} denote the arc $x_h \rightarrow x_m$ in the dependency parse tree of the corresponding sentence containing words x_h and x_m, and $\mathcal{A} = \{a_{hm}\}$ denote the set of dependency arcs. The subtree constraint ensures that word x_m is selected only if its head x_h is selected. In order to guarantee the readability, grammar constraints are added to prohibit the breaks of some specific arcs. For example, Clarke and Lapata (2008) never deleted an arc whose dependency label is *SUB, OBJ, PMOD, SBAR* or *VC*. In this paper, we use $\mathcal{B} \subseteq \mathcal{A}$ to denote the set of these arcs that must not be broken in summarization. We use o_{jk} to denote the indices of words corresponding to the k^{th} occurrence of c_j. For example, suppose the j^{th} concept *European Union* appears twice in the document: $x_{22}x_{23} = x_{50}x_{51}$ =*European Union*, then $o_{j1} = \{22, 23\}, o_{j2} = \{50, 51\}$.

The compressive summarization model can be formulated as an integer programming problem

$$
\begin{aligned}
\max_{\mathbf{z},\mathbf{v}} \quad & \sum_{j=1}^{J} w_j \cdot v_j + \sum_{a_{hm} \in \mathcal{A}} w_{a_{hm}} z_h (1 - z_m) \\
\text{s.t.} \quad & v_j = \bigcup_k \prod_{i \in o_{jk}} z_i \qquad \forall j \\
& \sum_i z_i \leq L \\
& z_h \geq z_m \quad \forall a_{hm} \in \mathcal{A} \\
& z_h = z_m \quad \forall a_{hm} \in \mathcal{B} \\
& \mathbf{z}, \mathbf{v} \text{ are binary}
\end{aligned} \tag{2}
$$

According to the subtree deletion model, the score of arc a_{hm} is included if $z_h = 1$ and $z_m = 0$, which can be formulated as $w_{a_{hm}} \cdot z_h(1 - z_m)$. The first constraint is similar to that in extractive summarization, that is, a concept is selected if and only if any of its occurrence is selected. The third and fourth constraints are the subtree constraints and manually defined grammar constraints respectively. In the rest of the paper, without loss of generality, we remove the fourth constraint by directly substituting one variable for the other.

Finding the optimal summary is generally NP-hard. Unlike extractive summarization where the scale of the problem (the number of sentences and concepts) is small, the number of variables in compressive summarization is linear in the number of words, which is usually thousands on the TAC datasets. Hence solving such a problem using ILP based decoding algorithms is not efficient especially when the document set is large.

3 Fast Decoding via Graph Cuts

In this section, we introduce our fast decoding algorithm. We assume that all the concepts are word n-grams, and their scores are non-negative. The non-negativity assumption can reduce the computational complexity, but is also reasonable: the score of a concept denotes its informativeness, hence should be non-negative. For example, Li et al. (2013b) proposed to use the estimated normalized frequencies of concepts as scores, which are essentially non-negative. The basic idea of our method is to approximate the above optimization problem (2) by the supermodular binary quadratic programming (SBQP) problem:

$$\begin{aligned} \max_{\mathbf{z}} \quad & \sum_i \beta_i z_i + \sum_{ij} \alpha_{ij} z_i z_j \\ \text{s.t.} \quad & \mathbf{z} \text{ is binary} \end{aligned} \tag{3}$$

where $\alpha_{ij} \geq 0$. It is known that such a binary quadratic function is supermodular, and its maximum can be solved efficiently using graph max-flow/min-cut (Billionnet and Minoux, 1985; Kolmogorov and Zabih, 2004). Now the problem is to find the optimal α, β for a good approximation.

3.1 Formulate Grammar Constraints and Subtree Deletion Model by SBQF

We show that the subtree deletion model can be formulated equivalently using SBQF. First, we can eliminate the constraint $z_h \geq z_m$ by adding a penalty term to the objective function. That is,

$$\begin{aligned} \max \quad & f(\mathbf{z}) \\ \text{s.t.} \quad & z_h \geq z_m \\ & \mathbf{z} \text{ is binary} \end{aligned}$$

is equivalent to

$$\begin{aligned} \max \quad & f(\mathbf{z}) - \infty(1 - z_h)z_m \\ \text{s.t.} \quad & \mathbf{z} \text{ is binary} \end{aligned}$$

We can see that the penalty term $-\infty(1-z_h)z_m$ excludes $z_h = 0$, $z_m = 1$ from the feasible set, and for $z_h \geq z_m$, both problems have the same objective function value. Hence the two problems are equivalent. Notice that the coefficient of quadratic term in $-\infty(1 - z_h)z_m$ is positive, hence the penalty term is supermodular.

Now we eliminate the third constraint in problem (2) using the penalized objective function described above. Note that the fourth constraint has been eliminated by variable substitution, we have

$$\begin{aligned} \max_{\mathbf{z},\mathbf{v}} \quad & \sum_{j=1}^{J} w_j \cdot v_j + \sum_{a_{hm} \in \mathcal{A}} w_{a_{hm}} z_h (1 - z_m) \\ & -\infty \sum_{a_{hm} \in \mathcal{A}} (1 - z_h) z_m \\ \text{s.t.} \quad & v_j = \bigcup_k \prod_{i \in o_{jk}} z_i \qquad \forall j \\ & \sum_i z_i \leq L \\ & \mathbf{z}, \mathbf{v} \text{ are binary} \end{aligned} \tag{4}$$

We can see that for each arc a_{hm}, there must be a positive quadratic term $+\infty z_h z_m$ in the objective function, which guarantees the supermodularity of the objective function, no matter what $w_{a_{hm}}$ is.

3.2 Eliminate Length Constraint Using Lagrangian Relaxation

Problem (4) is NP-hard, because for any feasible $\mathbf{v}$, it is a SBQP with a length constraint. Since size constrained minimum cut problem is generally NP-hard (Nagano et al., 2011), Problem (4) can not be cast as a SBQP as long as P $\neq$ NP. One popular way to deal with the size constrained optimization problem is Lagrangian relaxation. We introduce Lagrangian multiplier λ to the length constraint in Problem (4), and get

$$\begin{aligned} \min_{\lambda} \max_{\mathbf{z},\mathbf{v}} \quad & \sum_{j=1}^{J} w_j \cdot v_j + \sum_{a_{hm} \in \mathcal{A}} w_{a_{hm}} z_h (1 - z_m) \\ & -\infty \sum_{a_{hm} \in \mathcal{A}} (1 - z_h) z_m \\ & +\lambda (L - \sum_i z_i) \\ \text{s.t.} \quad & v_j = \bigcup_k \prod_{i \in o_{jk}} z_i \qquad \forall j \\ & \lambda \geq 0 \\ & \mathbf{z}, \mathbf{v} \text{ are binary} \end{aligned} \tag{5}$$

We solve the relaxed problem iteratively. In each iteration, we fix λ and solve the inner maximization problem (details described below). The score of

each word is penalized by λ — with larger λ, fewer words are selected. Hence the summary length can be adjusted by λ. The optimal λ can be found using binary search. We maintain an upper bound λ_{max} , and a lower bound λ_{min}, which is initially 0. In each iteration, we choose $\lambda = \frac{1}{2}(\lambda_{max} + \lambda_{min})$ and search the optimal $\mathbf{z}$. If the duality gap vanishes, i.e., $\lambda(L - \sum_i z_i) = 0$ and $\sum_i z_i \leq L$, then we get the global solution of Problem (4). Otherwise, if $\sum_i z_i > L$, then the current λ is too small, so we set $\lambda_{min} = \lambda$; otherwise, $\lambda > 0$ and $\sum_i z_i < L$, we set $\lambda_{max} = \lambda$. The search process terminates if $\lambda_{max} - \lambda_{min}$ is less than a predefined threshold.

3.3 Eliminate v Using Supermodular Relaxation

Now we consider the inner maximization Problem (5). It is still not a SBQP, since the objective function is not a linear function of $z_i z_j$. We propose to approximate the objective function using SBQP. Our solution consists of two steps. First we relax the first constraint of Problem (5) by bounding the objective function with a family of supermodular pseudo boolean functions (Boros and Hammer, 2002). Second we reformulate these pseudo boolean functions equivalently as quadratic functions.

Similar to the bounding strategy in (Qian and Liu, 2013), we relax the logical disjunction by linearization. Using the fact that for any binary vector $\mathbf{a}$, we have

$$\bigcup a_i = \max_{\mathbf{p}\in\Delta} \sum_i p_i a_i$$

where Δ denotes the probability simplex

$$\Delta = \{\mathbf{p} | \sum_k p_k = 1, p_k \geq 0\}$$

We have

$$\begin{aligned} v_j &= \bigcup_k \prod_{i\in o_{jk}} z_i \\ &= \max_{\mathbf{p}_j\in\Delta} \sum_k p_{jk} \prod_{i\in o_{jk}} z_i \end{aligned}$$

Plug the equation above into the objective function of Problem (5), we get the following optimization problem

$$\begin{aligned} \max_{\mathbf{z},\mathbf{p}} \quad & \sum_{j=1}^{J} \left(\sum_k p_{jk} w_j \prod_{i\in o_{jk}} z_i \right) \\ & + \sum_{a_{hm}\in\mathcal{A}} w_{a_{hm}} z_h (1 - z_m) \\ & -\infty \sum_{a_{hm}\in\mathcal{A}} (1 - z_h) z_m \\ & +\lambda(L - \sum_i z_i) \\ \text{s.t.} \quad & \mathbf{z} \text{ is binary} \\ & \mathbf{p}_j \in \Delta \qquad \forall j \end{aligned} \tag{6}$$

Let $Q(\mathbf{p}, \mathbf{z})$ denote the objective function of Problem (6). Given $\mathbf{p}$, we can see that Q is a supermodular pseudo boolean function because coefficients of all non-linear terms are non-negative. Using the fact that for any binary vector $\mathbf{a} = [a_1, \dots a_r]^T$, $a_i \in \{0, 1\}, 1 \leq i \leq r$,

$$\prod_{i=1}^{r} a_i = \max_{b\in\{0,1\}} \left(\sum_{i=1}^{r} a_i - r + 1 \right) b$$

(Freedman and Drineas, 2005), we get the following equivalent optimization problem of Problem (6)

$$\begin{aligned} \max_{\mathbf{z},\mathbf{p},\mathbf{q}} \quad & \sum_{j=1}^{J} \sum_k p_{jk} w_j q_{jk} \left(\sum_{i\in o_{jk}} z_i - |o_{jk}| + 1 \right) \\ & + \sum_{a_{hm}\in\mathcal{A}} w_{a_{hm}} z_h (1 - z_m) \\ & -\infty \sum_{a_{hm}\in\mathcal{A}} (1 - z_h) z_m \\ & +\lambda(L - \sum_i z_i) \\ \text{s.t.} \quad & \mathbf{z}, \mathbf{q} \text{ are binary} \\ & \mathbf{p}_j \in \Delta \qquad \forall j \end{aligned} \tag{7}$$

where $|o_{jk}|$ is the size of o_{jk}.

Let $R(\mathbf{z}, \mathbf{p}, \mathbf{q})$ denote the objective function of Problem (7), to search the optimal point, we alternatively update $\mathbf{p}$ and $\mathbf{z}, \mathbf{q}$. First we initialize $\mathbf{p} = \mathbf{p}(0)$. In each iteration, we first fix $\mathbf{p}$. It is obvious that Problem (7) is a SBQP, hence the optimal $\mathbf{z}, \mathbf{q}$ can be solved efficiently using max-flow/min-cut. Then we fix $\mathbf{z}, \mathbf{q}$, and update $\mathbf{p}$ using projected

subgradient. That is

$$\mathbf{p}_j^{\text{new}} = P_\Delta\left(\mathbf{p}_j + \frac{\partial R}{\partial \mathbf{p}_j}\alpha\right) \quad (8)$$

where $\alpha > 0$ is the step size in line search, and function $P_\Delta(q)$ denotes the projection of q onto the feasible set Δ

$$P_\Delta(\mathbf{q}) = \min_{\mathbf{p}\in\Delta} \|\mathbf{p} - \mathbf{q}\|_2$$

which can be solved efficiently by sorting (Duchi et al., 2008).

3.4 Initialize p Using Convex Relaxation

Since R is non-concave, searching its maximum using subgradient method suffers from local optimality. Though one can use techniques such as branch-and-bound for exact inference (Qian and Liu, 2013; Gormley and Eisner, 2013), here for fast decoding, we use convex relaxation to choose a good seed $\mathbf{p}(0)$. Recall that p_{jk} denotes the percentage of the k^{th} occurrence contributing to c_j. The larger p_{jk} is, the more likely the k^{th} occurrence is selected. To estimate such likelihood, we replace the binary constraint in extractive summarization (Problem (1)) by $\mathbf{0} \leq \mathbf{y}, \mathbf{v} \leq \mathbf{1}$, since solving a relaxed LP is much faster than ILP. Suppose $\mathbf{y}^*$ is the optimal solution for such a relaxed LP problem, we initialize $\mathbf{p}$ by

$$p_{jk} = \frac{y^*_{n_{jk}}}{\sum_k y^*_{n_{jk}}} \quad (9)$$

If for all k, $y^*_{n_{jk}} = 0$, then we initialize p_{jk} using uniform distribution

$$p_{jk} = \frac{1}{|o_j|}$$

where $|o_j|$ is the frequency of c_j.

3.5 Summary

For clarity, we summarize our decoding algorithm in Algorithm 1. Initial λ_{max} can be arbitrarily large. In our experiments, we set $\lambda_{max} = \sum_j w_j$, which empirically guarantees the summary length $\sum_i z_i \leq L$ when $\lambda = \lambda_{max}$. The choice of the step size for updating $\mathbf{p}$ is similar to the projected subgradient method in dual decomposition (Koo et al., 2010).

Algorithm 1 Compressive Summarization via Graph Cuts

Require: Scores of concepts $\{w_j\}$ and arcs $\{w_{a_{hm}}\}$, max summary length L.
Ensure: Compressive summarization $\mathbf{z}^*$, where z_i indicates whether the i^{th} word is selected.
Solve the relaxed LP of Problem (1) (replace the binary constraint by $\mathbf{0} \leq \mathbf{y}, \mathbf{v} \leq \mathbf{1}$) to get $\mathbf{y}$.
Initialize $\mathbf{p}(0)$ using Eq (9).
Initialize sufficient large λ_{max}, and $\lambda_{min} = 0$
while $\lambda_{max} - \lambda_{min} > \epsilon$ **do**
  Set $\lambda = \frac{1}{2}(\lambda_{min} + \lambda_{max})$
  Set $\mathbf{p} = \mathbf{p}(0)$.
  repeat
    Fix $\mathbf{p}$, solve Problem (7) to get $\mathbf{z}$ using max-flow/min-cut.
    Update $\mathbf{p}$ using Eq (8).
  until convergence
  if $\sum_i z_i > L$ **then**
    $\lambda_{min} = \lambda$
  else if $\sum_i z_i < L$ **then**
    $\lambda_{max} = \lambda$
  else
    break
  end if
end while

4 Features and Hard Constraints

We choose discriminative models to learn the scores of concepts and arcs. For concept c_j, its score is

$$w_j = \theta^T_{\text{concept}} \mathbf{f}_{\text{concept}}(c_j)$$

where $\mathbf{f}_{\text{concept}}(c_j)$ is the feature vector of c_j, and θ_{concept} is the corresponding weight vector of feature $\mathbf{f}_{\text{concept}}(c_j)$. Similarly, score $w_{a_{hm}}$ is defined as

$$w_{a_{hm}} = \theta^T_{\text{arc}} \mathbf{f}_{\text{arc}}(a_{hm})$$

Though our algorithm can handle general word n-gram concepts, we restrict the concepts to word bigrams, which have been widely used recently in the sentence-based ILP extractive summarization systems. For a concept c_j, we define the following features, some of which have been used in previous work (Brandow et al., 1995; Aker and Gaizauskas, 2009; Edmundson, 1969; Radev, 2001; Li et al., 2013b). All of these features are non-negative.

- Term frequency: the frequency of c_j in the given topic.

- Stop word ratio: ratio of stop words in c_j. The value can be $\{0, 0.5, 1\}$.
- Similarity with topic title: the number of common words in these two strings, divided by the length of the longer string.
- Document ratio: percentage of documents containing c_j.
- Sentence ratio: percentage of sentences containing c_j.
- Sentence-title similarity: word unigram/bigrams cosine similarity between the sentence containing c_j and the topic title. For concepts appearing in multiple sentences, we choose the maximal similarity.
- Sentence-query similarity: word unigram/bigram cosine similarity between the sentence containing c_j and the topic query (concatenation of topic title and description). For concepts appearing in multiple sentences, we choose the maximal similarity.
- Sentence position: position of the sentence containing c_j in the document. For concepts appearing in multiple sentences, we choose the minimum.
- Sentence length: length of the sentence containing c_j. For concepts appearing in multiple sentences, we choose the maximum.
- Paragraph starter: binary feature indicating whether c_j appears in the first sentence of a paragraph.

For subtree deletion model, we define the following features for arc a_{hm}.

- POS tags of head word x_h and child word x_m and their concatenations.
- Dependency label of arc a_{hm} and its parent arc.
- Word x_m if x_m is a conjunction word or preposition word. Word x_h if x_m is a conjunction word or preposition word.
- Binary feature indicating whether the modifier x_m is a temporal word such as *Friday*.

We also define some hard constraints for subtree deletion to improve the readability of the generated compressed sentences.

- **C0** Arc a_{hm} can be cut only if one of the two conditions holds: (1) there is a comma, colon, or semicolon between the head and the modifier; (2) the modifier word is a preposition (POS tag is *IN*) or a wh-word, such as *what, who, whose* (corresponds to POS tag *IN, WDT, WP, WP$, WRB*).
- **C1** Arcs with dependency labels *SUB, OBJ, PRD, SBAR* or *VC* can not be cut.
- **C2** Arcs in set phrases like *so far, more than, according to* can not be cut.
- **C3** All arcs in coordinate structures can not be cut, such as *cats and dogs*.

Note that compared with previous work, our compression is more conservative. Constraint C0 allows only a small portion of arcs to be cut. This is based on our observation of the sentence compression corpus: removing preposition phrases (PP) or sub-clauses can greatly reduce the length of sentence, while hurting the readability little. Cutting other arcs like *NMOD* usually removes only one or two words, and possibly affects the sentence's readability.

5 Experimental Results

5.1 Experimental Setup

Due to the lack of training data for compressive summarization, we learn the subtree deletion model and the concept model separately. Specifically, the sentence compression dataset (Clarke and Lapata, 2008) (referred as CL08) is used for subtree deletion model training (θ_{arc}). A sentence pair in the corpus is kept for training the subtree deletion model if the compressed sentence can be derived by deleting subtrees from the parse tree of the original sentence. There are $3,178$ out of $5,739$ such pairs. The concept model (θ_{concept}) is learned from the TAC2009 dataset. We create the oracle extractive summaries with the maximal bigram recall as the reference summary. TAC2010 data is used as

	Corpus	Sent.	Words	Topics
Train	TAC2009	4,216	117,304	44
	CL08	3,178	52,624	N/A
Develop	TAC2010	2,688	72,609	46
Test	TAC2008	4,518	123,946	48

Table 1: Corpus statistics. Training data consist of two parts, TAC2009 for learning the concept model, CL08 (Clarke and Lapata, 2008) for learning the subtree deletion model.

development set for various parameter tuning. Table 1 has the descriptions of all the data used.

We choose averaged perceptron for fast training. The number of iterations is tuned on the development data. Remind that our algorithm is based on the assumption that scores of concepts are non-negative, $\forall j, w_j \geq 0$. We assume that feature vector $\mathbf{f}_{\text{concept}}$ is non-negative (e.g., term frequency, n-gram features), then $\theta_{\text{concept}} \geq 0$ is required to guarantee the non-negativity of w_j. Therefore, we project θ_{concept} onto the non-negative space after each iteration. Since training is offline, we use ILP based exact inference for accurate learning. [1]

To control the contributions of the concept model and the subtree deletion model, we introduce a parameter μ, and modify the original maximization problem (Problem 2) to:

$$\max_{\mathbf{z},\mathbf{v}} \quad \sum_{j=1}^{J} w_j \cdot v_j + \mu \times \sum_{a_{hm} \in \mathcal{A}} w_{a_{hm}} z_h (1 - z_m)$$

We tune μ on TAC2010 dataset. For max-flow/min-cut, in our experiments, we implemented the improved shortest augmented path (SAP) method (Edmonds and Karp, 1972).

For performance measure of the summaries, we use both ROUGE and linguistic quality. ROUGE has been widely used for summarization performance and can measure the informativeness of the summaries (content match between system and reference summaries). Since joint compression and summarization tends to pick isolated words to maximize the information coverage in the system generated summaries, it may have poor readability. Therefore we conduct human evaluation for the linguistic quality for various systems. The linguistic quality consists of two parts. One evaluates the grammar quality within a sentence. Annotators marked if a compressed sentence is grammatically correct. Typical grammar errors include lack of verb or subordinate clause. The other evaluates the coherence between sentences, including the order of sentences and irrelevant sentences. For compressive summaries, we removed the sentences with grammar errors when evaluating coherence. The overall linguistic quality score is the combined score of the percentage of grammatically correct sentences and the correct ordering of the summary sentences. The score is scaled and ranges from 1 (bad) to 10 (good).

[1] we choose the GLPK as our ILP solver, which is used in (Berg-Kirkpatrick et al., 2011)

5.2 Results on the Development Set

We conducted a series of experiments on the development dataset to investigate the effect of the non-negative score assumption, SBQP approximation, and initialization. First, we build a standard ILP based compressive summarizer without the non-negative score assumption. We varied μ over $\{2^{-4}, 2^{-3}, \ldots 2^{4}\}$ and selected the optimal $\mu = 2^{-2}$ according to both ROUGE-2 score and linguistic quality. This interpolation weight is used in all the other experiments.

To study the impact of the non-negative score assumption, we build another summarizer by replacing the concept model with the one trained under the non-negative constraint. We also compared three different initialization strategies for $\mathbf{p}$. The first one is uniform initialization, i.e., $p_{jk} = \frac{1}{|o_j|}$. The second one is a greedy approach, where extractive summarization is obtained by greedy search (i.e., add the top ranked sentence iteratively), then we use the corresponding $\mathbf{y}$ and Eq (9) to initialize $\mathbf{p}$. The last one is our convex relaxation method described in Section 3.4.

Table 2 shows the comparison results. For comparison, we also include the sentence-based ILP extractive summarization results. We can see that the impact of initial $\mathbf{p}$ is substantial. Using convex relaxation helps our method to survive from local optimality. The non-negativity assumption has very little effect on the standard compressive summarization (comparing the first two rows). This empirical result demonstrates the appropriateness of the assumption we use in our proposed method.

System	R-2	LQ
ILP ($\mu = 2^{-2}$)	11.22	6.3
ILP (Non Neg.)	11.18	6.4
Graph Cut (uniform)	9.54	5.9
Graph Cut (greedy)	10.13	6.2
Graph Cut (LP)	11.06	6.1
Sent Extractive	10.11	7.3

Table 2: Experimental results on development dataset. R-2 and LQ are short for ROUGE-2 score multiplied by 100, and linguistic quality respectively.

5.3 Results on Test Dataset

Table 3 shows the summarization results for various systems on the TAC2008 data set. We show both the summarization performance and the speed[2] of the system. The presented systems include our graph-cut based method, the ILP based compression and summarization, and the sentence-based extractive summarization. ILP 2-step refers to the 2-step fast decoding strategy proposed by (Berg-Kirkpatrick et al., 2011).

We also list the performance of some state-of-the-art systems, including the two ICSI systems (Gillick et al., 2008), the compressive summarization system of Berg-Kirkpatrick et al. (2011) (GBK'11), the multi-aspect ILP system of Woodsend and Lapata (2012)(WL'12) and the dual decomposition based system (Almeida and Martins, 2013) (AM'13). Note that for these referred systems since the linguistic quality results are not comparable due to different judgment methods. For our graph-cut based method, to study the tradeoff between the readability of the summary and the ROUGE scores, we present two versions for this method: one uses all the constraints (C0-C3), the other does not use C0.

We can see that our proposed method balanced speed and quality. Compared with ILP, we achieved competitive ROUGE scores, but with about 100x speedup. Our method is also faster than the 2-step ILP system. We also tried another state-of-the-art LP solver, Gurobi version 5.5[3], it achieves 0.17 seconds per topic, much faster than GLPK, but still slower than ours. Regarding the grammar constraints used in our system, from the two results for our graph-cut based method, we can see that adding constraint C0 significantly decreases the R-2 score but improves the language quality. This shows that word-based joint compression and summarization can improve ROUGE score; however, we need to keep in mind about linguistic quality and find a tradeoff between the ROUGE score and the linguistic quality. Almeida and Martins (2013) trained their model on extra corpora using multi-task learning, and achieved better results than ours. The results of our system and theirs showed that Lagrangian relaxation based method combined with combinatorial optimization algorithms such as dynamic programming or minimum cut can exploit the inner structure of problems and achieve significant speedup over ILP.

System	R-2	R-SU4	LQ	sec.
Graph Cut	11.74	14.54	6.5	0.056
Graph Cut w/o C0	12.05	14.71	5.4	0.053
ILP	11.86	14.62	6.6	5.5
ILP (Non Neg.)	11.82	14.60	6.6	5.2
ILP (2-step)	11.72	14.49	6.5	1.1
Sent Extractive	11.06	13.93	7.1	0.13
ICSI-1	11.0	13.4	-	$0.38^{\dagger}$
ICSI-2	11.1	14.3	-	-
BGK'11	11.70	14.38	$6.5^{\dagger}$	-
WL'12	11.37	14.47	-	-
AM'13	12.30^{+}	15.18^{+}	$4.2^{\dagger}$	$0.41^{\dagger}$

Table 3: Experimental results on TAC2008 dataset. Columns 2-5 are scores of ROUGE-2, ROUGE-SU4, linguistic quality, and speed (seconds per topic). ROUGE-2 and ROUGE-SU4 scores are multiplied by 100. All the experiments are conducted on the platform Intel Core i5-2500 CPU 3.30GHz. † numbers are not directly comparable due to different annotations or platforms. $^{+}$ extra resources are used.

Four example summaries produced by our system are shown below. Words in gray are not selected in the summary.

[2]For fair comparison, we only recode the running time for decoding. Other time costs like feature extraction, IO operations are excluded.

[3]www.gurobi.com

India's space agency is ready to send a man to space within seven years if the government gives the nod, while preparations have lready begun for the launch of an unmanned lunar mission, a top official said. India will launch more missions to the moon if its maiden unmanned spacecraft Chandrayaan-1, slated to be launched by 2008, is successful a top space fficial said Tuesday. The United States, the European Space Agency, China, Japan and India are all planning lunar missions during the ext decade.India is "a step ahead" of China in satellite technology and can surpass Beijing in space research by tapping the talent of its huge pool of young scientists, India's space research chief said Monday. The space agencies of India and France signed an agreement on Friday to cooperate in launching a satellite in four years that will help make climate predictions more accurate. The Indian Space Research Organization (ISRO) has short-listed experiments from five nations including the United States, Britain and Germany, for a slot on India's unmanned moon mission Chandrayaan-1 to be undertaken by 2006-2007, the Press Trust of India (PTI) reported Monday. A three-member Afghan delegation is in Bangalore seeking help to set up a high-tech telemedicine facility in 10 Afghan cities linked via Indian satellites, Indo-Asian News Service reported Saturday.

A woman was killed in Mississippi when a tree crashed on her car, becoming the 11th fatality blamed on the powerful Hurricane Katrina that slammed the US Gulf coast after pounding Florida, local TV reported Monday. The bill for the Hurricane Katrina disaster effort has so far reached 2.87 billion dollars, federal officials said on Tuesday. The official death toll from Hurricane Katrina has risen to 118 people in and around the swamped city of New Orleans, officials said Thursday. The Foreign Ministry on Friday reported the first confirmed death of a Guatemalan due to Hurricane Katrina in the United States. The Ugandan government has pledged 200,000 US dollars toward relief and rebuilding efforts in the aftermath of Hurricane Katrina, local press reported on Friday. Swiss Reinsurance Co., the world's second largest reinsurance company on Monday doubled to 40 billion US dollars its initial estimate of the global insured losses caused by Hurricane Katrina in the United States.

The A380 'superjumbo', which will be presented to the world in a lavish ceremony in southern France on Tuesday, will be profitable from 2008, its maker Airbus told the French financial newspaper La Tribune. The A380 will take over from the Boeing 747 as the biggest jet in the skies. An association of residents living near Paris's Charles-de-Gaulle airport on Wednesday denounced the noise pollution generated by the giant Airbus A380, after the new airliner's maiden flight. One problem that Airbus is encountering with its new A380 is that the craft pushes the envelope on the maximum size of a commercial airplane. With a whisper more than a roar, the largest passenger airliner ever built, the Airbus 380, took off on its maiden flight Wednesday.

"When she came in, she was in good spirits," a prison staffer told the New York Daily News. Martha Stewart, the American celebrity homemaker who had her own cooking and home improvement TV show, reported to a federal prison in Alderson, West Virginia, on Friday to serve a five-month sentence for lying about a stock sale. Home fashion guru Martha Stewart said on Friday that she has adjusted to prison life and is keeping busy behind bars since reporting a week ago to a federal penal camp in West Virginia, where she is serving a five-month sentence for lying about a stock sale. The lawyer said he did not know what she is writing, but Stewart has suggested since her conviction that she might write a book about her recent experience with the legal system. Walter Dellinger, the lawyer leading the appeal, said on NBC's "Today" that Stewart is exploring "innovative ways to do microwave cooking" The lawyer said he did not know with her fellow inmates. As Martha Stewart arrives at the red-brick federal prison in Alderson, W. Va., on Friday to begin a five-month sentence, the company she founded is focused both on life without her and on life once she returns.

In most cases, the removed phrases do not hurt the readability of the summaries. The errors are mainly caused by the break of sub-clauses or main clauses that are separated by commas, for example, the fourth sentence in the last summary, *The lawyer said he did not know what she is writing*. The compressed sentence is grammatically correct, but semantically incomplete. Other errors are due to the lack of verb, subject, or object, or incorrect removal of PP, such as the last sentence of the last summary.

6 Conclusion

In this paper, we propose a fast decoding algorithm for compressive summarization using graph cuts. Our idea is to approximate the original ILP problem using supermodular binary quadratic programming (SBQP) problem. Under the assumption that scores of concepts are non-negative, we eliminate subtree constraints and grammar constraints, and relax the length constraint and non-supermodular part of the problem step by step. Our experimental results showed that the graph cut based method achieved competitive performance compared to ILP, while about 100 times faster.

There are several possibilities for further research involving our graph cut algorithms. One idea is to apply it to the language model based compression method (Clarke and Lapata, 2008). The other is to adapt it to social media text summarization task, where text is much more noisy. As graph cut is a general method, applying it to other binary structured learning tasks is also an interesting direction.

Acknowledgments

We'd like to thank three anonymous reviewers for their valuable comments. This work is partly supported by NSF award IIS-0845484 and DARPA under Contract No. FA8750-13-2-0041. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

A. Aker and R. Gaizauskas. 2009. Summary generation for toponym-referenced images using object type language models. In *Proceedings of RANLP*.

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*, pages 196–206, August.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL-HLT*, pages 481–490, June.

A. Billionnet and M. Minoux. 1985. Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics*, 12(1):1 – 11.

Endre Boros and Peter L. Hammer. 2002. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1C3):155 – 225.

Ronald Brandow, Karl Mitze, and Lisa F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing & Management*, 31(5):675 – 685.

Yllias Chali and Sadid A. Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. In *COLING*, pages 457–474.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *J. Artif. Intell. Res. (JAIR)*, 31:399–429.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of ICML*, pages 272–279.

Jack Edmonds and Richard M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April.

H. P. Edmundson. 1969. New methods in automatic extracting. *J. ACM*, 16(2):264–285, April.

Daniel Freedman and Petros Drineas. 2005. Energy minimization via graph cuts: Settling what is possible. In *CVPR (2)*, pages 939–946.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18, June.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The ICSI summarization system at tac 2008. In *Proceedings of the Text Understanding Conference*.

Matthew R. Gormley and Jason Eisner. 2013. Nonconvex global optimization for latent-variable models. In *Proceedings of ACL*, pages 444–454, August.

Vladimir Kolmogorov and Ramin Zabih. 2004. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP 2010*, pages 1288–1298, October.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013a. Document summarization via guided sentence compression. In *Proceedings of EMNLP (to appear)*, October.

Chen Li, Xian Qian, and Yang Liu. 2013b. Using supervised bigram-based ILP for extractive summarization. In *Proceedings of ACL*, pages 1004–1013, August.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of ACL*, pages 510–520, June.

Fei Liu and Yang Liu. 2009. From extractive to abstractive meeting summaries: Can it be done by sentence compression? In *Proceedings of ACL-IJCNLP 2009*, pages 261–264, August.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 1–9.

Kiyohito Nagano, Yoshinobu Kawahara, and Kazuyuki Aihara. 2011. Size-constrained submodular minimization through minimum norm base. In *ICML*, pages 977–984.

Xian Qian and Yang Liu. 2013. Branch and bound algorithm for dependency parsing with non-local features. *TACL*, 1:105–151.

Dragomir R. Radev. 2001. Experiments in single and multidocument summarization using mead. In *In First Document Understanding Conference*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP-CoNLL*, pages 233–243, July.

Inducing Document Plans for Concept-to-text Generation

Ioannis Konstas and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
ikonstas@inf.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In a language generation system, a content planner selects which elements must be included in the output text and the ordering between them. Recent empirical approaches perform content selection without any ordering and have thus no means to ensure that the output is coherent. In this paper we focus on the problem of generating text from a database and present a trainable end-to-end generation system that includes both content selection and ordering. Content plans are represented intuitively by a set of grammar rules that operate on the document level and are acquired automatically from training data. We develop two approaches: the first one is inspired from Rhetorical Structure Theory and represents the document as a tree of discourse relations between database records; the second one requires little linguistic sophistication and uses tree structures to represent global patterns of database record sequences within a document. Experimental evaluation on two domains yields considerable improvements over the state of the art for both approaches.

1 Introduction

Concept-to-text generation broadly refers to the task of automatically producing textual output from non-linguistic input (Reiter and Dale, 2000). Depending on the application and the domain at hand, the input may assume various representations including databases, expert system knowledge bases, simulations of physical systems, or formal meaning representations. Generation systems typically follow a pipeline architecture consisting of three components: *content planning* (selecting and ordering the parts of the input to be mentioned in the output text), *sentence planning* (determining the structure and lexical content of individual sentences), and *surface realization* (verbalizing the chosen content in natural language). Traditionally, these components are hand-engineered in order to ensure output of high quality.

More recently there has been growing interest in the application of learning methods because of their promise to make generation more robust and adaptable. Examples include learning which content should be present in a document (Duboue and McKeown, 2002; Barzilay and Lapata, 2005), how it should be aligned to utterances (Liang et al., 2009), and how to select a sentence plan among many alternatives (Stent et al., 2004). Beyond isolated components, a few approaches have emerged that tackle concept-to-text generation end-to-end. Due to the complexity of the task, most models simplify the generation process, e.g., by treating sentence planning and surface realization as one component (Angeli et al., 2010), by implementing content selection without any document planning (Konstas and Lapata, 2012; Angeli et al., 2010; Kim and Mooney, 2010), or by eliminating content planning entirely (Belz, 2008; Wong and Mooney, 2007).

In this paper we present a trainable end-to-end generation system that captures all components of the traditional pipeline, including document planning. Rather than breaking up the generation process into a sequence of local decisions, each learned separately (Reiter et al., 2005; Belz, 2008; Chen and Mooney, 2008; Kim and Mooney, 2010), our model performs content planning (i.e., document planning and content selection), sentence planning (i.e., lex-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1503–1514,
Seattle, Washington, USA, 18-21 October 2013.

Database Records
temp(time:6-21, min:**9**, mean:**15**, max:**21**) wind-spd(time:6-21, min:**15**, mean:**20**, max:**30**) sky-cover(time:6-9, percent:*25-50*) sky-cover(time:9-12, percent:*50-75*) wind-dir(time:6-21, mode:SSE) gust(time:6-21, min:**20**, mean:**30**, max:**40**)
Output Text
Cloudy, with a high around 20. South southeast wind between 15 and 30 mph. Gusts as high as 40 mph.

(a) WEATHERGOV

Database Records
desktop(cmd:lclick, name:*start*, type:button) start(cmd:lclick, name:*settings*, type:button) start-target(cmd:lclick, name:*control panel*, type:button) win-target(cmd:dblclick, name:*users and passwords*, type:item) contMenu(cmd:lclick, name:*advanced*, type:tab) action-contMenu(cmd:lclick, name:*advanced*, type:button)
Output Text
Click start, point to settings, and then click control panel. Double-click users and passwords. On the advanced tab, click advanced.

(b) WINHELP

Figure 1: Database records and corresponding text for (a) weather forecasting and (b) Windows troubleshooting. Each record has a type (e.g., win-target), and a set of fields. Each field has a value, which can be categorical (in typewriter), an integer (in **bold**), or a literal string (in *italics*).

icalization of input entries), and surface realization jointly. We focus on the problem of generating text from a database. The input to our model is a set of database records and collocated descriptions, examples of which are shown in Figure 1.

Given this input, we define a probabilistic context-free grammar (PCFG) that captures the structure of the database and how it can be verbalized. Specifically, we extend the model of Konstas and Lapata (2012) which also uses a PCFG to perform content selection and surface realization, but does not capture any aspect of document planning. We represent content plans with grammar rules which operate on the document level and are embedded on top of the original PCFG. We essentially learn a discourse grammar following two approaches. The first one is linguistically naive but applicable to multiple languages and domains; it extracts rules representing global patterns of record sequences within a sentence and among sentences from a training corpus. The second approach learns document plans based on *Rhetorical Structure Theory* (RST; Mann and Thomson, 1988); it therefore has a solid linguistic foundation, but is resource intensive as it assumes access to a text-level discourse parser.

We learn document plans automatically using both representations and develop a tractable decoding algorithm for finding the best output, i.e., derivation in our grammar. To the best of our knowledge, this is the first data-driven model to incorporate document planning in a joint end-to-end system. Experimental evaluation on the WEATHERGOV (Liang et al., 2009) and WINHELP (Branavan et al., 2009) domains shows that our approach improves over Konstas and Lapata (2012) by a wide margin.

2 Related Work

Content planning is a fundamental component in a natural generation system. Not only does it determine which information-bearing units to talk about, but also arranges them into a structure that creates coherent output. It is therefore not surprising that many content planners have been based on theories of discourse coherence (Hovy, 1993; Scott and de Souza, 1990). Other work has relied on generic planners (Dale, 1988) or schemas (Duboue and McKeown, 2002). In all cases, content plans are created manually, sometimes through corpus analysis. A few researchers recognize that this top-down approach to planning is too inflexible and adopt a generate-and-rank architecture instead (Mellish et al., 1998; Karamanis, 2003; Kibble and Power, 2004). The idea is to produce a large set of candidate plans and select the best one according to a ranking function. The latter is typically developed manually taking into account constraints relating to discourse coherence and the semantics of the domain.

Duboue and McKeown (2001) present perhaps the first empirical approach to content planning. They use techniques from computational biology to learn the basic patterns contained within a plan and the ordering among them. Duboue and McKeown (2002) learn a tree-like planner from an aligned corpus of semantic inputs and corresponding human-authored outputs using evolutionary al-

gorithms. More recent data-driven work focuses on end-to-end systems rather than individual components, however without taking document planning into account. For example, Kim and Mooney (2010) first define a generative model similar to Liang et al. (2009) that selects which database records to talk about and then use an existing surface realizer (Wong and Mooney, 2007) to render the chosen records in natural language. Their content planner has no notion of coherence. Angeli et al. (2010) adopt a more unified approach that builds on top of the alignment model of Liang et al. (2009). They break record selection into a series of locally coherent decisions, by first deciding on what records to talk about. Each choice is based on a history of previous decisions, which is encoded in the form of discriminative features in a log-linear model. Analogously, they choose fields for each record, and finally verbalize the input using automatically extracted domain-specific templates from training data.

Konstas and Lapata (2012) propose a joint model, which recasts content selection and surface realization into a parsing problem. Their model optimizes the choice of records, fields and words simultaneously, however they still select and order records locally. We replace their content selection mechanism (which is based on a simple markovized chaining of records) with global document representations. A *plan* in our model is identified either as a sequence of sentences, each containing a sequence of records, or as a tree where the internal nodes denote discourse information and the leaf nodes correspond to records.

3 Problem Formulation

The generator takes as input a set of database records $\mathbf{d}$ and outputs a text g that verbalizes some of these records. Each record token $r_i \in \mathbf{d}$, with $1 \leq i \leq |\mathbf{d}|$, has a type $r_i.t$ and a set of fields f associated with it. Fields have different values $f.v$ and types $f.t$ (i.e., integer, categorical, or literal strings). For example, in Figure 1b, win-target is a record type with three fields: cmd (denotes the action the user must perform on an object on their screen, e.g., left-click), name (denotes the name of the object), and type (denotes the type of the object). The values of these fields are `dblclick`, *users and passwords*, and `item`; name is a literal string, the rest are categorical.

During training, our algorithm is given a corpus consisting of several *scenarios*, i.e., database records paired with texts w (see Figure 1). For each scenario, the model first decides on a *global document plan*, i.e., it selects which *types of records* belong to each sentence (or phrase) and how these sentences (or phrases) should be ordered. Then it selects appropriate record tokens for each type and progressively chooses the most relevant fields; then, based on the values of the fields, it generates the final text, word by word.

Grammar Rules
1. $\mathrm{S} \rightarrow \mathrm{R}(\mathrm{start})$
2. $\mathrm{R}(r_i.t) \rightarrow \mathrm{FS}(r_j, start)\ \mathrm{R}(r_j.t) \mid \mathrm{FS}(r_j, start)$
3. $\mathrm{FS}(r, r.f_i) \rightarrow \mathrm{F}(r, r.f_j)\ \mathrm{FS}(r, r.f_j) \mid \mathrm{F}(r, r.f_j)$
4. $\mathrm{F}(r, r.f) \rightarrow \mathrm{W}(r, r.f)\ \mathrm{F}(r, r.f) \mid \mathrm{W}(r, r.f)$
5. $\mathrm{W}(r, r.f) \rightarrow \alpha \mid \mathrm{g}(f.v)$

Figure 2: Grammar G of the original model. Parentheses denote features, and impose constraints on the grammar.

4 Original Model

Our work builds on the model developed by Konstas and Lapata (2012). The latter is essentially a PCFG which captures both the structure of the input database and the way it renders into natural language. This grammar-based approach lends itself well to the incorporation of document planning which has traditionally assumed tree-like representations. We first briefly describe the original model and then present our extensions in Section 5.

Grammar Grammar G in Figure 2 defines a set of non-recursive CFG rewrite rules that capture the structure of the database, i.e., the relationship between records, records and fields, fields and words. These rules are domain-independent and could be applied to any database provided it follows the same structure. Non-terminal symbols are in capitals, the terminal symbol α corresponds to the vocabulary of the training set and $g(f.v)$ is a function which generates integers given the field value $f.v$. Note that all non-terminals have features (in parentheses) which

act as constraints and impose non-recursion (e.g., in rule (2) $i \neq j$, so that a record cannot emit itself).

Rule (1) defines the expansion from the start symbol S to the first record R of type *start*. The rules in (2) implement content selection, by choosing appropriate records from the database and generating a sequence. $\mathrm{R}(r_i.t)$ is the source record, $\mathrm{R}(r_j.t)$ is the target record and $\mathrm{FS}(r_j.start)$ is a place-holder symbol for the set of fields of record token r_j. This method is locally optimal, since it only keeps track of the previous type of record for each re-write. The rules in (3) conclude content selection on the field level, i.e., after we have chosen a record, we select and order the corresponding fields. Finally, the rules in (4) and (5) correspond to surface realization. The former rule binarizes the sequence of words emitted by a particular field $r.f$ in an attempt to capture local dependencies between words, such as multi-word expressions (e.g., right click, radio button). The latter rule defines the emission of words and integer numbers[1], given a field type and its value. Note that the original model lexicalizes field values of categorical and integer type only.

Training The rules of grammar G are associated with weights that are learned using the EM algorithm (Dempster et al., 1977). During training, the records, fields and values of database $\mathbf{d}$ and the words w from the associated text are observed, and the model learns the mapping between them. Notice that we use w to denote the gold-standard text and g to refer to the words generated by the model. The mapping between the database and the observed text is unknown and thus the weights of the rules define a hidden correspondence h between records, fields and their values.

Decoding Given a trained grammar G and an input scenario from a database $\mathbf{d}$, the model generates text by finding the most likely derivation, i.e., sequence of rewrite rules for the input. Although resembling parsing, the generation task is subtly different. In parsing, we observe a string of words and our goal is to find the most probable syntactic structure, i.e., hidden correspondence $\hat{h}$. In generation, however, the string is not observed; instead, we must find the best text $\hat{g}$, by maximizing both over h and g, where $g = g_1 \dots g_N$ is a sequence of words licensed by G. More formally:

$$\hat{g} = f\Big(\arg\max_{g,h} P\big((g,h)\big)\Big) \tag{1}$$

where f is a function that takes as input a derivation tree (g,h) and returns $\hat{g}$. Konstas and Lapata (2012) use a modified version of the CYK parser (Kasami, 1965; Younger, 1967) to find $\hat{g}$. Specifically, they intersect grammar G with a n-gram language model and calculate the most probable generation $\hat{g}$ as:

$$\hat{g} = f\Big(\arg\max_{g,h} p(g) \cdot p(g,h \,|\, \mathbf{d})\Big) \tag{2}$$

where $p(g,h \,|\, \mathbf{d})$ is the decoding likelihood for a sequence of words $g = g_1 \dots g_N$ of length N and the hidden correspondence h that emits it, i.e., the likelihood of the grammar for a given database input scenario $\mathbf{d}$. $p(g)$ is a measure of the quality of each output and is provided by the n-gram language model.

5 Extensions

In this section we extend the model of Konstas and Lapata (2012) by developing two more sophisticated content selection approaches which are informed by a global plan of the document to be generated.

5.1 Planning with Record Sequences

Grammar Our key idea is to replace the content selection mechanism of the original model with a *document plan* which essentially defines a grammar on record types. We split a document into sentences, each terminated by a full-stop. Then a sentence is further split into a sequence of record types. Contrary to the original model, we observe a complete sequence[2] of record types, split into sentences. This way we learn domain-specific patterns of frequently occurring record type sequences among the sentences of a document, as well as more local structures within a sentence. We thus substitute rules (1)–(2) in Figure 2 with sub-grammar G_{RSE} based on record type sequences:

Definition 1 (G_{RSE} grammar)

$$G_{RSE} = \{\Sigma_R,\, N_{RSE},\, P_{RSE},\, D\}$$

[1] The function $g(f.v) : \mathbb{Z} \rightarrow \mathbb{Z}$, generates an integer in the following six ways (Liang et al., 2009): identical, rounding up/down to a multiple of 5, rounding off a multiple of 5 and adding or subtracting some noise modelled by a geometric distribution.

[2] Note that a sequence is different from a permutation, as we may allow repetitions or omissions of certain record types.

where Σ_R is a set of terminal symbols $R(r.t)$, and N_{RSE} is a set of non-terminal symbols:

$$N_{RSE} = \{D, SENT\}$$

where D represents the start symbol and $SENT$ a sequence of records. P_{RSE} is a set of production rules of the form:

(a) $D \rightarrow SENT(t_i, \ldots, t_j) \ \ldots \ SENT(t_l, \ldots, t_m)$

(b) $SENT(t_i, \ldots, t_j) \rightarrow R(r_a.t_i) \ \ldots \ R(r_k.t_j) \cdot$

where t is a record type, t_i, t_j, t_l and t_m may overlap and r_a, r_k are record tokens of type t_i and t_j respectively. The corresponding weights for the production rules P_{RSE} are:

Definition 2 (G_{RSE} weights)

(a) $p(t_i, \ldots, t_j, \ldots t_l, \ldots, t_m \mid D)$

(b) $p(t_i) \cdot \ldots \cdot p(t_j) = \frac{1}{|\mathbf{s}(t_i)|} \cdot \ldots \cdot \frac{1}{|\mathbf{s}(t_j)|}$

where $\mathbf{s}(t)$ is a function that returns the set of records with type t (Liang et al., 2009).

Rule (a) defines the expansion from the start symbol D to a sequence of sentences, each represented by the non-terminal $SENT$. Similarly to the original grammar G, we employ the use of features (in parentheses) to denote a sequence of record types. The same record types may recur in different sentences, but not in the same one. The weight of rule (a) is simply the joint probability of all the record types present, ordered and segmented appropriately into sentences in the document, given the start symbol.

Once record types have been selected (on a per sentence basis) we move on to rule (b) which describes how each non-terminal $SENT$ expands to an ordered sequence of records R, as they are observed within a sentence (see the terminal symbol '.' at the end of the rule). Notice that a record type t_i may correspond to several record tokens r_a. Rules (3)–(5) in grammar G make decisions on these tokens based on the overall content of the database and the field/value selection. The weight of this rule is the product of the weights of each record type. This is set to the uniform distribution over $\{1, ..., |\mathbf{s}(t)|\}$ for record type t, where $|\mathbf{s}(t)|$ is the number of records with that type.

Figure 3d shows an example tree for the database input in Figure 1b, using G_{RSE} and assuming that the alignments between records and text are given. The top level of the tree refers to the sequence of record types as they are observed in the text. The first sentence contains three records with types 'desktop', 'start' and 'start-target', each corresponding to the textual segments *click start*, *point to settings*, *and then click control panel*. The next level on the tree, denotes the choice of record tokens for each sentence, provided that we have decided on the choice and order of their types (see Figure 3b). In Figure 3d, the bottom-left sub-tree corresponds to the choice of the first three records of Figure 1b.

Training A straightforward way to train the extended model would be to embed the parameters of G_{RSE} in the original model and then run the EM algorithm using inside-outside at the E-step. Unfortunately, this method will induce a prohibitively large search space. Rule (a) enumerates all possible combinations of record type sequences and the number grows exponentially even for a few record types and a small sequence size. To tackle this problem, we extracted rules for G_{RSE} from the training data, based on the assumption that there will be far fewer unique sequences of record types per dataset than exhaustively enumerating all possibilities.

For each scenario, we obtain a word-by-word alignment between the database records and the corresponding text. In our experiments we used Liang et al.'s (2009) unsupervised model, however any other semi- or fully supervised method could be used. As we show in Section 7, the quality of the alignment inevitably correlates with the quality of the extracted grammar and the decoder's output. We then map the aligned record tokens to their corresponding types, merge adjacent words with the same type and segment on punctuation (see Figure 3b). Next, we create the corresponding tree according to G_{RSE} (Figure 3d) and binarize it. We experimented both with left and right binarization and adhered to the latter, as it obtained a more compact set of rules. Finally, we collectively count the rule weights on the resulting treebank and extract a rule set, discarding rules with frequency less than three.

Using the extracted (weighted) G_{RSE} rules, we run the EM algorithm via inside-outside and learn the weights for the remaining rules in G. Decoding remains the same as in Konstas and Lapata (2012); the only requirement is that the extracted grammar remains binarized in order to guarantee the cubic

desktop$_1$	start$_1$	start-target$_1$	win-target$_1$	contMenu$_1$	action-contMenu$_1$
Click start,	point to settings,	and then click control panel.	Double-click users and passwords.	On the advanced tab ,	click advanced.

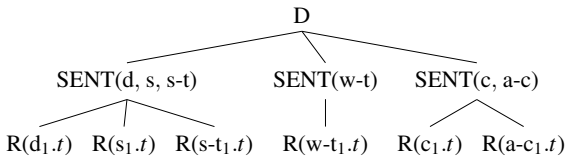
(a) Record token alignments

[desktop start start-target‖win-target‖contMenu action-contMenu‖]

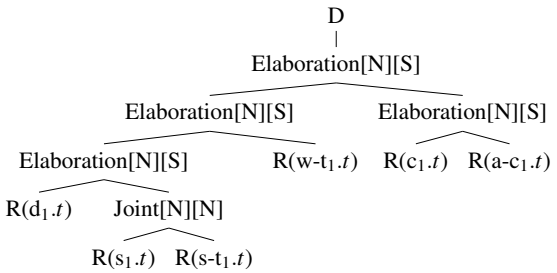
(b) Record type segmentation

[Click start,]$^{desktop_1.t}$ [point to settings,]$^{start_1.t}$ [and then click control panel.]$^{start-target_1.t}$ [Double-click users and passwords.]$^{win-target_1.t}$ [On the advanced tab,]$^{contMenu_1.t}$ [click advanced.]$^{action-contMenu_1.t}$

(c) Segmentation of text into EDUs



(d) Document plan using the G_{RSE} grammar



(e) Document plan using the G_{RST} grammar

Figure 3: Grammar extraction example from the WINHELP domain using G_{RSE} and G_{RST}. For G_{RSE}, we take the alignments of records on words and map them to their corresponding types (a); we then segment record types into sentences (b); and finally, create a tree using grammar G_{RSE} (c). For G_{RST}, we segment the text into EDUs based on the records they align to (d) and output the discourse tree (omitted here for brevity's sake); we build the document plan once we substitute the EDUs with their corresponding record types (e).

bound of the Viterbi search algorithm. Note that the original grammar is limited to the generation of categorical and integer values. We extend it to support the generation of strings. The following rule adds a simple verbatim lexicalization for string values:

$$\mathrm{W}(r, r.f) \rightarrow gen\_str(f.v, i)$$
$$gen\_str(f.v, i) : V \rightarrow V,\ f.v \in V$$

where V is the set of words for the fields of type string, and gen_str is a function that takes the value of a string-typed field $f.v$, and the position i in the string, and generates the corresponding word at that position. For example, gen_str(*users and passwords*, 3) = *passwords*. The weight of this rule is set to 1.

5.2 Planning with Rhetorical Structure Theory

Grammar RST (Mann and Thompson, 1988) is a theory of text organization which provides a framework for analyzing text. A basic tenet of the theory is that a text consists of hierarchically organized text spans or elementary discourse units (EDUs) that stand in various relations to one another (e.g., *Elaboration*, *Attribution*). These "rhetorical relations" hold between two adjacent parts of the text, where typically, one part is "nuclear" and one a "satellite". An analysis of a text consists in identifying the relations holding between successively larger parts of the text, yielding a natural hierarchical description of the rhetorical organization of the text. From its very inception, RST was conceived as a way to characterize text and textual relations for the purpose of text generation.

In order to create a RST-inspired document plan for our input (i.e., database records paired with texts), we make the following assumption: each record corresponds to a unique non-overlapping span in the collocated text, and can be therefore mapped to an EDU. Assuming the text has been segmented and aligned to a sequence of records, we can create a discourse tree with record types (in place of their corresponding EDUs) as leaf nodes. Again, we define a sub-grammar G_{RST} which replaces rules (1)–(2) from Figure 2:

Definition 3 (G_{RST} grammar)

$$G_{RST} = \{\Sigma_R, N_{RST}, P_{RST}, D\}$$

where Σ_R is the alphabet of leaf nodes as defined in Section 5.1, N_{RST} is a set of non-terminals corresponding to rhetorical relations augmented with nucleus-satellite information (e.g., *Elaboration[N][S]* stands for the elaboration relation between the nucleus EDU left-adjoining with the satellite EDU), P_{RST} is the set of production rules of the form $P_{RST} \subseteq N_{RST} \times \{N_{RST} \cup \Sigma_R\} \times \{N_{RST} \cup \Sigma_R\}$ associated with a weight for each rule, and $D \in N_{RST}$ is the root symbol. Figure 3e gives the discourse tree for the database input of Figure 1b, using G_{RST}.

Training In order to obtain the weighted productions of G_{RST}, we use an existing state-of-the-art discourse parser[3] (Feng and Hirst, 2012) trained on the RST-DT corpus (Carlson et al., 2001). The latter contains a selection of 385 Wall Street Journal articles which have been annotated using the framework of RST and an inventory of 78 rhetorical relations, classified into 18 coarse-grained categories (Carlson and Marcu, 2001). Figure 4 gives a comparison of the distribution of relations extracted for the two datasets we used, against the gold-standard annotation of RST-DT. The statistics for the RST-DT corpus are taken from Williams and Power (2008). The relative frequencies of relations on both datasets follow closely the distribution of those in RST-DT, thus empirically supporting the application of the RST framework to our data.

We segment each document in our training set into EDUs based on the record-to-text alignments given by the model of Liang et al. (2009) (see Figure 3c). We then run the discourse parser on the resulting EDUs, and retrieve the corresponding discourse tree; the internal nodes are labelled with one of the RST relations. Finally, we replace the leaf EDUs with their respective terminal symbols $R(r.t) \in \Sigma_R$ (Figure 3e) and collect the resulting grammar productions; their weights are calculated via maximum likelihood estimation based on their collective counts in the parse trees licensed by G_{RST}. Training and decoding of the extended generation model (after we embed G_{RST} in the original grammar G) is performed identically to Section 5.1.

[3]Publicly available from `http://www.cs.toronto.edu/~weifeng/software.html`.

6 Experimental Design

Data Since our aim was to evaluate the planning component of our model, we used datasets whose documents are at least a few sentences long. Specifically, we generated weather forecasts and troubleshooting guides for an operating system. For the first domain (henceforth WEATHERGOV) we used the dataset of Liang et al. (2009), which consists of 29,528 weather scenarios for 3,753 major US cities (collected over four days). The database has 12 record types, each scenario contains on average 36 records, 5.8 out of which are mentioned in the text. A document has 29.3 words and is four sentences long. The vocabulary is 345 words. We used 25,000 scenarios from WEATHERGOV for training, 1,000 scenarios for development and 3,528 scenarios for testing.

For the second domain (henceforth WINHELP) we used the dataset of Branavan et al. (2009), which consists of 128 scenarios. These are articles from Microsoft's Help and Support website[4] and contain step-by-step instructions on how to perform tasks on the Windows 2000 operating system. In its original format, the database provides a semantic representation of the textual guide, i.e., it represents the user's actions on the operating system's UI. We semi-automatically converted this representation into a schema of records, fields and values, following the conventions adopted in Branavan et al. (2009).[5] The final database has 13 record types. Each scenario has 9.2 records and each document 51.92 words with 4.3 sentences. The vocabulary is 629 words. We performed 10-fold cross-validation on the entire dataset for training and testing. Compared to WEATHERGOV, WINHELP documents are longer with a larger vocabulary. More importantly, due to the nature of the domain, i.e., giving instructions, content selection is critical not only in terms of what to say but also in what order.

Grammar Extraction and Parameter Setting We obtained alignments between database records and textual segments for both domains and grammars (G_{RSE} and G_{RST}) using the unsupervised model of Liang et al. (2009). On WEATHERGOV, we extracted a G_{RSE} grammar with 663 rules (after bi-

[4]`support.microsoft.com`

[5]The dataset can be downloaded from `http://homepages.inf.ed.ac.uk/ikonstas/index.php?page=resources`

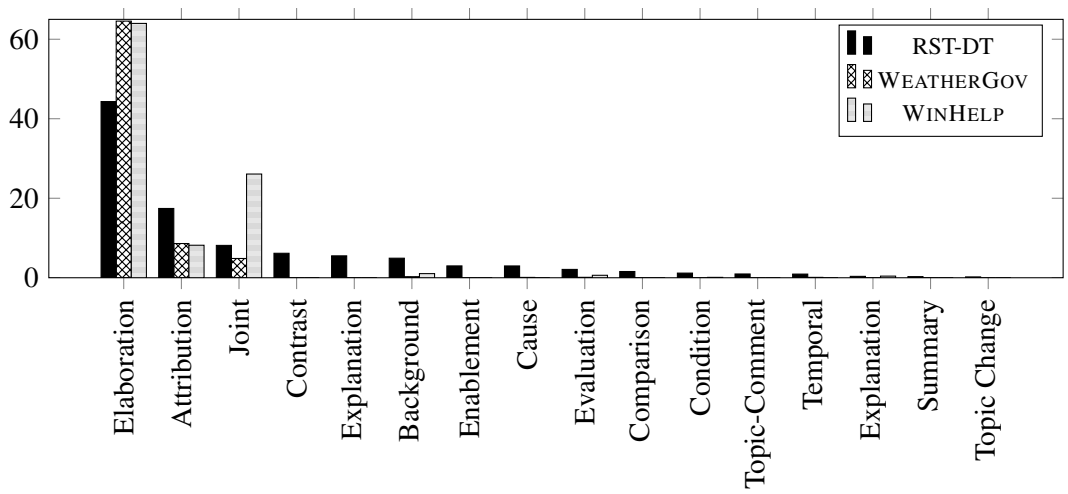


Figure 4: Distribution of RST relations on WEATHERGOV, WINHELP, and the RST-DT (Williams and Power, 2008).

narization). The WINHELP dataset is considerably smaller, and as a result the procedure described in Section 5.1 yields a very sparse grammar. To alleviate this, we *horizontally markovized* the right-hand side of each rule (Collins, 1999; Klein and Manning, 2003).[6] After markovization, we obtained a G_{RSE} grammar with 516 rules. On WEATHERGOV, we extracted 434 rules for G_{RST}. On WINHELP we could not follow the horizontal markovization procedure, since the discourse trees are already binarized. Instead, we performed *vertical markovization*, i.e., annotated each non-terminal with their parent node (Johnson, 1998) and obtained a G_{RST} grammar with 419 rules. The model of Konstas and Lapata (2012) has two parameters, namely the number of k-best lists to keep in each derivation, and the order of the language model. We tuned k experimentally on the development set and obtained best results with 60 for WEATHERGOV and 120 for WINHELP. We used a trigram model for both domains, trained on each training set.

Evaluation We compared two configurations of our system, one with a content planning component based on record type sequences (G_{RSE}) and another one based on RST (G_{RST}). In both cases content plans were extracted from (noisy) unsupervised alignments. As a baseline, we used the original model of Konstas and Lapata (2012). We also compared our model to Angeli et al.'s system (2010), which is state of the art on WEATHERGOV.

System output was evaluated automatically, using the BLEU modified precision score (Papineni et al., 2002) with the human-written text as reference. In addition, we evaluated the generated text by eliciting human judgments. Participants were presented with a scenario and its corresponding verbalization and were asked to rate the latter along three dimensions: fluency (is the text grammatical?), semantic correctness (does the meaning conveyed by the text correspond to the database input?) and coherence (is the text comprehensible and logically structured?). Participants used a five point rating scale where a high number indicates better performance. We randomly selected 12 documents from the test set (for each domain) and produced output with the system of Konstas and Lapata (2012) (henceforth K&L), our two models using G_{RSE} and G_{RST}, respectively, and Angeli et al. (2010) (henceforth ANGELI). We also included the original text (HUMAN) as gold-standard. We obtained ratings for 60 (12×5) scenario-text pairs for each domain. Examples of the documents shown to the participants are given in Table 1.

The study was conducted over the Internet us-

[6] When horizontally markovizing, we can encode an arbitrary amount of context in the intermediate non-terminals that result from this process; in our case we store h=1 horizontal siblings plus the mother left-hand side (LHS) non-terminal, in order to uniquely identify the Markov chain. For example, $A \rightarrow B\ C\ D$ becomes $A \rightarrow B\ \langle A \ldots B \rangle$, $\langle A \ldots B \rangle \rightarrow C\ \langle A \ldots C \rangle$, $\langle A \ldots C \rangle \rightarrow D$.

	WEATHERGOV	WINHELP
G_{RSE}	Showers before noon. Cloudy, with a high near 38. Southwest wind between 3 and 8 mph. Chance of precipitation is 55 %.	Right-click my network places, and then click properties. Right-click local area connection, and click properties. Click to select the file and printer sharing for Microsoft networks, and then click ok.
G_{RST}	Showers likely. Mostly cloudy, with a high around 38. South wind between 1 and 8 mph. Chance of precipitation is 55 %.	Right-click my network places, and then click properties. Right-click local area connection. Click file and printer sharing for Microsoft networks, and click ok.
K&L	A chance of showers. Otherwise, cloudy, with a high near 38. Southwest wind between 3 and 8 mph.	Right-click my network places, click properties. Right-click local area connection. Click to select the file and printer sharing for Microsoft networks, and then click ok.
ANGELI	A chance of rain or drizzle after 9am. Mostly cloudy, with a high near 38. Southwest wind between 3 and 8 mph. Chance of precipitation is 50 %.	Right-click my network places, and then click properties on the tools menu, and then click properties. Right-click local area connection, and then click properties. Click file and printer sharing for Microsoft networks, and then click ok.
HUMAN	A 50 percent chance of showers. Cloudy, with a high near 38. Southwest wind between 3 and 6 mph.	Right-click my network places, and then click properties. Right-click local area connection, and then click properties. Click to select the file and printer sharing for Microsoft networks check box. Click ok.

Table 1: Human-authored text and system output on WEATHERGOV and WINHELP.

ing Amazon Mechanical Turk[7], and involved 200 volunteers (100 for WEATHERGOV, and 100 for WINHELP), all self reported native English speakers. For WINHELP, we made sure participants were computer-literate and familiar with the Windows operating system by administering a short questionnaire prior to the experiment.

7 Results

The results of the automatic evaluation are summarized in Table 2. Overall, our models outperform K&L's system by a wide margin on both datasets. The two content planners (G_{RSE} and G_{RST}) perform comparably in terms of BLEU. This suggests that document plans induced solely from data are of similar quality to those informed by RST. This is an encouraging result given that RST-style discourse parsers are currently available only for English. ANGELI performs better on WEATHERGOV possibly due to better output quality on the surface level. Their system defines trigger patterns that specifically lexicalize record fields containing numbers. In contrast, on WINHELP it is difficult to explicitly specify such patterns, as none of the record fields are numeric; as a result their system performs poorly compared to the other models.

To assess the impact of the alignment on the content planner, we also extracted G_{RSE} from cleaner alignments which we obtained automatically via human-crafted heuristics for each domain. The heuristics performed mostly anchor matching between database records and words in the text (e.g., the value `Lkly` of the field rainChance, matches with the string *rain likely* in the text). Using these alignments, G_{RSE} obtained a BLEU score of 39.23 on WEATHERGOV and 41.35 on WINHELP. These results indicate that improved alignments would lead to more accurate grammar rules. WEATHERGOV seems more sensitive to the alignments than WINHELP. This is probably because the dataset shows more structural variations in the choice of record types at the document level, and therefore the grammar extracted from the unsupervised alignments is noisier. Unfortunately, performing this kind of analysis for G_{RST} would require gold standard segmentation of our training corpus into EDUs which we neither have nor can easily approximate via heuristics.

The results of our human evaluation study are shown in Table 3. We carried out an Analysis of Variance (ANOVA) to examine the effect of system

[7] https://www.mturk.com

Model	WEATHERGOV	WINHELP
G_{RSE}	35.60	40.92
G_{RST}	36.54	40.65
K&L	33.70	38.26
ANGELI	38.40	32.21

Table 2: Automatic evaluation of system output using BLEU-4.

	WEATHERGOV			WINHELP		
Model	FL	SC	CO	FL	SC	CO
G_{RSE}	4.25	3.75	4.18	3.59	3.21	3.35
G_{RST}	4.10	3.68	4.10	3.45	3.29	3.22
K&L	3.73	3.25	3.59	3.27	2.97	2.93
ANGELI	3.90	3.44	3.82	3.44	2.79	2.97
HUMAN	4.22	3.72	4.11	4.20	4.41	4.25

Table 3: Mean ratings for fluency (FL), semantic correctness (SC) and coherence (CO) on system output elicited by humans.

type (G_{RSE}, G_{RST}, K&L, ANGELI, and HUMAN) on fluency, semantic correctness and coherence ratings. Means differences of 0.2 or more are significant at the 0.05 level using a post-hoc Tukey test. Interestingly, we observe that document planning improves system output overall, not only in terms of coherence. Across all dimensions our models are perceived better than K&L and ANGELI. As far as coherence is concerned, the two content planners are rated comparably (differences in the means are not significant). Both G_{RSE} and G_{RST} are significantly better than the comparison systems (ANGELI and K&L). Table 1 illustrates examples of system output along with the gold standard content selection for reference, for the WEATHERGOV and WINHELP domains, respectively.

In sum, we observe that integrating document planning either via G_{RSE} or G_{RST} boosts performance. Document plans induced from record sequences exhibit similar performance, compared to those generated using expert-derived linguistic knowledge. Our systems are consistently better than K&L both in terms of automatic and human evaluation and are close or better than the supervised model of Angeli et al. (2010). We also show that feeding the system with a grammar of better quality can achieve state-of-the-art performance, without further changes to the model.

8 Conclusions

In this paper, we have proposed an end-to-end system that generates text from database input and captures all components of the traditional generation pipeline, including document planning. Document plans are induced automatically from training data and are represented intuitively by PCFG rules capturing the structure of the database and the way it renders to text. We proposed two complementary approaches to inducing content planners. In a first linguistically naive approach, a document is modelled as a sequence of sentences and each sentence as a sequence of records. Our second approach draws inspiration from Rhetorical Structure Theory (Mann and Thomson, 1988) and represents a document as a tree with intermediate nodes corresponding to discourse relations, and leaf nodes to database records.

Experiments with both approaches demonstrate improvements over models that do not incorporate document planning. In the future, we would like to tackle more challenging domains, such as NFL recaps, financial articles and biographies (Howald et al., 2013; Schilder et al., 2013). Our models could also benefit from the development of more sophisticated planners either via grammar refinement or more expressive grammar formalisms (Cohn et al., 2010).

Acknowledgments

We are grateful to Percy Liang and Gabor Angeli for providing us with their code and data. Thanks to Giorgio Satta and Charles Sutton for helpful comments and suggestions. We also thank the members of the Probabilistic Models reading group at the University of Edinburgh for useful feedback.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology and*

Empirical Methods in Natural Language Processing, pages 331–338, Vancouver, British Columbia.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, Suntec, Singapore.

L. Carlson and D. Marcu. 2001. Discourse tagging reference manual. Technical report, Univ. of Southern California / Information Sciences Institute.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of International Conference on Machine Learning*, pages 128–135, Helsinki, Finland.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 11(November):3053–3096.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Robert Dale. 1988. *Generating referring expressions in a domain of objects and processes*. Ph.D. thesis, University of Edinburgh.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society, series B*, 39(1):1–38.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 172–179.

Pablo A. Duboue and Kathleen R. McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of International Natural Language Generation*, pages 89–96, Ramapo Mountains, NY.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 60–68, Jeju Island, Korea.

Eduard Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385.

Blake Howald, Ravikumar Kondadadi, and Frank Schilder. 2013. Domain adaptable semantic clustering in statistical nlg. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 143–154, Potsdam, Germany, March. Association for Computational Linguistics.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, December.

Nikiforos Karamanis. 2003. *Entity Coherence for Descriptive Text Structuring*. Ph.D. thesis, University of Edinburgh.

Tadao Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA.

Rodger Kibble and Richard Power. 2004. Optimising referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

Joohyun Kim and Raymond Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd Conference on Computational Linguistics*, pages 543–551, Beijing, China.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430. Association for Computational Linguistics Morristown, NJ, USA.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761, Montréal, Canada.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

William C. Mann and Sandra A. Thomson. 1988. Rhetorical structure theory. *Text*, 8(3):243–281.

Chris Mellish, Alisdair Knott, Jon Oberlander, and Mick O'Donnell. 1998. Experiments using stochastic search for text planning. In *Proceedings of International Natural Language Generation*, pages 98–107, New Brunswick, NJ.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

Frank Schilder, Blake Howald, and Ravi Kondadadi. 2013. Gennext: A consolidated domain adaptable nlg system. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 178–182, Sofia, Bulgaria, August. Association for Computational Linguistics.

Donia Scott and Clarisse Sieckenius de Souza. 1990. Getting the message across in RST-based text generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, New York.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of Association for Computational Linguistics*, pages 79–86, Barcelona, Spain.

Sandra Williams and Richard Power. 2008. Deriving rhetorical complexity data from the rst-dt corpus. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, May.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of the Human Language Technology and the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–179, Rochester, NY.

Daniel H Younger. 1967. Recognition and parsing for context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Single-Document Summarization as a Tree Knapsack Problem

Tsutomu Hirao† **Yasuhisa Yoshida**† **Masaaki Nishino**† **Norihito Yasuda**‡ **Masaaki Nagata**†
†NTT Communication Science Laboratories, NTT Corporation
2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237, Japan
{hirao.tsutomu,yoshida.y,nishino.masaaki,
nagata.masaaki}@lab.ntt.co.jp
‡ Japan Science and Technology Agency
North 14 West 9, Sapporo, Hokkaido, 060-0814, Japan
yasuda@erato.ist.hokudai.ac.jp

Abstract

Recent studies on extractive text summarization formulate it as a combinatorial optimization problem such as a *Knapsack Problem*, a *Maximum Coverage Problem* or a *Budgeted Median Problem*. These methods successfully improved summarization quality, but they did not consider the rhetorical relations between the textual units of a source document. Thus, summaries generated by these methods may lack logical coherence. This paper proposes a single document summarization method based on the trimming of a discourse tree. This is a two-fold process. First, we propose rules for transforming a rhetorical structure theory-based discourse tree into a dependency-based discourse tree, which allows us to take a tree-trimming approach to summarization. Second, we formulate the problem of trimming a dependency-based discourse tree as a *Tree Knapsack Problem*, then solve it with integer linear programming (ILP). Evaluation results showed that our method improved ROUGE scores.

1 Introduction

State-of-the-art extractive text summarization methods regard a document (or a document set) as a set of textual units (e.g. sentences, clauses, phrases) and formulate summarization as a combinatorial optimization problem, *i.e.* selecting a subset of the set of textual units that maximizes an objective without violating a length constraint. For example, McDonald (2007) formulated text summarization as a *Knapsack Problem*, where he selects a set of textual units that maximize the sum of significance scores of each unit. Filatova et al. (2004) proposed a summarization method based on a *Maximum Coverage Problem*, in which they select a set of textual units that maximizes the weighted sum of the conceptual units (e.g. unigrams) contained in the set. Although, their greedy solution is only an approximation, Takamura et al. (2009a) extended it to obtain the exact solution. More recently, Takamura et al. (2009b) regarded summarization as a *Budgeted Median Problem* and obtain exact solutions with integer linear programming.

These methods successfully improved ROUGE (Lin, 2004) scores, but they still have one critical shortcoming. Since these methods are based on subset selection, the summaries they generate cannot preserve the rhetorical structure of the textual units of a source document. Thus, the resulting summary may lack coherence and may not include significant textual units from a source document.

One powerful and potential way to overcome the problem is to include discourse tree constraints in the summarization procedure. Marcu (1998) regarded a document as a Rhetorical Structure Theory (RST) (William Charles, Mann and Sandra Annear, Thompson, 1988)-based discourse tree (RST-DT) and selected textual units according to a preference ranking derived from the tree structure to make a summary. Daumé et al. (2002) proposed a document compression method that directly models the probability of a summary given an RST-DT by using a noisy-channel model. These methods generate well-organized summaries, however, since they do not formulate summarizations as combinatorial op-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1515–1520,
Seattle, Washington, USA, 18-21 October 2013.

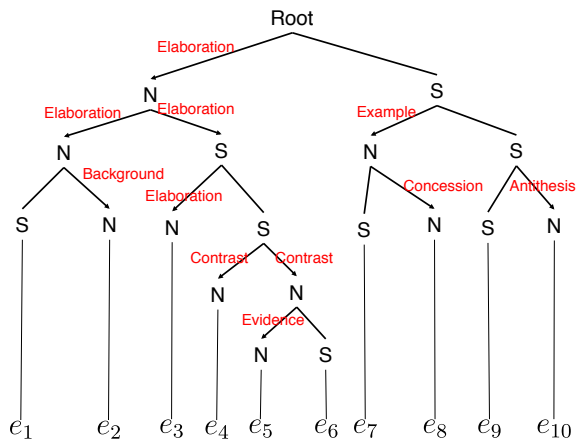


Figure 1: Example RST-DT from (Marcu, 1998).

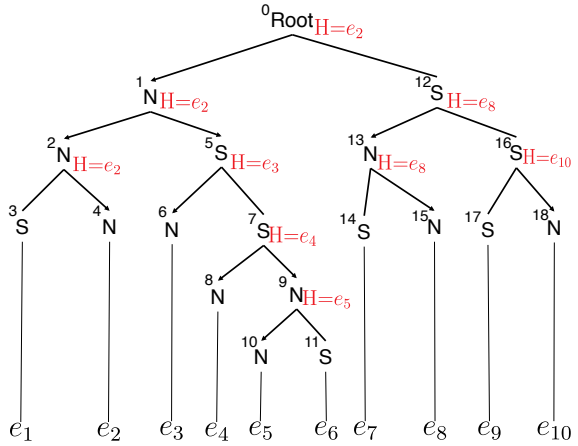


Figure 2: Heads of non-terminal nodes.

timization problems, the optimality of the generated summaries is not guaranteed.

In this paper, we propose a single document summarization method based on the trimming of a discourse tree based on the *Tree Knapsack Problem*. If a discourse tree explicitly represents parent-child relationships between textual units, we can apply the well-known tree-trimming approach to a discourse tree and reap the benefit of combinatorial optimization methods. In other words, to apply the tree-trimming approach, we need a tree whose all nodes represent textual units. Unfortunately, the RST-DT does not allow it, because textual units in the RST-DT are located only on leaf nodes and parent-child relationship between textual units are represented implicitly at higher positions in a tree. Therefore, we first propose rules that transform an RST-DT into a dependency-based discourse tree (DEP-DT) that explicitly defines the parent-child relationships. Second, we treat it as a rooted subtree selection, in other words, a *Tree Knapsack Problem* and formulate the problem as an ILP.

2 From RST-DT to DEP-DT

2.1 RST-DT

According to RST, a document is represented as an RST-DT whose terminal nodes correspond to elementary discourse units (EDU)s[1] and whose non-terminal nodes indicate the role of the contiguous EDUs namely, 'nucleus (N)' or 'satellite (S)'. A nucleus is more important than a satellite in terms of the writer's purpose. That is, a satellite is a child of a nucleus in the RST-DT. Some discourse relations such as 'Elaboration', 'Contrast' and 'Evidence' between a nucleus and a satellite or two nuclei are defined. Figure 1 shows an example of an RST-DT.

2.2 DEP-DT

An RST-DT is not suitable for tree trimming because it does not always explicitly define parent-child relationships between textual units. For example, if we consider how to trim the RST-DT in Figure 1, when we drop e_8, we have to drop e_7 because of the parent-child relationship defined between e_7 and e_8, *i.e.* e_7 is a satellite (child) of the nucleus (parent) e_8. On the other hand, we cannot judge whether we have to drop e_9 or e_{10} because the parent-child relationships are not explicitly defined between e_8 and e_9, e_8 and e_{10}. This view motivates us to produce a discourse tree that explicitly defines parent-child relationships and whose root node represents the most important EDU in a source document. If we can obtain such a tree, it is easy to formulate summarization as a *Tree Knapsack Problem*.

To construct a discourse tree that represents the parent-child relationships between EDUs, we propose rules for transforming an RST-DT to a dependency-based discourse tree (DEP-DT). The procedure is defined as follows:

1. For each non-terminal node excluding the par-

[1]EDUs roughly correspond to clauses.

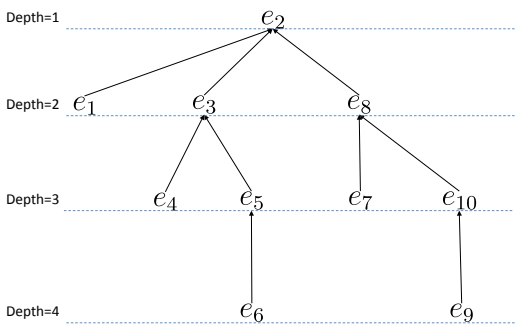


Figure 3: The DEP-DT obtained from the RST-DT in Figure 1.

ent of an EDU in the RST-DT, we define a 'head'. Here, a 'head' of a non-terminal node is the leftmost descendant EDU whose parent is N. In Figure 2, 'H' indicates the 'head' of each node.

2. For each EDU whose parent is N, we pick the nearest S with a 'head' from the EDU's ancestors and we add the EDU to the DEP-DT as a child of the head of the S's parent. If there is no nearest S, the EDU is the root of the DEP-DT. For example, in Figure 2, the nearest S to e_3 that has a head is node 5 and the head of node 5's parent is e_2. Thus, e_3 is a child of e_2.

3. For each EDU whose parent is S, we pick the nearest non-terminal with a 'head' from the ancestors and we add the EDU to the DEP-DT as a child of the head of the non-terminal node. For example, the nearest non-terminal node of e_9 that has a head is node 16 and the head of node 16 is e_{10}. Thus, e_9 is a child of e_{10}.

Figure 3 shows the DEP-DT obtained from the RST-DT in Figure 1. The DEP-DT expresses the parent-child relationship between the EDUs. Therefore, we have to drop e_7, e_9 and e_{10} when we drop e_8. Note that, by applying the rules, discourse relations defined between non-terminals of an RST-DT are eliminated. However, we believe that these relations are no needed for the summarization that we are attempting to realize.

3 Tree Knapsack Model for Single-Document Summarization

3.1 Formalization

We denote T as a set of all possible rooted subtrees obtained from a DEP-DT. $F(t)$ is the significance score for a rooted subtree $t \in T$ and L is the maximum number of words allowed in a summary. The optimal subtree t^* is defined as follows:

$$t^* = \arg\max_{t \in T} F(t) \quad (1)$$

$$s.t. \quad \text{Length}(t) \leq L. \quad (2)$$

Here, we define $F(t)$ as

$$F(t) = \sum_{e \in E(t)} \frac{\mathcal{W}(e)}{\text{Depth}(e)}. \quad (3)$$

$E(t)$ is the set of EDUs contained in t, $\text{Depth}(e)$ is the depth of an EDU e within the DEP-DT. For example, $\text{Depth}(e_2) = 1$, $\text{Depth}(e_6) = 4$ for the DEP-DT of Figure 3. $\mathcal{W}(e)$ is defined as follows:

$$\mathcal{W}(e) = \sum_{w \in W(e)} \text{tf}(w, D). \quad (4)$$

$W(e)$ is the set of words contained in e and $\text{tf}(w, D)$ is the term frequency of word w in a document D.

3.2 ILP Formulation

We formulate the optimization problem in the previous section as a *Tree Knapsack Problem*, which is a kind of *Precedence-Constrained Knapsack Problem* (Samphaiboon and Yamada, 2000) and we can obtain the optimal rooted subtree by solving the following ILP problem[2]:

$$\underset{\boldsymbol{x}}{\text{maximize}} \quad \sum_{i=1}^{N} \frac{\mathcal{W}(e_i)}{\text{Depth}(e_i)} x_i \quad (5)$$

$$s.t. \quad \sum_{i=1}^{N} \ell_i x_i \leq L \quad (6)$$

$$\forall i : x_{\text{parent}(i)} \geq x_i \quad (7)$$

$$\forall i : x_i \in \{0, 1\}, \quad (8)$$

[2]A similar approach has been applied to sentence compression (Filippova and Strube, 2008).

	ROUGE-1		ROUGE-2	
	F	R	F	R
TKP(G)	**.310**^{H,K,L}	**.321**^{G,H,K,L}	**.108**	**.112**^{H}
TKP(H)	.281^{H}	.284^{H}	.092	.093
Marcu(G)	.291^{H}	.272^{H}	.101	.093
Marcu(H)	.236	.219	.073	.068
MCP	.279	.295^{H}	.073	.077
KP	.251	.266^{H}	.071	.075
LEAD	.255	.240	.092	.086

Table 1: ROUGE scores of the RST discourse treebank dataset. In the table, ^{G,H,K,L} indicate a method statistically significant against Marcu(G), Marcu(H), KP, LEAD, respectively.

where $\boldsymbol{x}$ is an N-dimensional binary vector that represents the summary, *i.e.* x_i=1 denotes that the i-th EDU is included in the summary. N is the number of EDUs in a document, ℓ_i is the length (the number of words) of the i-th EDU, and parent(i) indicates the ID of the parent of the i-th EDU in the DEP-DT. Constraint (6) ensures that the length of a summary is less than limit L. Constraint (7) ensures that a summary is a rooted subtree of the DEP-DT. Thus, $x_{\text{parent}(i)}$ is always 1 when the i-th EDU is included in the summary.

In general, the *Tree Knapsack Problem* is NP-hard, but fortunately we can obtain the optimal solution in a feasible time by using ILP solvers for documents of practical tree size. In addition, bottom-up DP (Lukes, 1974) and depth-first DP algorithms (Cho and Shaw, 1997) are known to find the optimal solution efficiently.

4 Experimental Evaluation

4.1 Settings

We conducted an experimental evaluation on the test collection for single document summarization evaluation contained in the RST Discourse Treebank (RST-DTB)(Carlson et al., 2001) distributed by the Linguistic Data Consortium (LDC)[3]. The RST-DTB Corpus includes 385 Wall Street Journal articles with RST annotation, and 30 of these documents also have one human-made reference summary. The average length of the reference summaries corresponds to about 10 % of the words in the source document.

We compared our method (TKP) with Marcu's method (Marcu) (Marcu, 1998), a simple knapsack model (KP), a maximum coverage model (MCP) and a lead method (LEAD). MCP is known to be a state-of-the-art method for multiple document summarization and we believe that MCP also performs well in terms of single document summarization. LEAD is also a widely used summarizer that simply takes the first K textual units of the document. Although this is a simple heuristic rule, it is known as a state-of-the-art summarizer (Nenkova and McKeown, 2011).

For our method, we examined two types of DEP-DT. One was obtained from the gold RST-DT. The other was obtained from the RST-DT produced by a state-of-the-art RST parser, HILDA (duVerle and Prendinger, 2009; Hernault et al., 2010). For Marcu's method, we examined both the gold RST-DT and HILDA's RST-DT. We re-implemented HILDA and re-trained it on the RST-DT Corpus excluding the 30 documents used in the evaluation. The F-score of the parser was around 0.5. For KP, we exclude constraint (7) from the ILP formulation of TKP and set the depth of all EDUs in equations (3) and (5) at 1. For MCP, we use tf (equation (4)) as the word weight.

We evaluated the summarization systems with ROUGE version 1.5.5 [4]. Performance metrics were the recall (R) and F-score (F) of ROUGE-1,2.

4.2 Results and Discussion

Table 1 shows the evaluation results. In the table, TKP(G) and TKP(H) denote methods with the DEP-DT obtained from the gold RST-DT and from HILDA, respectively. Marcu(G) and Marcu(H) denote Marcu's method described in (Marcu, 1998) with gold RST-DT and with HILDA, respectively. We performed a multiple comparison test for the differences among ROUGE scores, we calculated the p-values between systems with the Wilcoxon signed-rank test (Wilcoxon, 1945) and used the False Discovery Rate (FDR) (Benjamini and Hochberg, 1995) to calculate adjusted p-values, in order to limit false positive rate to 5%.

From the table, TKP(G) and Marcu(G) achieved

[3] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T07

[4] Options used: -n 2 -s -m -x

Reference:
The Fuji apple may one day replace the Red Delicious as the number one U.S. apple. Since the Red Delicious has been over-planted and prices have dropped to new lows, the apple industry seems ready for change. Along with growers, supermarkets are also trying different varieties of apples. Although the Fuji is smaller and not as perfectly shaped as the Red Delicious, it is much sweeter, less mealy and has a longer shelf life.

TKP(G):
We'll still have mom and apple pie. A Japanese apple called the Fuji. Some fruit visionaries say the Fuji could someday tumble the Red Delicious from the top of America's apple heap. It has a long shelf life. Now, even more radical changes seem afoot. The Delicious hegemony won't end anytime soon. New apple trees grow slowly. But the apple industry is ripe for change. There's a Fuji apple cult.

Marcu(G):
We'll still have mom and apple pie. On second thought, make that just mom. The Fuji could someday tumble the Red Delicious from the top of America's apple heap. Now, even more radical changes seem afoot. The Delicious hegemony won't end anytime soon. More than twice as many Red Delicious apples are grown as the Golden variety, America's No. 2 apple. But the apple industry is ripe for change.

MCP:
Called the Fuji. It has a long shelf life. New apple trees grow slowly. Its roots are patriotic. I'm going to have to get another job this year. Scowls. They still buy apples mainly for big, red good looks. Japanese researchers have bred dozens of strains of Fujis. Mr. Auvil, the Washington grower, says. Stores sell in summer. The " big boss " at a supermarket chain even rejected his Red Delicious recently. Many growers employ.

LEAD:
Soichiro Honda's picture now hangs with Henry Ford's in the U.S. Automotive Hall of Fame, and the game-show " Jeopardy " is soon to be Sony-owned. But no matter how much Japan gets under our skin, we'll still have mom and apple pie. On second thought, make that just mom. A Japanese apple called the Fuji is cropping up in orchards the way Hondas did on U.S. roads.

Figure 4: Summaries obtained from wsj_1128.

better results than MCP, KP and LEAD, although some of the comparisons are not significant. In particular, TKP(G) achieved the highest ROUGE scores on all measures. On ROUGE-1 Recall, TKP(G) significantly outperformed Marcu(G), Marcu(H), KP and LEAD. These results support the effectiveness of our method that utilizes the discourse structure. Comparing TKP(H) with Marcu(H), the former achieved higher scores with statistical significance on ROUGE-1. In addition, Marcu(H) was outperformed by MCP, KP and LEAD. The results confirm the effectiveness of our summarization model and trimming proposal for DEP-DT. Moreover, the difference between TKP(G) and TKP(H) was smaller than that between Marcu(G) and Marcu(H). This implies that our method is more robust against discourse parser error than Marcu's method.

Figure 4 shows the example summaries generated by TKP(G), Marcu(G), MCP and LEAD, respectively for an article, wsj_1128. Since TKP(G) and Marcu(G) utilize a discourse tree, the summary generated by TKP(G) is similar to that generated by Marcu(G) but it is different from those generated by MCP and LEAD.

5 Conclusion

This paper proposed rules for transforming an RST-DT to a DEP-DT to obtain the parent-child relationships between EDUs. We treated a single document summarization method as a *Tree Knapsack Problem*, *i.e.* the summarizer selects the best rooted subtree from a DEP-DT. To demonstrate the effectiveness of our method, we conducted an experimental evaluation using 30 documents selected from the RST Discourse Treebank Corpus. The results showed that our method achieved the highest ROUGE-1,2 scores.

References

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)*, 57(1):289–300.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proc. of the SIGDIAL01*, pages 1–10.

Geon Cho and Dong X Shaw. 1997. A depth-first dynamic programming algorithm for the tree knap-

sack problem. *INFORMS Journal on Computing*, 9(4):431–438.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proc. of the 40th ACL*, pages 449–456.

David duVerle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proc. of the Joint Conference of the 47th ACL and 4th IJCNLP*, pages 665–673.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence extraction. In *Proc. of the 20th COLING*, pages 397–403.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proc. of the 5th International Natural Language Generation Conference (INLG)*, pages 25–32.

Hugo Hernault, Helmut Prendinger, David A duVerle, and Mitsuru Ishizuka. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proc. of Workshop on Text Summarization Branches Out*, pages 74–81.

J. A. Lukes. 1974. Efficient algorithm for the partitioning of trees. *IBM Journal of Research and Development*, 18(3):217–224.

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *Proc. of the 6th Workshop on Very Large Corpora*, pages 206–215.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proc. of the 29th ECIR*, pages 557–564.

Ani Nenkova and Kathaleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233.

Natthawut Samphaiboon and Takeo Yamada. 2000. Heuristic and exact algorithms for the precedence-constrained knapsack problem. *Journal of Optimization Theory and Applications*, 105(3):659–676.

Hiroya Takamura and Manabu Okumura. 2009a. Text summarization model based on maximum coverage problem and its variant. In *Proc. of the 12th EACL*, pages 781–789.

Hiroya Takamura and Manabu Okumura. 2009b. Text summarization model based on the budgeted median problem. In *Proceedings of the 18th CIKM*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

William Charles, Mann and Sandra Annear, Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

A Hierarchical Entity-based Approach to Structuralize User Generated Content in Social Media: A Case of Yahoo! Answers

Baichuan Li[1,2]*, **Jing Liu**[3]*, **Chin-Yew Lin**[4], **Irwin King**[1,2], **and Michael R. Lyu**[1,2]
[1]Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
[2]Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[3]Harbin Institute of Technology, Harbin 150001, P.R. China
[4]Microsoft Research Asia, Beijing 100080, P.R. China
bcli@cse.cuhk.edu.hk jliu@ir.hit.edu.cn cyl@microsoft.com
{king,lyu}@cse.cuhk.edu.hk

Abstract

Social media like forums and microblogs have accumulated a huge amount of user generated content (UGC) containing human knowledge. Currently, most of UGC is listed as a whole or in pre-defined categories. This "list-based" approach is simple, but hinders users from browsing and learning knowledge of certain topics effectively. To address this problem, we propose a hierarchical entity-based approach for structuralizing UGC in social media. By using a large-scale entity repository, we design a three-step framework to organize UGC in a novel hierarchical structure called "cluster entity tree (CET)". With Yahoo! Answers as a test case, we conduct experiments and the results show the effectiveness of our framework in constructing CET. We further evaluate the performance of CET on UGC organization in both user and system aspects. From a user aspect, our user study demonstrates that, with CET-based structure, users perform significantly better in knowledge learning than using traditional list-based approach. From a system aspect, CET substantially boosts the performance of two information retrieval models (i.e., vector space model and query likelihood language model).

1 Introduction

With the development of Web 2.0, social media websites—such as online forums, blogs, microblogs, social networks, and community question answering (CQA) portals—have become the mainstream of web, where users create, share, and exchange information with each other. As a result, more and more UGC is accumulated, with social media websites retaining a huge amount of human knowledge and user experience. At present, most of UGC is organized in a list structure with extra information (e.g., category hierarchies in online forums), or without any other information.

Table 1: Sample questions about *Edinburgh*

1. Where can i buy a **hamburger** in **Edinburgh**?
2. Where can I get a **shawarma** in **Edinburgh**?
3. How long does it take to drive between **Glasgow** and **Edinburgh**?
4. Whats the difference between **Glasgow** and **Edinburgh**?
5. Good **hotels** in **London** and **Edinburgh**?
6. Looking for nice , clean cheap **hotel** in **Edinburgh**?
7. Does anyone know of a reasonably cheap **hotel** in **Edinburgh** that is near to **Niddry Street South** ?
8. Who can recommend a affordable **hotel** in **Edinburgh City Center**?

This "list-of-content" (list-based approach) is simple and straightforward, but ineffective for browsing and knowledge learning. Consider the following case: a user wants to spend his vacation in Edinburgh. He visits a CQA website to explore which aspects are mostly asked. In this scenario, he may browse some relevant categories like "Travel:United Kingdom:Edinburgh" to get useful information. He may also issue a query like "travel in Edinburgh" to search relevant questions. However, both the browsing and the searching give the user a list of relevant contents (e.g., questions shown in Table 1), not the direct knowledge. Thus, the user has to read these contents, understand them, classify them into various topics, and gain valuable

*This work was done when the first two authors were on internship at MSRA.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1521–1532,
Seattle, Washington, USA, 18-21 October 2013.

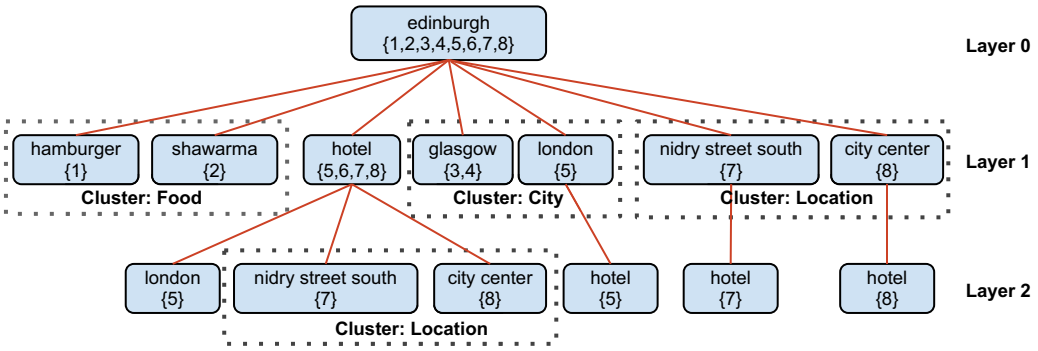


Figure 1: An CET constructed from questions about *Edinburgh*

knowledge himself. Obviously, it is ineffective and time-consuming.

The above problem calls for a new approach to structuralize UGC in social media, which facilitates users to seek knowledge (e.g., travel information about Edinburgh) more effectively. Traditionally, we can utilize topic models (Blei et al., 2003) or social tagging to structuralize UGC. However, for topic models, it is not easy to control the granularity of topics, and it is hard for users to interpret a topic only based on the multinomial distribution (Mei et al., 2007). For social tagging, it is not applicable in many sites and has sparsity problem (Shepitsen et al., 2008). Thus, both topic models and social tagging are not suitable for structuralizing UGC in social media.

In this paper, we propose a novel hierarchical entity-based approach, i.e., "cluster entity tree" or CET, to structuralize UGC in social media by leveraging an existing large-scale entity repository. Figure 1 shows how CET structuralizes UGC in Table 1. In this CET, each node contains one (named) entity and a set of question IDs. With "edinburgh" as the root entity, layer 1 includes all entities that co-occur with "edinburgh". Similarly, entities on layer 2 co-occur with their parent entities on layer 1 and the root entity "edinburgh". For example, "city center" co-occurs with "hotel" and "edinburgh" in Question 8. Deeper layers provide more specific topics about different aspects of Edinburgh (e.g., Edinburgh's hotels). As shown in Fig. 1, the corresponding question IDs are also attached in each node. In addition, entities which share the same parent are clustered to different groups (see dashed rectangles in Fig. 1)[1]. Through this hierarchical structure, we can easily browse corresponding questions and answers, and learn knowledge about Edinburgh more effectively, such as food and location. Moreover, since similar entities (and corresponding questions) are grouped in each layer, it also helps the system manage similar contents.

By utilizing a large-scale entity repository, CET avoids the granularity, interpretation, and sparsity problems. Entity repositories like Freebase[2] provide a large number of named entities across various pre-defined topics, which avoid the granularity and sparsity problems. In addition, they usually give descriptions of entities, which prevent the interpretation problem. Therefore, CET is more suitable for structuralizing UGC.

In this paper, we propose a three-step framework to construct CETs.

1. Entity extraction. In this step, we extract entities from documents using an existing entity repository.
2. Tree construction. we build the co-occurrence relationship between any two entities and construct hierarchical "entity trees (ETs)".
3. Hierarchical entity clustering. In an ET, some entities are more similar than other entities which share the same parents. Therefore, on each layer of the ET we cluster entities with the same parents (e.g., "london", "nidry street south", and "city center" on layer 2 of Fig. 1) and finally construct a CET.

We select Yahoo! Answers as a test case to evaluate 1) the performance of our framework for con-

[1]Single-node clusters are not surrounded by rectangles, like "hotel{5}" on layer 2.

[2]http://www.freebase.com/

structing CET and 2) the effectiveness of CET in UGC structuralization. Yahoo! Answers is a popular CQA portal, where users post questions and provide answers in different categories. Experimental results demonstrate the good performance of our framework for constructing CET. We further evaluate the effectiveness of CET in structuralizing UGC from both user and system aspects. From a user aspect, our user study show that, with CET-based organization, users perform significantly better in knowledge learning than using list-based approach. From a system aspect, CET boosts systems' information retrieval substantially: the mean reciprocal rank (MRR) of vector space model (VSM) and query likelihood language model (QLLM) are improved by 9.3% and 8.2%, respectively.

To summarize, our contributions are three-fold:

1. We propose a novel hierarchical entity-based approach to structuralize UGC in social media. To our knowledge, we are the first to utilize entities to structuralize UGC for content browsing and knowledge learning at a large scale;
2. We present a three-step framework to construct CETs and show its effectiveness from empirical results;
3. We demonstrate the significant advantages of our approach for both users and systems in knowledge learning and retrieval.

The paper proceeds as follows. We present related work in Section 2. We detail our framework to construct CETs and show empirical results in Section 3. Section 4 and Section 5 evaluate the effectiveness of CET on knowledge organization from user and system aspects respectively. We conclude the paper in Section 6.

2 Related Work

UGC organization in social media. Most UGC in social media is unstructured, or organized in a predefined category hierarchy. These categories give shallow semantics of UGC, and boosts the performance of information retrieval (Cao et al., 2008; Duan et al., 2008; Cao et al., 2012) and recommendation (Guo et al., 2008; Li et al., 2011). With new content kept adding into a hierarchy, we need to maintain category hierarchy (Yuan et al., 2012) to make content within the same category more topically cohesive.

Apart from category hierarchy, UGC can also be organized by topic models and tags. Topic models, such as latent Dirichlet allocation (LDA) (Blei et al., 2003), are widely applied in document clustering and classification, However, it is not trivial to control the granularity of topics (Chen et al., 2011). What's more, it is generally difficult to understand a topic only from the multinomial distribution (Mei et al., 2007). Social tagging provides an alternative approach in document organization (Gupta et al., 2010). In some UGC websites like stackoverflow[3], users usually add tags to describe their contents. However, tags are not widely applicable and tags are usually sparse (Shepitsen et al., 2008). Our hierarchical entity-based approach prevents these problems by employing a large-scale entity repository. As the entity repository provides a unified set of entities across various of pre-defined topics, and gives descriptions of entities, CET avoids the granularity, interpretation and sparsity issues.

The work in (Zhu et al., 2013), which automatically generates and updates topic terms to organize UGC, is mostly related to our work. In this paper, given a root topic, subtopics and lower-level topics are extracted from UGC, which form a hierarchical structure to organize corresponding UGC. However, in (Zhu et al., 2013) more external sources are utilized to identify subtopics. In addition, relationships among subtopics which under the same parent are not investigated. The metro maps proposed in (Shahaf et al., 2013) are also related to our work. Different from (Shahaf et al., 2013), we employ a large-scale entity repository to extract more meaningful and interpretable key terms (entities), which make each subtopic much easier to understand.

Entity extraction. In our framework, we leverage an entity repository to extract named entities from UGC. A common approach is to utilize a Named Entity Recognition (NER) system like Stanford NER (Finkel et al., 2005), which recognizes the names of things (e.g., person and product names) from texts. For cross-domain NER, (Rüd et al., 2011) employed search engines. For short-text NER, (Liu et al., 2012) proposed a graphical model. However, most of above systems are restricted from

[3]http://stackoverflow.com/

producing labels for a few entity classes. To address the problem, (Ling and Weld, 2012) defined a fine-grained set of 112 tags based on Freebase for entity extraction. However, this approach still faces the "low-recall" problem in the real world. Our approach, which leverages a large-scale entity repository, addresses this issue.

Entity-based document classification and retrieval. Entity repository has been employed in other research areas, like document classification and retrieval. (Schonhofen, 2006) utilized Wikipedia to classify documents. (Yerva et al., 2012a) proposed an entity-based classification for tweets. In addition, entity-based retrieval models were proposed and applied in both QA archives (Singh, 2012a) and tweets (Yerva et al., 2012b). Besides, (Singh, 2012b) proposed an entity-based translation language model and demonstrated that it outperformed classical translation language model in question retrieval. However, to the best of our knowledge, no previous study leverages entities to organize UGC in social media.

3 CET Construction

In this section, we formulate the framework to construct CET and show the empirical results. Firstly, we provide the definitions of the entity repository and CET.

Definition 3.1 (Entity Repository) *Let $ER = \{R, g\}$ be an entity repository, where R is a set of named entities and $g : R \times R$ is a mapping function that defines the similarity of any two entities.*

Note that we do not require a hierarchical structure in an ER (like Freebase); only a similarity function is needed.

Definition 3.2 (Cluster Entity Tree) *Let D be a set of documents, $ER = \{R, g\}$ be an entity repository, e be an entity, a cluster entity tree $CET_e = (v_e, V, E, C)$ is defined as a tree structure, with the root node v_e, node set V, edge set E, and cluster set C. Each node $v_s \in V$ on CET_e includes an entity extracted from the set of documents $D_e \in D$ containing e, and a list $L(s)$ which stores the indexes of documents containing entity s and its superior entities. If v_s is v_t's parent node, entity t must co-occur with s and s's all superior entities at least once in the same document. Each element of C (one cluster) includes a set of nodes which share the same parent node, and the entities within a cluster are more similar to each other than the entities in other clusters.*

3.1 Framework

This section shows our three-step framework for constructing CET: entity extraction (Section 3.2.1), tree construction (Section 3.2.2), and hierarchical entity clustering (Section 3.2.3).

3.1.1 Entity Extraction

We adopt a simple entity repository based approach to extract entities, which address the "low-recall" problem for traditional NER methods (details are given in Section 3.3.2). This approach involves two phases: candidate entity extraction and entropy-based filtering.

Candidate entity extraction. We employ the Stanford Parser[4] to parse each document to a parse tree. Then, we extract all noun phrases, preprocess them (including stemming), and extract the noun phrases which are included in our entity repository. In our experiments, we adopt a large-scale enterprise entity repository (anonymized for blind reviews).

Entropy-based filtering. The candidate entities generated from the last step may contain many false examples, which are not relevant to the main semantics of documents, like "we", "how do i", etc. To filter them, we propose an entropy-based method. Given a document with a category label (or tags, which are available in most UGC sites), we get the distributions of each candidate entity over all top categories. The entropy of a candidate entity e_i is calculated as follows:

$$Entropy(e_i) = -\sum_{c=1}^{|C|} P_c(e_i) log P_c(e_i), \quad (1)$$

where $|C|$ is the number of top categories and $P_c(e_i)$ is the number of e_i in category c divided by all number of candidate entities in that category.

Top-ranked entities are general terms among categories. We set a threshold α and remove all candidate entities with entropy larger than α. The setting of α is a tradeoff: higher values will introduce more noise, while smaller values will lead to decreased

[4] http://nlp.stanford.edu/software/lex-parser.shtml

recall. In our experiments, we empirically set α as 1.5 since it provides the most satisfying results.

3.1.2 Tree Construction

Given an entity (e.g., "edinburgh"), we first search documents containing this entity and make the entity together with document ids as the root node. Then, from searched documents we find all entities that co-occur with the root entity. These entities and corresponding document ids form layer-1 nodes of the entity tree (see the example in Fig. 1). Afterwards, for each entity in layer-1 nodes, we search entities that co-occur with it and its superiors, combine them and corresponding document ids as new nodes, and put these new nodes under current node, which form layer-2 nodes. Iteratively, we construct the entity tree with the given entity as the root.

3.1.3 Hierarchical Entity Clustering

Under the same parent, some entities[5] may share similar topics. Therefore, the final step is to hierarchically cluster entities with the same parents at different layers of entity trees. This step not only facilitates knowledge learning but also reduces the width of a tree. In this paper, we follow the work in (Hu et al., 2012) and employ an agglomerative clustering algorithm for two reasons: 1) it is easy to implement and its time complexity is $O(N^2)$; 2) there is no need to set the number of clusters. Although the clustering results may be influenced by instance order, our empirical results demonstrate its effectiveness. Any other advanced algorithms like spectral clustering (Ng et al., 2002) can also be applied, but that is not the emphasis of this paper.

Algorithm. For a set of entities with the same parent, the agglomerative clustering algorithm works as follows:

1. Select one entity and create a new cluster which contains the entity;
2. Select the next entity e_i, create an empty candidate list, calculate the similarity between the entity and all existing clusters. Three strategies are employed[6]:
   - AC-MAX: If the similarity between entity e_i and entity e_j in one of the clusters (the first one) is larger than threshold θ_{max}, we put the cluster index and corresponding similarity in the candidate list.
   - AC-MIN: If the similarity between entity e_i and any entity e_j in one of the clusters is larger than threshold θ_{min}, we put the cluster index and corresponding similarity in the candidate list.
   - AC-AVG: If the mean similarity between entity e_i and any entity e_j in one of the clusters is larger than threshold θ_{avg}, we put the cluster index and corresponding similarity in the candidate list.
3. If the candidate list is not empty, put e_i in the cluster with highest similarly.
4. If the candidate list is empty, a new cluster with e_i as the element will be created.
5. Stop when all entities are clustered.

Similarity Function. In our entity repository, the similarity between two entities is computed using the approach in (Shi et al., 2010), which estimates the similarity of two terms according to their *first-order* and *second-order* co-occurrences. For example, "such as NP, NP" is a good pattern for detecting similar entities using *first-order* co-occurrences. In addition, if two entities usually co-occur with a third entity (*second-order* co-occurrence), these two entities are likely to be similar. To construct similarity functions, pattern-based approaches (Ohshima et al., 2006; Zhang et al., 2009) utilize *first-order* co-occurrences while distributional similarity approaches (Pasca et al., 2006; Pennacchiotti and Pantel, 2009) employ *second-order* co-occurrences. In the following, we briefly introduce the pattern-based approach (PB) and the distributional similarity approach (DS) in (Shi et al., 2010).

PB. Some well-designed patterns are leveraged to extract similar entities from a huge repository of webpages. The set of terms extracted by applying a pattern one time is called a raw semantic class (RASC). Given two entities t_a and t_b, PB calculates their similarity based on the number of RASCs containing both of them (Zhang et al., 2009):

$$Sim(t_a, t_b) = \log(1 + \sum_{i=1}^{r_{ab}} P_{ab_i})) \cdot \sqrt{idf(t_a) \cdot idf(t_b)}, \quad (2)$$

where $idf(t_a) = \log(1 + \frac{N}{C(t_a)})$, P_{ab_i} is a pattern which can generate RASC(s) containing both term

[5]Here we use the entity to represent the node.

[6]We modify the clustering algorithm in (Hu et al., 2012) slightly to assign a unique cluster for each entity.

t_a and term t_b, r_{ab} is the total number of such patterns, N is the total number of RASCs, and $C(t_a)$ is the number of RASCs containing t_a. The above similarity is normalized using the following function:

$$Sim_{PB}(t_a, t_b) == \frac{\log Sim(t_a, t_b)}{2 \log Sim(t_a, t_a)} + \frac{\log Sim(t_a, t_b)}{2 \log Sim(t_b, t_b)}. \quad (3)$$

DS. DS approach assumes that terms appearing in similar contexts tend to be similar. In this approach, a term is represented by a feature vector, with each feature corresponding to a context in which the term appears. The similarity between two terms is computed as the similarity between their corresponding feature vectors. Jaccard similarity is employed to estimate the similarity between two terms. Suppose the feature vector of t_a and t_b are **x** and **y** respectively,

$$Sim_{DS}(t_a, t_b) = \frac{\sum_i \min(x_i, y_i)}{\sum_i (x_i) + \sum_i (y_i) - \sum_i \min(x_i, y_i)}. \quad (4)$$

Shi et al. (Shi et al., 2010) found that PB performed better when dealing with proper nouns; while DS was relatively good at estimating similarity of other types of entities. The similarity function in our ER follows the suggestion of (Shi et al., 2010): if at least one entity is proper noun, PB is employed; otherwise DS is used.

3.2 Experiments

3.2.1 Setup

We evaluate the performance of our framework employing questions from Yahoo! Answers. 54.7 million questions are crawled from all 26 top categories in Yahoo! Answers, which consist of question titles and corresponding categories. From these questions, we construct the following two test sets for evaluating entity extraction and entity clustering:

Set EE. This set is employed to evaluate the performance of entity extraction. It contains 520 randomly sampled questions, 20 from each top category. One author is asked to label entities for each question.

Set EC. This set is constructed to automatically evaluate hierarchical entity clustering and select the best clustering strategy. The construction process is as follows. First, we map the four top categories of Yahoo! Answers to some categories of Freebase manually, as shown in Table 2. Second, from questions at each top category of Yahoo! Answers, we extract entities which appear exactly once in the corresponding Freebase categories. For instance, if an entity is extracted from questions in the category *Computers & Internet*, and it appears two or more times in *Computer* and *Internet* categories in Freebase, it will be filtered. Therefore, each entity is attached with a unique Freebase category label (i.e., the ground truth for clustering). Questions containing at least two entities are selected for Set EC. Table 3 reports the statistics. Intuitively, entities with a same Freebase category label should be in one cluster.

Table 2: Category mapping between Yahoo! Answers and FreeBase

Yahoo! Answers	FreeBase
Cars & Transportation	Aviation, Transportation, Boats Spaceflight, Automotive, Bicycles, Rail
Computers & Internet	Computer, Internet
Sports	Soccer, Olympics,Sports, American football, Baseball,Basketball,Ice Hockey,Martial Arts, Cricket,Tennis,Boxing,Skiing
Travel	Travel, Location, Transportation

Table 3: Number of questions and entities in Set EC

Category	Number of Questions	Number of Entities
Cars & Transportation	1,220,427	3,267,596
Computers & Internet	2,912,280	7,324,655
Sports	2,363,758	6,230,868
Travel	1,347,801	3,728,286

Note that Set EC only covers a small set of real entities and clustering on Set EC is partial clustering. However, it leverages Freebase labels and avoids manual labeling, which is time-consuming. Furthermore, partial clustering results are enough for evaluating different strategies' performance and choosing the best strategy.

Following the common practice, we evaluate entity extraction using precision, recall, and $F1$ score. For evaluating entity clustering results, we adopt B-cubed metrics. As reported in (Amigó et al., 2009), B-cubed metrics are more suitable than traditional metrics, such as NMI and purity.

Table 4: Entity extraction for various methods

Method	Precision	Recall	F1
Standord NER	0.750	0.155	0.257
FIGER	0.763	0.154	0.256
Freebase	0.644	0.595	0.619
Our	0.647	0.809	0.719

Table 5: Clustering results using AC-MAX (θ_{max}=0.1)

Level	*Travel*				*Cars & Transportation*				*Computer & Internet*				*Sports*			
	Count	P	R	F1	Count	P	R	F1	Count	P	R	F1	Count	P	R	F1
1	748	0.972	0.653	0.743	1281	0.948	0.868	0.897	3064	0.913	0.664	0.743	890	0.941	0.883	0.901
2	200	0.974	0.730	0.798	1202	0.989	0.956	0.965	11344	0.961	0.842	0.879	636	0.978	0.964	0.963
3	120	1.000	0.833	0.890	858	1.000	0.981	0.988	8184	0.978	0.899	0.920	492	0.965	0.882	0.899
4	NA	NA	NA	NA	1776	1.000	0.980	0.986	3648	0.990	0.908	0.934	1080	0.978	0.844	0.881
5	NA	NA	NA	NA	NA	NA	NA	NA	2520	1.000	0.952	0.968	NA	NA	NA	NA
Total	1068	0.976	0.688	0.770	5117	0.984	0.946	0.959	28760	0.968	0.857	0.891	3098	0.965	0.886	0.907

3.2.2 Results: Entity Extraction

Two ERs (i.e., ours and Freebase) are employed in entity extraction for comparison. In addition, we compare our approach with Stanford NER (Finkel et al., 2005) and fine-grained entity recognition (FIGER) (Ling and Weld, 2012). Table 4 reports the results of different methods. We can find that Stanford NER and FIGER get a relatively high precision in extracting entities. However, their recalls are very low and only about 15% of entities are recognized. With the help of entity repositories, recall is significantly improved with a small decrease of precision. Therefore, the $F1$s of entity-based approaches are much higher. This observation shows the great advantage of utilizing an entity repositories in entity extraction and the effectiveness of our approach. As our ER performs better than Freebase, we adopt it as our entity repository in the following evaluations.

3.2.3 Results: Hierarchical Entity Clustering

Table 5 reports the count of clusters, B-Cubed Precision, Recall, and F1 for different layers of clustering across four categories using **AC-MAX**. In our experiments, **AC-MAX** performed better than **AC-MIN** and **AC-AVG**. Due to space limitations, we only report the results of **AC-MAX** here. For **AC-MAX**, we changed the settings of θ from 0.01 to 0.9, and the best performance was achieved when θ_{max} was set at around 0.1. From Table 5 we can find that, although **AC-MAX**'s accuracy varies across categories (e.g., the $F1$ of *Transportation* is much higher than that of *Travel*), it performs well in general. Thus, we adopt **AC-MAX** with $\theta = 0.1$ for hierarchical entity clustering.

4 User Study

In this section, we investigate the influence of CET on users' content browsing and knowledge learning from a user study. In the study, we design 24 tasks in four popular Yahoo! Answers categories (see Table 2). For each category, we design three ***knowledge-learning*** tasks and three ***question-search*** tasks, as shown in Table A.1 in the supplementary material. A ***knowledge-learning*** task asks for some knowledge about a main entity from question texts. For instance, "find the games running on macbook pro" requires game names as the answer, where the main entity is "macbook pro". A ***question-search*** task, however, asks users to find similar questions to the question in the task. For example, "questions about who will win the MVP in NBA this year" asks for finding similar questions, and filling their question IDs as the answer. For each task, we give some suggested keywords (entities) to facilitate information gathering.

To evaluate user experience, we ask participants to fill out a questionnaire after each task. Following the work in (Kato et al., 2012), we collect information from 5 aspects: familiarity, easiness, satisfaction, adequate time, and helpfulness. A 5-point Likert scale is designed for each questionnaire. "5" means the participant totally agrees while "1" means the participant totally disagrees.

4.1 Setup

Programs. We develop two programs in our user study. One is CET-based, and the other is traditional list-based[7]. The list-based program searches questions by utilizing Apache Lucene[8]. The standard analyzer and the default search algorithm are adopted. For each query, top 200 most relevant questions are retrieved.

Data. We extract 70,195 questions which contain at least one of the 24 main entities (see Table A.1)

[7]The interface of CET-based program is provided in the supplementary material (see Fig. A.1). The interface of list-based program is similar, but the CET display area is replaced by a flat-ranked list.

[8]http://lucene.apache.org/core/

Table 6: User study results

	Knowledge-learning Tasks		*Question-search* Tasks	
	CET-based	List-based	CET-based	List-based
# Queries	2.99	4.47	2.56	3.38
# Answers	8.32	6.06	10.60	10.92
Precision	0.38	0.19	0.40	0.44
Time	136.44	121.87	103.71	87.75

Table 7: Questionnaire results

	Knowledge-learning Tasks		*Question-search* Tasks	
	CET-based	List-based	CET-based	List-based
Familiarity	3.18	3.22	3.07	3.28
Easiness	3.64	3.66	4.10	4.06
Satisfaction	3.70	2.94	3.86	3.44
Enough Time	3.87	3.83	4.44	4.54
Helpfulness	4.16	3.03	4.31	3.71

in the four categories. For each question, we extract the entities with the help of our entity repository. For each main entity, we build the corresponding CET from all extracted questions.

Participants. Sixteen volunteers are invited in the user study. They are first briefly informed of the research design and taught how to use two programs. To familiarize the participants with our programs promptly, we provide demonstrations using sample entities. Each volunteer is asked to finish 12 tasks (6 ***knowledge-learning*** tasks and 6 ***question-search*** tasks) using the CET-based program and 12 other tasks using the list-based program in random order. Thus, each task is finished by exactly 8 different participants using each program.

4.2 Results and Discussions

Table 6 reports the user study results, where we give the statistics for users' performance with the two programs. We evaluate from the number of queries issued, number of answers found, the precision of answers, and query time for each task. As our 24 tasks contain both knowledge-learning tasks and question-search tasks, we report their results separately. Z-tests are employed for significance tests.

From Table 6, we observe that more queries are issued in the knowledge-learning tasks than in the question-search tasks using both programs. However, the CET-based program reduces the number of queries substantially in both tasks. Because the CET-based program provides a series of clustered entities, it helpes users further refine queries through clicking on entities rather than reconstructing a new query. However, the list-based program only lists relevant questions, and users have to issue new queries according to returned questions.

By using the CET-based program, volunteers find more answers in knowledge-learning tasks ($z = 1.69, p < 0.05$). The reason is that the CET-based program clusters similar results in the same group, and if the user finds one answer she can easily get more answers. On the contrary, the list-based program returns a list of questions, and users need to find answers question-by-question. For question-search tasks, users of the list-based program find more answers, but the difference is not significant ($z = 0.19$). As the list-based program returns similar questions as top-ranked results, users are able to fill in answers easily. For CET-based program users, they have to find corresponding key entities in the CETs first. Therefore, they spend more time (the fourth row in Table 6) finding entities and less time filling answers. It is worth noting that our GUI prototype for CET is non-optimal, and users' searching time on CET-based program can be further reduced with better user interface.

The precision of answers from CET-based program users is twice of that from list-based program users ($z = 4.15, p < 0.0001$) in knowledge-learning tasks, which demonstrates the advantage of CET in helping knowledge-learning. For question-search tasks, CET-based program users perform slightly worse than list-based program users, but the difference is not significant ($z = 0.48$). Since users of the CET-based program spend more time finding entities, they have limited time to check the answers.

In both tasks, users spend more time on the CET-based program. According to users' post-user-study feedbacks, a few volunteers reported that they sometimes spent a considerable amount of time on finding entities from CETs; however, one positive observation is that most users find "the entity-based interface" very interesting, which stimulates them to spend more time on exploring answers.

The questionnaires reveal more about user experience on these two programs (see Table 7). Users' responses to task familiarity and easiness are similar. However, users of entity-based interface are more satisfied in both knowledge-learning tasks ($z = 3.98, p < 0.0001$) and question-search tasks ($z =$

1.38), and they feel that entity-based interface is more helpful in finding answers for both knowledge-learning tasks ($z = 6.47, p < 0.0001$) and question-search tasks ($z = 2.55, p < 0.01$). These promising observations show that CET helps knowledge learning greatly through structuralizing content.

5 CET-based Question Re-Ranking

In this section, we show that CET also helps systems to better retrieve information through re-ranking. In the following, we continue to use Yahoo! Answers as a test case.

Algorithm 1 CET-based search results re-ranking

Input: query q, question collection Q, a ranked list of k relevant questions $Q_q = \{q_1, q_2, ..., q_k\}$ to q, an entity repository ER, an empty list Θ.

Output: A new ranked list of questions.

1: Extract entities from each question of Q and construct a entity co-occurrence graph;
2: Get the PageRank score of each entity;
3: **if** There is no entity e in q **then**
4: return Q_q;
5: **else**
6: Identify the key entity e from q which has the highest PageRank score;
7: Construct the CET cet_e from Q based on ER;
8: **for** each question q_i **do**
9: For all entities in q_i, build a entity chain C in descending order of PageRank scores;
10: From C extract the first entity $\hat{e}$ that is not similar to e;
11: **if** $\hat{e}$ exists **then**
12: Put q_i in the corresponding cluster of nodes;
13: **else**
14: Put q_i in Θ;
15: **end if**
16: **end for**
17: **end if**
18: Rank all clusters on cet_e according to their first elements' original ranking;
19: Output the final ranking cluster by cluster and append the questions in Θ at the last.

Intuitively, questions sharing similar topics should be ranked similarly. However, traditional question retrieval models (Cao et al., 2010) such as QLLM and VSM do not capture key semantics and give more weights for entity terms. CET provides a feasible way to address this issue. By utilizing CET, entities are given more weight while trivial words are not. In addition, through clustering questions with similar topics, those questions which are ranked lower will be brought higher by their top-ranked neighbors.

Table 8: Re-ranking results for VSM and QLLM (* means that $p < 0.05$ in students' t-test)

	VSM	Re-ranking	QLLM	Re-ranking
MRR	0.3838	**0.4195*** (9.30%)	0.3593	**0.3889*** (8.24%)
MAP	0.3376	**0.3558*** (5.39%)	0.3326	**0.3479*** (4.60%)
Prec@1	0.2500	**0.3125*** (25.00%)	0.2438	**0.2688*** (10.25%)

Algorithm 1 illustrates the re-ranking algorithm in detail. We first extract entities from each question of the whole question collection, and construct a entity co-occurrence graph (Line 1). Then, we calculate the PageRank score of each entity (Line 2). Line 3-5 check whether q contains at least one entity. If the answer is no, we return the original ranking. Otherwise, we identify the key entity in q (Line 6) and construct the CET cet_e whose root entity is e (Line 7). Line 8-16 iteratively put questions in corresponding clusters of cet_e. In Line 8, we first build an entity chain for question q_i, in which entities of q_i are ranked according to their PageRank scores. Afterwards, the first entity $\hat{e}$, which is not similar to e (the similarity is calculated in Section 4.2.1 and the threshold of similarity is set to 0.1), is picked up as the main aspect of e, and q_i is grouped into the corresponding cluster on cet_e (Line 7-8). If $\hat{e}$ does not exist, we put q_i in a new cluster (Line 13-14). Then, we rank all clusters according to their first elements' original rankings (Line 18) and output the final re-ranked list (Line 19).

We perform our re-ranking on 160 randomly selected questions from *Computers & Internet* and *Travel* categories of our data set[9]. Each category contains 80 questions. All other questions in these two categories constitute the question collection Q. For each question, we utilize the VSM and QLLM[10] respectively to get the top 15 most relevant questions (excluding itself). The correct ranking is manually labeled and checked by two annotators. We firstly employed the VSM and QLLM respectively to retrieve the top 15 results and then obtained manual judgments. Given a retrieved question by VSM

[9]These 160 questions are not used for constructing the entity co-occurrence graph.

[10]Following (Zhai and Lafferty, 2004), we set λ to 0.2.

or QLLM, two assessors are asked to label it with "relevant" or "irrelevant". If their annotations are opposite, the third assessor is involved to determine the final label.

We re-rank these questions using Algorithm 1. Table 8 shows the results of MRR, mean average precision (MAP), and Precision@1. We can see that CET-based re-ranking improves the performance of standard retrieval models substantially. For VSM, our re-ranking boosts the MRR and MAP by 9.3% and 5.4%, respectively. It is worth noting that our re-ranking improves Prec@1 significantly: from 0.25 to 0.31. The reason is that traditional methods may give relatively low weights to the key terms (entities), while CET-based re-ranking addresses the problem. QLLM and re-ranking report similar results. Figures 2 and 3 illustrate the performance of various approaches across categories. We find that our re-ranking is neither category-biased nor algorithm-biased, yet it performs better than original models on both categories. The above results demonstrate that, by utilizing the hierarchical entity-based approach, CET greatly improves the retrieval performance of these two standard models.

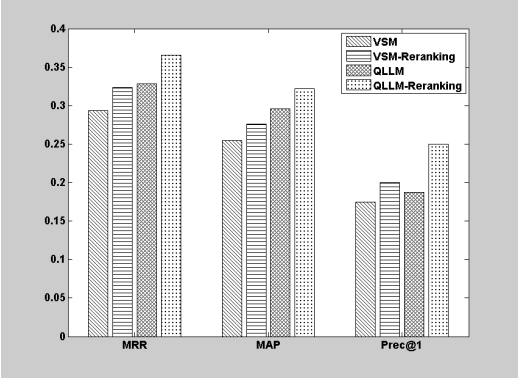


Figure 2: Re-ranking results of *Computer & Internet*

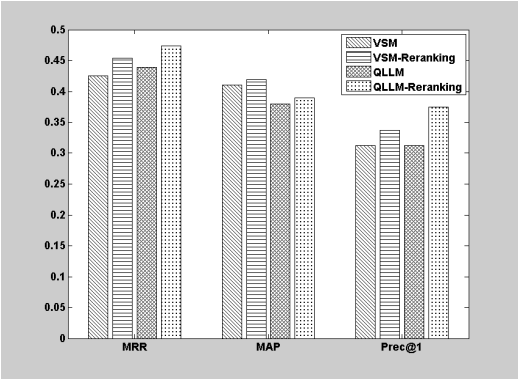


Figure 3: Re-ranking results of *Travel*

6 Conclusion and Future Work

Traditional list-based organization of UGC in social media is not effective for content browsing and knowledge learning due to large volume of documents. To address this problem, we propose a novel hierarchical entity-based approach to structuralize UGC in social media. By using a large-scale entity repository, we construct a three-step framework to organize knowledge in "cluster entity trees". Experimental results show the effectiveness of the framework in constructing CET. We further evaluate the performance of CET on knowledge organization from both user and system aspects. Our user study demonstrates that, with CET-based organization, users perform significantly better in knowledge learning than using list-based approach. In addition, CET boosts systems' content search performance substantially through re-ranking.

To our best knowledge, this work is the first attempt to utilize entities to structuralize UGC in social media, and there are some limitations to be improved in our future work. First, we employ Yahoo! Answers as our test data, in which questions (documents) are usually short. We observe that nearly 92% of all 54.7 million questions contain 1-4 entities, which means the depth of CETs are usually not so deep. However, long documents, such as Blog posts, will lead to deep CETs and hinder users' knowledge learning. Second, our current entity extraction focuses on named entities instead of canonical entities. In the future, we plan to employ document summarization techniques to shorten the depth of CETs. We also aim to incorporate semantic analysis and normalize named entities to canonical entities, which make CET more suitable for practical use.

Acknowledgments

The work described in this paper was fully supported by the Shenzhen Major Basic Research Program (Project No. JC201104220300A) and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK413212, CUHK415212).

References

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.*, 12(4):461–486, August.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. Recommending questions using the mdl-based tree cut model. In *Proc. of WWW*, WWW '08, pages 81–90.

Xin Cao, Gao Cong, Bin Cui, and Christian S. Jensen. 2010. A generalized framework of exploring category information for question retrieval in community question answer archives. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 201–210, New York, NY, USA. ACM.

Xin Cao, Gao Cong, Bin Cui, Christian S. Jensen, and Quan Yuan. 2012. Approaches to exploring category information for question retrieval in community question-answer archives. *ACM Trans. Inf. Syst.*, 30(2):7:1–7:38, May.

Mengen Chen, Xiaoming Jin, and Dou Shen. 2011. Short text classification improved by learning multi-granularity topics. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 1776–1781. AAAI Press.

Huizhong Duan, Yunbo Cao, Chin yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *In Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Tchnologies (ACL:HLT*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jinwen Guo, Shengliang Xu, Shenghua Bao, and Yong Yu. 2008. Tapping on the potential of QA community by recommending answer providers. In *Proc. of CIKM*, CIKM '08, pages 921–930.

Manish Gupta, Rui Li, Zhijun Yin, and Jiawei Han. 2010. Survey on social tagging techniques. *SIGKDD Explor. Newsl.*, 12(1):58–72, November.

Yunhua Hu, Yanan Qian, Hang Li, Daxin Jiang, Jian Pei, and Qinghua Zheng. 2012. Mining query subtopics from search log data. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 305–314, New York, NY, USA. ACM.

Makoto P. Kato, Tetsuya Sakai, and Katsumi Tanaka. 2012. Structured query suggestion for specialization and parallel movement: effect on search behaviors. In *Proc. of WWW*, WWW '12, pages 389–398.

Baichuan Li, Irwin King, and Michael R. Lyu. 2011. Question routing in community question answering: putting category in its place. In *Proc. of CIKM*, CIKM '11, pages 2041–2044.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *AAAI*.

Xiaohua Liu, Ming Zhou, Furu Wei, Zhongyang Fu, and Xiangyang Zhou. 2012. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 526–535, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proc. of KDD*, KDD '07, pages 490–499.

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. 2002. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856.

Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. 2006. Searching coordinate terms with their context from the web. In *Proceedings of the 7th international conference on Web Information Systems*, WISE'06, pages 40–47, Berlin, Heidelberg. Springer-Verlag.

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1400–1405. AAAI Press.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 238–247, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: using search engines for robust cross-domain named entity recognition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ACL-HLT '11, pages 965–975, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter Schonhofen. 2006. Identifying document topics using the wikipedia category network. In *Proceedings*

of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, WI '06, pages 456–462, Washington, DC, USA. IEEE Computer Society.

Dafna Shahaf, Jaewon Yang, Caroline Suen, Jeff Jacobs, Heidi Wang, and Jure Leskovec. 2013. Information cartography: creating zoomable, large-scale maps of information. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '13, pages 1097–1105, New York, NY, USA. ACM.

Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 259–266, New York, NY, USA. ACM.

Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 993–1001, Stroudsburg, PA, USA. Association for Computational Linguistics.

Amit Singh. 2012a. Entity based QA retrieval. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1266–1277, Stroudsburg, PA, USA. Association for Computational Linguistics.

Amit Singh. 2012b. Entity based translation language model. In *Proc. of WWW*, WWW '12 Companion, pages 599–600.

Surender Reddy Yerva, Zoltán Miklós, and Karl Aberer. 2012a. Entity-based classification of twitter messages. *IJCSA*, 9(1):88–115.

Surender Reddy Yerva, Zoltan Miklos, Flavia Grosan, Alexandru Tandrau, and Karl Aberer. 2012b. Tweetspector: entity-based retrieval of tweets. In *Proc. of SIGIR*, SIGIR '12, pages 1016–1016.

Quan Yuan, Gao Cong, Aixin Sun, Chin-Yew Lin, and Nadia Magnenat Thalmann. 2012. Category hierarchy maintenance: a data-driven approach. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 791–800, New York, NY, USA. ACM.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, April.

Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. 2009. Employing topic models for pattern-based semantic class discovery. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 459–467, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xingwei Zhu, Zhao-Yan Ming, Xiaoyan Zhu, and Tat-Seng Chua. 2013. Topic hierarchy construction for the organization of multi-source user generated contents. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '13, pages 233–242, New York, NY, USA. ACM.

Semantic Parsing on Freebase from Question-Answer Pairs

Jonathan Berant **Andrew Chou** **Roy Frostig** **Percy Liang**
Computer Science Department, Stanford University
{joberant,akchou}@stanford.edu {rf,pliang}@cs.stanford.edu

Abstract

In this paper, we train a semantic parser that scales up to Freebase. Instead of relying on annotated logical forms, which is especially expensive to obtain at large scale, we learn from question-answer pairs. The main challenge in this setting is narrowing down the huge number of possible logical predicates for a given question. We tackle this problem in two ways: First, we build a coarse mapping from phrases to predicates using a knowledge base and a large text corpus. Second, we use a *bridging* operation to generate additional predicates based on neighboring predicates. On the dataset of Cai and Yates (2013), despite not having annotated logical forms, our system outperforms their state-of-the-art parser. Additionally, we collected a more realistic and challenging dataset of question-answer pairs and improves over a natural baseline.

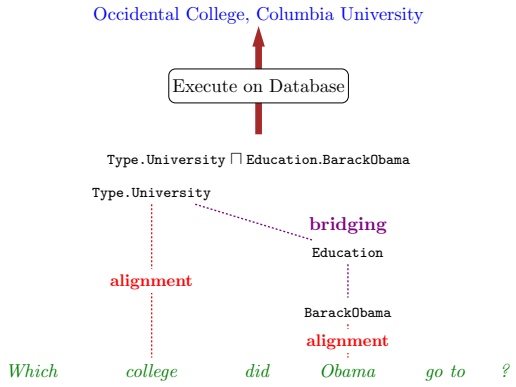


Figure 1: Our task is to map questions to answers via latent logical forms. To narrow down the space of logical predicates, we use a (i) coarse *alignment* based on Freebase and a text corpus and (ii) a *bridging* operation that generates predicates compatible with neighboring predicates.

1 Introduction

We focus on the problem of semantic parsing natural language utterances into logical forms that can be executed to produce denotations. Traditional semantic parsers (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowski et al., 2010) have two limitations: (i) they require annotated logical forms as supervision, and (ii) they operate in limited domains with a small number of logical predicates. Recent developments aim to lift these limitations, either by reducing the amount of supervision (Clarke et al., 2010; Liang et al., 2011; Goldwasser et al., 2011; Artzi and Zettlemoyer, 2011) or by increasing the number of logical predicates (Cai and Yates, 2013). The goal of this paper is to do both: learn a semantic parser without annotated logical forms that scales to the large number of predicates on Freebase.

At the lexical level, a major challenge in semantic parsing is mapping natural language phrases (e.g., *"attend"*) to logical predicates (e.g., `Education`). While limited-domain semantic parsers are able to learn the lexicon from per-example supervision (Kwiatkowski et al., 2011; Liang et al., 2011), at large scale they have inadequate coverage (Cai and Yates, 2013). Previous work on semantic parsing on Freebase uses a combination of manual rules (Yahya et al., 2012; Unger et al., 2012), distant supervision (Krishnamurthy and Mitchell, 2012), and schema

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1533–1544,
Seattle, Washington, USA, 18-21 October 2013.

matching (Cai and Yates, 2013). We use a large amount of web text and a knowledge base to build a coarse alignment between phrases and predicates—an approach similar in spirit to Cai and Yates (2013).

However, this alignment only allows us to generate a subset of the desired predicates. Aligning light verbs (e.g., *"go"*) and prepositions is not very informative due to polysemy, and rare predicates (e.g., *"cover price"*) are difficult to cover even given a large corpus. To improve coverage, we propose a new *bridging* operation that generates predicates based on adjacent predicates rather than on words.

At the compositional level, a semantic parser must combine the predicates into a coherent logical form. Previous work based on CCG requires manually specifying combination rules (Krishnamurthy and Mitchell, 2012) or inducing the rules from annotated logical forms (Kwiatkowski et al., 2010; Cai and Yates, 2013). We instead define a few simple composition rules which over-generate and then use model features to simulate soft rules and categories. In particular, we use POS tag features and features on the denotations of the predicted logical forms.

We experimented with two question answering datasets on Freebase. First, on the dataset of Cai and Yates (2013), we showed that our system outperforms their state-of-the-art system 62% to 59%, despite using no annotated logical forms. Second, we collected a new realistic dataset of questions by performing a breadth-first search using the Google Suggest API; these questions are then answered by Amazon Mechanical Turk workers. Although this dataset is much more challenging and noisy, we are still able to achieve 31.4% accuracy, a 4.5% absolute improvement over a natural baseline. Both datasets as well as the source code for SEMPRE, our semantic parser, are publicly released and can be downloaded from `http://nlp.stanford.edu/software/sempre/`.

2 Setup

Problem Statement Our task is as follows: Given (i) a knowledge base $\mathcal{K}$, and (ii) a training set of question-answer pairs $\{(x_i, y_i)\}_{i=1}^n$, output a semantic parser that maps new questions x to answers y via latent logical forms z and the knowledge base $\mathcal{K}$.

2.1 Knowledge base

Let $\mathcal{E}$ denote a set of *entities* (e.g., `BarackObama`), and let $\mathcal{P}$ denote a set of *properties* (e.g., `PlaceOfBirth`). A *knowledge base* $\mathcal{K}$ is a set of *assertions* $(e_1, p, e_2) \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ (e.g., (`BarackObama`, `PlaceOfBirth`, `Honolulu`)).

We use the Freebase knowledge base (Google, 2013), which has 41M non-numeric entities, 19K properties, and 596M assertions.[1]

2.2 Logical forms

To query the knowledge base, we use a logical language called Lambda Dependency-Based Compositional Semantics (λ-DCS)—see Liang (2013) for details. For the purposes of this paper, we use a restricted subset called *simple λ-DCS*, which we will define below for the sake of completeness.

The chief motivation of λ-DCS is to produce logical forms that are simpler than lambda calculus forms. For example, $\lambda x.\exists a.p_1(x,a) \wedge \exists b.p_2(a,b) \wedge p_3(b,e)$ is expressed compactly in λ-DCS as $p_1.p_2.p_3.e$. Like DCS (Liang et al., 2011), λ-DCS makes existential quantification implicit, thereby reducing the number of variables. Variables are only used for anaphora and building composite binary predicates; these do not appear in simple λ-DCS.

Each logical form in simple λ-DCS is either a *unary* (which denotes a subset of $\mathcal{E}$) or a *binary* (which denotes a subset of $\mathcal{E} \times \mathcal{E}$). The basic λ-DCS logical forms z and their denotations $[\![z]\!]_{\mathcal{K}}$ are defined recursively as follows:

- Unary base case: If $e \in \mathcal{E}$ is an entity (e.g., `Seattle`), then e is a unary logical form with $[\![z]\!]_{\mathcal{K}} = \{e\}$.
- Binary base case: If $p \in \mathcal{P}$ is a property (e.g., `PlaceOfBirth`), then p is a binary logical form with $[\![p]\!]_{\mathcal{K}} = \{(e_1, e_2) : (e_1, p, e_2) \in \mathcal{K}\}$.[2]
- Join: If b is a binary and u is a unary, then $b.u$ (e.g., `PlaceOfBirth.Seattle`) is a unary denoting a join and project: $[\![b.u]\!]_{\mathcal{K}} = \{e_1 \in \mathcal{E} : \exists e_2.(e_1, e_2) \in [\![b]\!]_{\mathcal{K}} \wedge e_2 \in [\![u]\!]_{\mathcal{K}}\}$.

[1] In this paper, we condense Freebase names for readability (`/people/person` becomes `Person`).

[2] Binaries can be also built out of lambda abstractions (e.g., $\lambda x.$`Performance.Actor`$.x$), but as these constructions are not central to this paper, we defer to (Liang, 2013).

- Intersection: If u_1 and u_2 are both unaries, then $u_1 \sqcap u_2$ (e.g., `Profession.Scientist` $\sqcap$ `PlaceOfBirth.Seattle`) denotes set intersection: $[\![u_1 \sqcap u_2]\!]_{\mathcal{K}} = [\![u_1]\!]_{\mathcal{K}} \cap [\![u_2]\!]_{\mathcal{K}}$.
- Aggregation: If u is a unary, then `count`(u) denotes the cardinality: $[\![\texttt{count}(u)]\!]_{\mathcal{K}} = \{|[\![u]\!]_{\mathcal{K}}|\}$.

As a final example, *"number of dramas starring Tom Cruise"* in lambda calculus would be represented as `count`$(\lambda x.\texttt{Genre}(x, \texttt{Drama}) \wedge \exists y.\texttt{Performance}(x, y) \wedge \texttt{Actor}(y, \texttt{TomCruise}))$; in λ-DCS, it is simply `count`$(\texttt{Genre.Drama} \sqcap \texttt{Performance.Actor.TomCruise})$.

It is useful to think of the knowledge base $\mathcal{K}$ as a directed graph in which entities are nodes and properties are labels on the edges. Then simple λ-DCS unary logical forms are tree-like graph patterns which pick out a subset of the nodes.

2.3 Framework

Given an utterance x, our semantic parser constructs a distribution over possible derivations $D(x)$. Each *derivation* $d \in D(x)$ is a tree specifying the application of a set of combination rules that culminates in the logical form $d.z$ at the root of the tree—see Figure 2 for an example.

Composition Derivations are constructed recursively based on (i) a lexicon mapping natural language phrases to knowledge base predicates, and (ii) a small set of composition rules.

More specifically, we build a set of derivations for each span of the utterance. We first use the lexicon to generate single-predicate derivations for any matching span (e.g., *"born"* maps to `PeopleBornHere`). Then, given any logical form z_1 that has been constructed over the span $[i_1 : j_1]$ and z_2 over a non-overlapping span $[i_2 : j_2]$, we generate the following logical forms over the enclosing span $[\min(i_1, i_2) : \max(j_1, j_2)]$: intersection $z_1 \sqcap z_2$, join $z_1.z_2$, aggregation $z_1(z_2)$ (e.g., if $z_1 =$ count), or bridging $z_1 \sqcap p.z_2$ for any property $p \in \mathcal{P}$ (explained more in Section 3.2).[3]

Note that the construction of derivations $D(x)$ allows us to skip any words, and in general heavily over-generates. We instead rely on features and learning to guide us away from the bad derivations.

[3]We also discard logical forms are incompatible according to the Freebase types (e.g., `Profession.Politician` $\sqcap$ `Type.City` would be rejected).

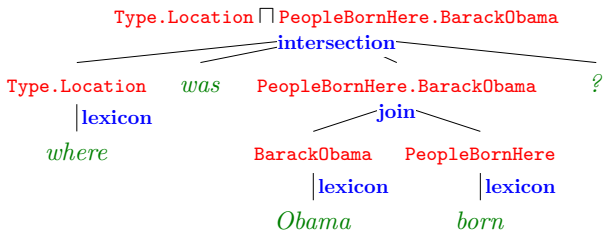


Figure 2: An example of a derivation d of the utterance *"Where was Obama born?"* and its sub-derivations, each labeled with composition rule (in blue) and logical form (in red). The derivation d skips the words *"was"* and *"?"*.

Modeling Following Zettlemoyer and Collins (2005) and Liang et al. (2011), we define a discriminative log-linear model over derivations $d \in D(x)$ given utterances x: $p_\theta(d \mid x) = \frac{\exp\{\phi(x,d)^\top \theta\}}{\sum_{d' \in D(x)} \exp\{\phi(x,d')^\top \theta\}}$, where $\phi(x, d)$ is a feature vector extracted from the utterance and the derivation, and $\theta \in \mathbb{R}^b$ is the vector of parameters to be learned. As our training data consists only of question-answer pairs (x_i, y_i), we maximize the log-likelihood of the correct answer ($[\![d.z]\!]_{\mathcal{K}} = y_i$), summing over the latent derivation d. Formally, our training objective is

$$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log \sum_{d \in D(x): [\![d.z]\!]_{\mathcal{K}} = y_i} p_\theta(d \mid x_i). \quad (1)$$

Section 4 describes an approximation of this objective that we maximize to choose parameters θ.

3 Approach

Our knowledge base has more than 19,000 properties, so a major challenge is generating a manageable set of predicates for an utterance. We propose two strategies for doing this. First (Section 3.1), we construct a lexicon that maps natural language phrases to logical predicates by aligning a large text corpus to Freebase, reminiscent of Cai and Yates (2013). Second, we generate logical predicates compatible with neighboring predicates using the bridging operation (Section 3.2). Bridging is crucial when aligning phrases is difficult or even impossible. The derivations produced by combining these predicates

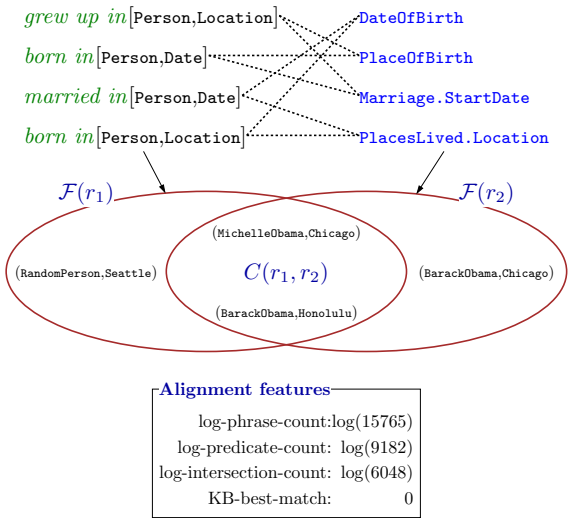


Figure 3: We construct a bipartite graph over phrases $\mathcal{R}_1$ and predicates $\mathcal{R}_2$. Each edge (r_1, r_2) is associated with alignment features.

are scored using features that capture lexical, syntactic and semantic regularities (Section 3.3).

3.1 Alignment

We now discuss the construction of a *lexicon* $\mathcal{L}$, which is a mapping from natural language phrases to logical predicates accompanied by a set of features. Specifically, for a phrase w (e.g., *"born in"*), $\mathcal{L}(w)$ is a set of entries (z, s), where z is a predicate and s is the set of features. A lexicon is constructed by alignment of a large text corpus to the knowledge base (KB). Intuitively, a phrase and a predicate align if they co-occur with many of the same entities.

Here is a summary of our alignment procedure: We construct a set of typed[4] phrases $\mathcal{R}_1$ (e.g., *"born in"*[Person,Location]) and predicates $\mathcal{R}_2$ (e.g., PlaceOfBirth). For each $r \in \mathcal{R}_1 \cup \mathcal{R}_2$, we create its *extension* $\mathcal{F}(r)$, which is a set of co-occurring entity-pairs (e.g., $\mathcal{F}$(*"born in"*[Person,Location]) $= \{$(BarackObama, Honolulu), ...$\}$. The lexicon is generated based on the overlap $\mathcal{F}(r_1) \cap \mathcal{F}(r_2)$, for $r_1 \in R_1$ and $r_2 \in R_2$.

Typed phrases 15 million triples (e_1, r, e_2) (e.g., (*"Obama"*, *"was also born in"*, *"August 1961"*)) were extracted from ClueWeb09 using the ReVerb open IE system (Fader et al., 2011). Lin et al. (2012) released a subset of these triples[5] where they were able to substitute the subject arguments with KB entities. We downloaded their dataset and heuristically replaced object arguments with KB entities by walking on the Freebase graph from subject KB entities and performing simple string matching. In addition, we normalized dates with SUTime (Chang and Manning, 2012).

We lemmatize and normalize each text phrase $r \in R_1$ and augment it with a type signature $[t_1, t_2]$ to deal with polysemy (*"born in"* could either map to PlaceOfBirth or DateOfBirth). We add an entity pair (e_1, e_2) to the extension of $\mathcal{F}(r[t_1, t_2])$ if the (Freebase) type of e_1 (e_2) is t_1 (t_2). For example, (BarackObama, 1961) is added to $\mathcal{F}$(*"born in"*[Person, Date]). We perform a similar procedure that uses a Hearst-like pattern (Hearst, 1992) to map phrases to unary predicates. If a text phrase $r \in R_1$ matches the pattern "(is|was a|the) x IN", where IN is a preposition, then we add e_1 to $\mathcal{F}(x)$. For (Honolulu, *"is a city in"*, Hawaii), we extract $x =$ *"city"* and add Honolulu to $\mathcal{F}$(*"city"*). From the initial 15M triples, we extracted 55,081 typed binary phrases (9,456 untyped) and 6,299 unary phrases.

Logical predicates Binary logical predicates contain (i) all KB properties[6] and (ii) concatenations of two properties $p_1.p_2$ if the intermediate type represents an event (e.g., the *married to* relation is represented by Marriage.Spouse). For unary predicates, we consider all logical forms Type.t and Profession.t for all (abstract) entities $t \in \mathcal{E}$ (e.g. Type.Book and Profession.Author). The types of logical predicates considered during alignment is restricted in this paper, but automatic induction of more compositional logical predicates is an interesting direction. Finally, we define the extension of a logical predicate $r \in \mathcal{R}_2$ to be its denotation, that is, the corresponding set of entities or entity pairs.

Lexicon construction Given typed phrases $\mathcal{R}_1$, logical predicates $\mathcal{R}_2$, and their extensions $\mathcal{F}$, we now generate the lexicon. It is useful to think of a

[4] Freebase associates each entity with a set of types using the Type property.

[5] http://knowitall.cs.washington.edu/linked_extractions/

[6] We filter properties from the domains user and base.

Category	Description
Alignment	Log of # entity pairs that occur with the phrase r_1 ($\lvert\mathcal{F}(r_1)\rvert$) Log of # entity pairs that occur with the logical predicate r_2 ($\lvert\mathcal{F}(r_2)\rvert$) Log of # entity pairs that occur with both r_1 and r_2 ($\lvert\mathcal{F}(r_1) \cap \mathcal{F}(r_2)\rvert$) Whether r_2 is the best match for r_1 ($r_2 = \arg\max_r \lvert\mathcal{F}(r_1) \cap \mathcal{F}(r)\rvert$)
Lexicalized	Conjunction of phrase w and predicate z
Text similarity	Phrase r_1 is equal/prefix/suffix of s_2 Phrase overlap of r_1 and s_2
Bridging	Log of # entity pairs that occur with bridging predicate b ($\lvert\mathcal{F}(b)\rvert$) Kind of bridging (# unaries involved) The binary b injected
Composition	# of intersect/join/bridging operations POS tags in join/bridging and skipped words Size of denotation of logical form

Table 1: Full set of features. For the alignment and text similarity, r_1 is a phrase, r_2 is a predicate with Freebase name s_2, and b is a binary predicate with type signature (t_1, t_2).

bipartite graph with left nodes $\mathcal{R}_1$ and right nodes $\mathcal{R}_2$ (Figure 3). We add an edge (r_1, r_2) if (i) the type signatures of r_1 and r_2 match[7] and (ii) their extensions have non-empty overlap ($\mathcal{F}(r_1) \cap \mathcal{F}(r_2) \neq \emptyset$). Our final graph contains 109K edges for binary predicates and 294K edges for unary predicates.

Naturally, non-zero overlap by no means guarantees that r_1 should map to r_2. In our noisy data, even *"born in"* and `Marriage.EndDate` co-occur 4 times. Rather than thresholding based on some criterion, we compute a set of features, which are used by the model downstream in conjunction with other sources of information.

We compute three types of features (Table 1). *Alignment* features are unlexicalized and measure association based on argument overlap. *Lexicalized* features are standard conjunctions of the phrase w and the logical form z. *Text similarity* features compare the (untyped) phrase (e.g., *"born"*) to the Freebase name of the logical predicate (e.g., *"People born here"*): Given the phrase r_1 and the Freebase name s_2 of the predicate r_2, we compute string similarity features such as whether r_1 and s_2 are equal, as well as some other measures of token overlap.

[7]Each Freebase property has a designated type signature, which can be extended to composite predicates, e.g., sig(`Marriage.StartDate`) = (`Person`, `Date`).

3.2 Bridging

While alignment can cover many predicates, it is unreliable for cases where the predicates are expressed weakly or implicitly. For example, in *"What government does Chile have?"*, the predicate is expressed by the light verb *have*, in *"What actors are in Top Gun?"*, it is expressed by a highly ambiguous preposition, and in *"What is Italy money?"* [sic], it is omitted altogether. Since natural language doesn't offer much help here, let us turn elsewhere for guidance. Recall that at this point our main goal is to generate a manageable set of candidate logical forms to be scored by the log-linear model.

In the first example, suppose the phrases *"Chile"* and *"government"* are parsed as `Chile` and `Type.FormOfGovernment`, respectively, and we hypothesize a connecting binary. The two predicates impose strong type constraints on that binary, so we can afford to generate all the binary predicates that type check (see Table 2). More formally, given two unaries z_1 and z_2 with types t_1 and t_2, we generate a logical form $z_1 \sqcap b.z_2$ for each binary b whose type signature is (t_1, t_2). Figure 1 visualizes bridging of the unaries `Type.University` and `Obama`.

Now consider the example *"What is the cover price of X-men?"* Here, the binary `ComicBookCoverPrice` is expressed explicitly, but is not in our lexicon since the language use is rare. To handle this, we allow bridging to generate a binary based on a single unary; in this case, based on the unary `X-Men` (Table 2), we generate several binaries including `ComicBookCoverPrice`. Generically, given a unary z with type t, we construct a logical form $b.z$ for any predicate b with type $(*, t)$.

Finally, consider the question *"Who did Tom Cruise marry in 2006?"*. Suppose we parse the phrase *"Tom Cruise marry"* into `Marriage.Spouse.TomCruise`, or more explicitly, $\lambda x.\exists e.\texttt{Marriage}(x, e) \wedge \texttt{Spouse}(e, \texttt{TomCruise})$. Here, the neo-Davidsonian event variable e is an intermediate quantity, but needs to be further modified (in this case, by the temporal modifier *2006*). To handle this, we apply bridging to a unary and the intermediate event (see Table 2). Generically, given a logical form $p_1.p_2.z'$ where p_2 has type $(t_1, *)$, and a unary z with type t, bridging injects z and

#	Form 1	Form 2	Bridging
1	`Type.FormOfGovernment`	`Chile`	`Type.FormOfGovernment` $\sqcap$ **`GovernmentTypeOf`**`.Chile`
2		`X-Men`	**`ComicBookCoverPriceOf`**`.X-Men`
3	`Marriage.Spouse.TomCruise`	`2006`	`Marriage.(Spouse.TomCruise` $\sqcap$ **`StartDate`**`.2006)`

Table 2: Three examples of the bridging operation. The bridging binary predicate b is in boldface.

constructs a logical form $p_1.(p_2.z' \sqcap b.z)$ for each logical predicate b with type (t_1, t).

In each of the three examples, bridging generates a binary predicate based on neighboring logical predicates rather than on explicit lexical material. In a way, our *bridging* operation shares with bridging anaphora (Clark, 1975) the idea of establishing a novel relation between distinct parts of a sentence. Naturally, we need features to distinguish between the generated predicates, or decide whether bridging is even appropriate at all. Given a binary b, features include the log of the predicate count $\log |\mathcal{F}(b)|$, indicators for the kind of bridging, an indicator on the binary b for injections (Table 1). In addition, we add all text similarity features by comparing the Freebase name of b with content words in the question.

3.3 Composition

So far, we have mainly focused on the generation of predicates. We now discuss three classes of features pertaining to their composition.

Rule features Each derivation d is the result of applying some number of intersection, join, and bridging operations. To control this number, we define indicator features on each of these counts. This is in contrast to the norm of having a single feature whose value is equal to the count, which can only represent one-sided preferences for having more or fewer of a given operation. Indicator features stabilize the model, preferring derivations with a well-balanced inventory of operations.

Part-of-speech tag features To guide the composition of predicates, we use POS tags in two ways. First, we introduce features indicating when a word of a given POS tag is skipped, which could capture the fact that skipping auxiliaries is generally acceptable, while skipping proper nouns is not. Second, we introduce features on the POS tags involved in a composition, inspired by dependency parsing (McDonald et al., 2005). Specifically, when we combine logical forms z_1 and z_2 via a join or bridging, we include a feature on the POS tag of (the first word spanned by) z_1 conjoined with the POS tag corresponding to z_2. Rather than using head-modifier information from dependency trees (Branavan et al., 2012; Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013; Poon, 2013), we can learn the appropriate relationships tailored for downstream accuracy. For example, the phrase *"located"* is aligned to the predicate `ContainedBy`. POS features can detect that if *"located"* precedes a noun phrase (*"What is located in Beijing?"*), then the noun phrase is the object of the predicate, and if it follows the noun phrase (*"Where is Beijing located?"*), then it is in subject position.

Note that our three operations (intersection, join, and bridging) are quite permissive, and we rely on features, which encode soft, overlapping rules. In contrast, CCG-based methods (Kwiatkowski et al., 2010; Kwiatkowski et al., 2011) encode the combination preferences structurally in non-overlapping rules; these could be emulated with features with weights clamped to $-\infty$.

Denotation features While it is clear that learning from denotations rather than logical forms is a drawback since it provides less information, it is less obvious that working with denotations actually gives us additional information. Specifically, we include four features indicating whether the denotation of the predicted logical form has size 0, 1, 2, or at least 3. This feature encodes presupposition constraints in a soft way: when people ask a question, usually there is an answer and it is often unique. This allows us to favor logical forms with this property.

4 Experiments

We now evaluate our semantic parser empirically. In Section 4.1, we compare our approach to Cai and Yates (2013) on their recently released dataset (henceforth, FREE917) and present results on a new

dataset that we collected (henceforth, WEBQUESTIONS). In Section 4.2, we provide detailed experiments to provide additional insight on our system.

Setup We implemented a standard beam-based bottom-up parser which stores the k-best derivations for each span. We use $k = 500$ for all our experiments on FREE917 and $k = 200$ on WEBQUESTIONS. The root beam yields the candidate set $\tilde{D}(x)$ and is used to approximate the sum in the objective function $\mathcal{O}(\theta)$ in (1). In experiments on WEBQUESTIONS, $\tilde{D}(x)$ contained 197 derivations on average.

We write the approximate objective as $\mathcal{O}(\theta; \tilde{\theta}) = \sum_i \log \sum_{d \in \tilde{D}(x_i;\tilde{\theta}):[\![d.z]\!]_{\mathcal{K}}=y_i} p(d \mid x_i; \theta)$ to explicitly show dependence on the parameters $\tilde{\theta}$ used for beam search. We optimize the objective by initializing θ_0 to 0 and applying AdaGrad (stochastic gradient ascent with per-feature adaptive step size control) (Duchi et al., 2010), so that θ_{t+1} is set based on taking a stochastic approximation of $\frac{\partial \mathcal{O}(\theta;\theta_t)}{\partial \theta}\big|_{\theta=\theta_t}$. We make six passes over the training examples.

We used POS tagging and named-entity recognition to restrict what phrases in the utterance could be mapped by the lexicon. Entities must be named entities, proper nouns or a sequence of at least two tokens. Unaries must be a sequence of nouns, and binaries must be either a content word, or a verb followed by either a noun phrase or a particle. In addition, we used 17 hand-written rules to map question words such as *"where"* and *"how many"* to logical forms such as `Type.Location` and `Count`.

To compute denotations, we convert a logical form z into a SPARQL query and execute it on our copy of Freebase using the Virtuoso engine. On WEBQUESTIONS, a full run over the training examples involves approximately 600,000 queries. For evaluation, we predict the answer from the derivation with highest probability.

4.1 Main results

4.1.1 FREE917

Cai and Yates (2013) created a dataset consisting of 917 questions involving 635 Freebase relations, annotated with lambda calculus forms. We converted all 917 questions into simple λ-DCS, executed them on Freebase and used the resulting answers to train and evaluate. To map phrases to Freebase entities we used the manually-created entity lexicon used by Cai and Yates (2013), which contains 1,100 entries. Because entity disambiguation is a challenging problem in semantic parsing, the entity lexicon simplifies the problem.

Following Cai and Yates (2013), we held out 30% of the examples for the final test, and performed all development on the remaining 70%. During development, we split the data and used 512 examples (80%) for training and the remaining 129 (20%) for validation. All reported development numbers are averaged across 3 random splits. We evaluated using accuracy, the fraction of examples where the predicted answer exactly matched the correct answer.

Our main empirical result is that our system, which was trained only on question-answer pairs, obtained 62% accuracy on the test set, outperforming the 59% accuracy reported by Cai and Yates (2013), who trained on full logical forms.

4.1.2 WEBQUESTIONS

Dataset collection Because FREE917 requires logical forms, it is difficult to scale up due to the required expertise of annotating logical forms. We therefore created a new dataset, WEBQUESTIONS, of question-answer pairs obtained from non-experts.

To collect this dataset, we used the Google Suggest API to obtain questions that begin with a wh-word and contain exactly one entity. We started with the question *"Where was Barack Obama born?"* and performed a breadth-first search over questions (nodes), using the Google Suggest API supplying the edges of the graph. Specifically, we queried the question excluding the entity, the phrase before the entity, or the phrase after it; each query generates 5 candidate questions, which are added to the queue. We iterated until 1M questions were visited; a random 100K were submitted to Amazon Mechanical Turk (AMT).

The AMT task requested that workers answer the question using only the Freebase page of the questions' entity, or otherwise mark it as unanswerable by Freebase. The answer was restricted to be one of the possible entities, values, or list of entities on the page. As this list was long, we allowed the user to filter the list by typing. We paid the workers $0.03 per question. Out of 100K questions, 6,642 were annotated identically by at least two AMT workers.

We again held out a 35% random subset of the

Dataset	# examples	# word types
GeoQuery	880	279
ATIS	5,418	936
FREE917	917	2,036
WEBQUESTIONS	5,810	4,525

Table 3: Statistics on various semantic parsing datasets. Our new dataset, WEBQUESTIONS, is much larger than FREE917 and much more lexically diverse than ATIS.

System	FREE917	WebQ.
ALIGNMENT	38.0	30.6
BRIDGING	66.9	21.2
ALIGNMENT+BRIDGING	**71.3**	**32.9**

Table 4: Accuracies on the development set under different schemes of binary predicate generation. In ALIGNMENT, binaries are generated only via the alignment lexicon. In BRIDGING, binaries are generated through the bridging operation only. ALIGNMENT+BRIDGING corresponds to the full system.

questions for the final test, and performed all development on the remaining 65%, which was further divided into an 80%–20% split for training and validation. To map entities, we built a Lucene index over the 41M Freebase entities.

Table 3 provides some statistics about the new questions. One major difference in the datasets is the distribution of questions: FREE917 starts from Freebase properties and solicits questions about these properties; these questions tend to be tailored to the properties. WEBQUESTIONS starts from questions completely independent of Freebase, and therefore the questions tend to be more natural and varied. For example, for the Freebase property `ComicGenre`, FREE917 contains the question *"What genre is Doonesbury?"*, while WEBQUESTIONS for the property `MusicGenre` contains *"What music did Beethoven compose?"*.

The number of word types in WEBQUESTIONS is larger than in datasets such as ATIS and GeoQuery (Table 3), making lexical mapping much more challenging. On the other hand, in terms of structural complexity WEBQUESTIONS is simpler and many questions contain a unary, a binary and an entity.

In some questions, the answer provided by AMT workers is only roughly accurate, because workers are restricted to selecting answers from the Freebase page. For example, the answer given by workers to the question *"What is James Madison most famous for?"* is *"President of the United States"* rather than *"Authoring the Bill of Rights"*.

Results AMT workers sometimes provide partial answers, e.g., the answer to *"What movies does Taylor Lautner play in?"* is a set of 17 entities, out of which only 10 appear on the Freebase page. We therefore allow partial credit and score an answer using the F_1 measure, comparing the predicted set of entities to the annotated set of entities.

As a baseline, we omit from our system the main contributions presented in this paper—that is, we disallow bridging, and remove denotation and alignment features. The accuracy on the test set of this system is 26.9%, whereas our full system obtains 31.4%, a significant improvement.

Note that the number of possible derivations for questions in WEBQUESTIONS is quite large. In the question *"What kind of system of government does the United States have?"* the phrase *"United States"* maps to 231 entities in our lexicon, the verb *"have"* maps to 203 binaries, and the phrases *"kind"*, *"system"*, and *"government"* all map to many different unary and binary predicates. Parsing correctly involves skipping some words, mapping other words to predicates, while resolving many ambiguities in the way that the various predicates can combine.

4.2 Detailed analysis

We now delve deeper to explore the contributions of the various components of our system. All ablation results reported next were run on the development set (over 3 random splits).

Generation of binary predicates Recall that our system has two mechanisms for suggesting binaries: from the alignment lexicon or via the bridging operation. Table 4 shows accuracies when only one or both is used. Interestingly, alignment alone is better than bridging alone on WEBQUESTIONS, whereas for FREE917, it is the opposite. The reason for this is that FREE917 contains questions on rare predicates. These are often missing from the lexicon, but tend to have distinctive types and hence can be predicted from neighboring predicates. In contrast, WEBQUESTIONS contains questions that are commonly searched for and focuses on popular predicates, therefore exhibiting larger lexical variation.

System	FREE917	WebQ.
FULL	**71.3**	**32.9**
-POS	70.5	28.9
-DENOTATION	58.6	28.0

Table 5: Accuracies on the development set with features removed. POS and DENOTATION refer to the POS tag and denotation features from Section 3.3.

System	FREE917	WebQ.
ALIGNMENT	**71.3**	32.9
LEXICALIZED	68.5	34.2
LEXICALIZED+ALIGNMENT	69.0	**36.4**

Table 6: Accuracies on the development set using either unlexicalized alignment features (ALIGNMENT) or lexicalized features (LEXICALIZED).

For instance, when training without an alignment lexicon, the system errs on *"When did Nathan Smith die?"*. Bridging suggests binaries that are compatible with the common types `Person` and `Datetime`, and the binary `PlaceOfBirth` is chosen. On the other hand, without bridging, the system errs on *"In which comic book issue did Kitty Pryde first appear?"*, which refers to the rare predicate `ComicBookFirstAppearance`. With bridging, the parser can identify the correct binary, by linking the types `ComicBook` and `ComicBookCharacter`. On both datasets, best performance is achieved by combining the two sources of information.

Overall, running on WEBQUESTIONS, the parser constructs derivations that contain about 12,000 distinct binary predicates.

Feature variations Table 5 shows the results of feature ablation studies. Accuracy drops when POS tag features are omitted, e.g., in the question *"What number is Kevin Youkilis on the Boston Red Sox"* the parser happily skips the NNPs *"Kevin Youkilis"* and returns the numbers of all players on the Boston Red Sox. A significant loss is incurred without denotation features, largely due to the parser returning logical forms with empty denotations. For instance, the question *"How many people were at the 2006 FIFA world cup final?"* is answered with a logical form containing the property `PeopleInvolved` rather than `SoccerMatchAttendance`, resulting in an empty denotation.

Next we study the impact of lexicalized versus unlexicalized features (Table 6). In the large WEBQUESTIONS dataset, lexicalized features helped, and so we added those features to our model when running on the test set. In FREE917 lexicalized features result in overfitting due to the small number of training examples. Thus, we ran our final parser on the test set without lexicalized features.

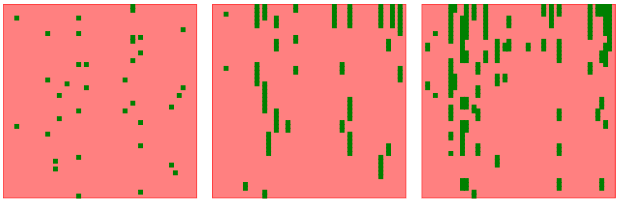
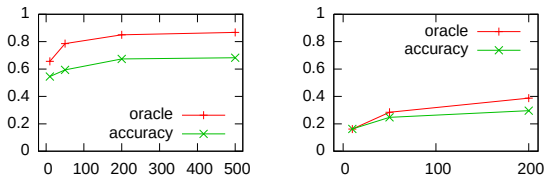


Figure 4: Beam of candidate derivations $\tilde{D}(x)$ for 50 WEBQUESTIONS examples. In each matrix, columns correspond to examples and rows correspond to beam position (ranked by decreasing model score). Green cells mark the positions of derivations with correct denotations. Note that both the number of good derivations and their positions improve as θ is optimized.



(a) FREE917 (b) WEBQUESTIONS

Figure 5: Accuracy and oracle as beam size k increases.

Effect of beam size An intrinsic challenge in semantic parsing is to handle the exponentially large set of possible derivations. We rely heavily on the k-best beam approximation in the parser keeping good derivations that lead to the correct answer. Recall that the set of candidate derivations $\tilde{D}(x)$ depends on the parameters θ. In the initial stages of learning, θ is far from optimal, so good derivations are likely to fall below the k-best cutoff of internal parser beams. As a result, $\tilde{D}(x)$ contains few derivations with the correct answer. Still, placing these few derivations on the beam allows the training procedure to bootstrap θ into a good solution. Figure 4 illustrates this improvement in $\tilde{D}(x)$ across early training iterations.

Smaller choices of k yield a coarser approxima-

tion in beam search. As we increase k (Figure 5), we see a tapering improvement in accuracy. We also see a widening gap between accuracy and oracle score,[8] as including a good derivation in $\tilde{D}(x)$ is made easier but the learning problem is made more difficult.

Error analysis The accuracy on WEBQUESTIONS is much lower than on FREE917. We analyzed WEBQUESTIONS examples and found several main causes of error: (i) Disambiguating entities in WEBQUESTIONS is much harder because the entity lexicon has 41M entities. For example, given *"Where did the battle of New Orleans start?"* the system identifies *"New Orleans"* as the target entity rather than its surrounding noun phrase. Recall that all FREE917 experiments used a carefully chosen entity lexicon. (ii) Bridging can often fail when the question's entity is compatible with many binaries. For example, in *"What did Charles Babbage make?"*, the system chooses a wrong binary compatible with the type `Person`. (iii) The system sometimes incorrectly draws verbs from subordinate clauses. For example, in *"Where did Walt Disney live before he died?"* it returns the place of death of Walt Disney, ignoring the matrix verb *live*.

5 Discussion

Our work intersects with two strands of work. The first involves learning models of semantics guided by denotations or interactions with the world. Besides semantic parsing for querying databases (Popescu et al., 2003; Clarke et al., 2010; Liang et al., 2011), previous work has looked at interpreting natural language for performing programming tasks (Kushman and Barzilay, 2013; Lei et al., 2013), playing computer games (Branavan et al., 2010; Branavan et al., 2011), following navigational instructions (Chen, 2012; Artzi and Zettlemoyer, 2013), and interacting in the real world via perception (Matuszek et al., 2012; Tellex et al., 2011; Krishnamurthy and Kollar, 2013). Our system uses denotations rather than logical forms as a training signal, but also benefits from denotation features, which becomes possible in the grounded setting.

The second body of work involves connecting natural language and open-domain databases. Several works perform relation extraction using distant supervision from a knowledge base (Riedel et al., 2010; Carlson et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). While similar in spirit to our alignment procedure for building the lexicon, one difference is that relation extraction cares about facts, aggregating over phrases, whereas a lexicon concerns specific phrases, thus aggregating over facts. On the question answering side, recent methods have made progress in building semantic parsers for the open domain, but still require a fair amount of manual effort (Yahya et al., 2012; Unger et al., 2012; Cai and Yates, 2013). Our system reduces the amount of supervision and has a more extensive evaluation on a new dataset.

Finally, although Freebase has thousands of properties, open information extraction (Banko et al., 2007; Fader et al., 2011; Masaum et al., 2012) and associated question answering systems (Fader et al., 2013) work over an even larger open-ended set of properties. The drawback of this regime is that the noise and the difficulty in canonicalization make it hard to perform reliable composition, thereby nullifying one of the key benefits of semantic parsing. An interesting midpoint involves keeping the structured knowledge base but augmenting the predicates, for example using random walks (Lao et al., 2011) or Markov logic (Zhang et al., 2012). This would allow us to map atomic words (e.g., *"wife"*) to composite predicates (e.g., $\lambda x.\texttt{Marriage.Spouse.}(\texttt{Gender.Female} \sqcap x)$). Learning these composite predicates would drastically increase the possible space of logical forms, but we believe that the methods proposed in this paper—alignment via distant supervision and bridging—can provide some traction on this problem.

Acknowledgments

We would like to thank Thomas Lin, Mausam and Oren Etzioni for providing us with open IE triples that are partially-linked to Freebase, and also Arun Chaganty for helpful comments. The authors gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0040.

[8] Oracle score is the fraction of examples for which $\tilde{D}(x)$ contains any derivation with the correct denotation.

References

Y. Artzi and L. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432.

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1:49–62.

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*, pages 1268–1277.

S. Branavan, D. Silver, and R. Barzilay. 2011. Learning to win by reading manuals in a Monte-Carlo framework. In *Association for Computational Linguistics (ACL)*, pages 268–277.

S. Branavan, N. Kushman, T. Lei, and R. Barzilay. 2012. Learning high-level planning from text. In *Association for Computational Linguistics (ACL)*, pages 126–135.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr, and T. M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

A. X. Chang and C. Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Language Resources and Evaluation (LREC)*, pages 3735–3740.

D. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Association for Computational Linguistics (ACL)*.

H. H. Clark. 1975. Bridging. In *Workshop on theoretical issues in natural language processing*, pages 169–174.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Fader, L. Zettlemoyer, and O. Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Association for Computational Linguistics (ACL)*.

D. Goldwasser, R. Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*, pages 1486–1495.

Google. 2013. Freebase data dumps (2013-06-09). `https://developers.google.com/freebase/data`.

M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Interational Conference on Computational linguistics*, pages 539–545.

R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Association for Computational Linguistics (ACL)*, pages 541–550.

J. Krishnamurthy and T. Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics (TACL)*, 1:193–206.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 754–765.

N. Kushman and R. Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 826–836.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

N. Lao, T. Mitchell, and W. W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Lei, F. Long, R. Barzilay, and M. Rinard. 2013. From natural language specifications to program input parsers. In *Association for Computational Linguistics (ACL)*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *As-*

sociation for Computational Linguistics (ACL), pages 590–599.

P. Liang. 2013. Lambda dependency-based compositional semantics. Technical report, ArXiv.

T. Lin, Mausam, and O. Etzioni. 2012. Entity linking at web scale. In *Knowledge Extraction Workshop (AKBC-WEKEX)*.

Masaum, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. 2012. Open language learning for information extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 523–534.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Association for Computational Linguistics (ACL)*, pages 91–98.

H. Poon. 2013. Grounded unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

A. Popescu, O. Etzioni, and H. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *International Conference on Intelligent User Interfaces (IUI)*, pages 149–157.

S. Riedel, L. Yao, and A. McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 148–163.

M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 455–465.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

C. Unger, L. Bhmann, J. Lehmann, A. Ngonga, D. Gerber, and P. Cimiano. 2012. Template-based question answering over RDF data. In *World Wide Web (WWW)*, pages 639–648.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. 2012. Natural language questions for the web of data. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 379–390.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

C. Zhang, R. Hoffmann, and D. S. Weld. 2012. Ontological smoothing for relation extraction with minimal supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Scaling Semantic Parsers with On-the-fly Ontology Matching

Tom Kwiatkowski **Eunsol Choi** **Yoav Artzi** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{tomk,eunsol,yoav,lsz}@cs.washington.edu

Abstract

We consider the challenge of learning semantic parsers that scale to large, open-domain problems, such as question answering with Freebase. In such settings, the sentences cover a wide variety of topics and include many phrases whose meaning is difficult to represent in a fixed target ontology. For example, even simple phrases such as 'daughter' and 'number of people living in' cannot be directly represented in Freebase, whose ontology instead encodes facts about gender, parenthood, and population. In this paper, we introduce a new semantic parsing approach that learns to resolve such ontological mismatches. The parser is learned from question-answer pairs, uses a probabilistic CCG to build linguistically motivated logical-form meaning representations, and includes an ontology matching model that adapts the output logical forms for each target ontology. Experiments demonstrate state-of-the-art performance on two benchmark semantic parsing datasets, including a nine point accuracy improvement on a recent Freebase QA corpus.

1 Introduction

Semantic parsers map sentences to formal representations of their underlying meaning. Recently, algorithms have been developed to learn such parsers for many applications, including question answering (QA) (Kwiatkowski et al., 2011; Liang et al., 2011), relation extraction (Krishnamurthy and Mitchell, 2012), robot control (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013), interpreting instructions (Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013), and generating programs (Kushman and Barzilay, 2013).

In each case, the parser uses a predefined set of logical constants, or an ontology, to construct meaning representations. In practice, the choice of ontology significantly impacts learning. For example, consider the following questions (Q) and candidate meaning representations (MR):

Q1: What is the population of Seattle?
Q2: How many people live in Seattle?
MR1: $\lambda x.population(Seattle, x)$
MR2: $count(\lambda x.person(x) \wedge live(x, Seattle))$

A semantic parser might aim to construct MR1 for Q1 and MR2 for Q2; these pairings align constants ($count$, $person$, etc.) directly to phrases ('How many,' 'people,' etc.). Unfortunately, few ontologies have sufficient coverage to support both meaning representations, for example many QA databases would only include the population relation required for MR1. Most existing approaches would, given this deficiency, simply aim to produce MR1 for Q2, thereby introducing significant lexical ambiguity that complicates learning. Such ontological mismatches become increasingly common as domain and language complexity increases.

In this paper, we introduce a semantic parsing approach that supports scalable, open-domain ontological reasoning. The parser first constructs a linguistically motivated domain-independent meaning representation. For example, possibly producing MR1 for Q1 and MR2 for Q2 above. It then uses a learned ontology matching model to transform this represen-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1545–1556,
Seattle, Washington, USA, 18-21 October 2013.

x :	How many people visit the public library of New York annually
l_0 :	$\lambda x.eq(x, count(\lambda y.people(y) \wedge \exists e.visit(y, \iota z.public(z) \wedge library(z) \wedge of(z, new\ york), e) \wedge annually(e)))$
y :	λx.library.public_library_system.annual_visits(x, new_york_public_library)
a :	13,554,002

x :	What works did Mozart dedicate to Joseph Haydn
l_0 :	$\lambda x.works(x) \wedge \exists e.dedicate(mozart, x, e) \wedge to(haydn, e)))$
y :	λx.dedicated_work(x) $\wedge$ $\exists$e.dedicated_by(mozart, e) $\wedge$ dedication(x, e) $\wedge$ dedicated_to(haydn, e)))
a :	{ String Quartet No. 19, Haydn Quartets, String Quartet No. 16, String Quartet No. 18, String Quartet No. 17 }

Figure 1: Examples of sentences x, domain-independent underspecified logical forms l_0, fully specified logical forms y, and answers a drawn from the Freebase domain.

tation for the target domain. In our example, producing either MR1, MR2 or another more appropriate option, depending on the QA database schema. This two stage approach enables parsing without any domain-dependent lexicon that pairs words with logical constants. Instead, word meaning is filled in on-the-fly through ontology matching, enabling the parser to infer the meaning of previously unseen words and more easily transfer across domains. Figure 1 shows the desired outputs for two example Freebase sentences.

The first parsing stage uses a probabilistic combinatory categorial grammar (CCG) (Steedman, 2000; Clark and Curran, 2007) to map sentences to new, *underspecified* logical-form meaning representations containing generic logical constants that are not tied to any specific ontology. This approach enables us to share grammar structure across domains, instead of repeatedly re-learning different grammars for each target ontology. The ontology-matching step considers a large number of type-equivalent domain-specific meanings. It enables us to incorporate a number of cues, including the target ontology structure and lexical similarity between the names of the domain-independent and dependent constants, to construct the final logical forms.

During learning, we estimate a linear model over derivations that include all of the CCG parsing decisions and the choices for ontology matching. Following a number of recent approaches (Clarke et al., 2010; Liang et al., 2011), we treat all intermediate decisions as latent and learn from data containing only easily gathered question answer pairs. This approach aligns naturally with our two-stage parsing setup, where the final logical expression can be directly used to provide answers.

We report performance on two benchmark datasets: GeoQuery (Zelle and Mooney, 1996) and Freebase QA (FQ) (Cai and Yates, 2013a). GeoQuery includes a geography database with a small ontology and questions with relatively complex, compositional structure. FQ includes questions to Freebase, a large community-authored database that spans many sub-domains. Experiments demonstrate state-of-the-art performance in both cases, including a nine point improvement in recall for the FQ test.

2 Formal Overview

Task Let an ontology $\mathcal{O}$ be a set of logical constants and a knowledge base $\mathcal{K}$ be a collection of logical statements constructed with constants from $\mathcal{O}$. For example, $\mathcal{K}$ could be facts in Freebase (Bollacker et al., 2008) and $\mathcal{O}$ would define the set of entities and relation types used to encode those facts. Also, let y be a logical expression that can be executed against $\mathcal{K}$ to return an answer $a = \text{EXEC}(y, \mathcal{K})$. Figure 1 shows example queries and answers for Freebase. Our goal is to build a function $y = \text{PARSE}(x, \mathcal{O})$ for mapping a natural language sentence x to a domain-dependent logical form y.

Parsing We use a two-stage approach to define the space of possible parses $\text{GEN}(x, \mathcal{O})$ (Section 5). First, we use a CCG and word-class information from Wiktionary[1] to build domain-independent underspecified logical forms, which closely mirror the linguistic structure of the sentence but do not use constants from $\mathcal{O}$. For example, in Figure 1, l_0 denotes the underspecified logical forms paired with each sentence x. The parser then maps this intermediate representation to a logical form that uses constants from $\mathcal{O}$, such as the y seen in Figure 1.

[1] www.wiktionary.com

Learning We assume access to data containing question-answer pairs $\{(x_i, a_i) : i = 1 \dots n\}$ and a corresponding knowledge base $\mathcal{K}$. The learning algorithm (Section 7.1) estimates the parameters of a linear model for ranking the possible entires in $\text{GEN}(x, \mathcal{O})$. Unlike much previous work (e.g., Zettlemoyer and Collins (2005)), we do not induce a CCG lexicon. The lexicon is open domain, using no symbols from the ontology $\mathcal{O}$ for $\mathcal{K}$. This allows us to write a single set of lexical templates that are reused in every domain (Section 5.1). The burden of learning word meaning is shifted to the second, ontology matching, stage of parsing (Section 5.2), and modeled with a number of new features (Section 7.2) as part of the joint model.

Evaluation We evaluate on held out question-answer pairs in two benchmark domains, Freebase and GeoQuery. Following Cai and Yates (2013a), we also report a cross-domain evaluation where the Freebase data is divided by topics such as sports, film, and business. This condition ensures that the test data has a large percentage of previously unseen words, allowing us to measure the effectiveness of the real time ontology matching.

3 Related Work

Supervised approaches for learning semantic parsers have received significant attention, e.g. (Kate and Mooney, 2006; Wong and Mooney, 2007; Muresan, 2011; Kwiatkowski et al., 2010, 2011, 2012; Jones et al., 2012). However, these techniques require training data with hand-labeled domain-specific logical expressions. Recently, alternative forms of supervision were introduced, including learning from question-answer pairs (Clarke et al., 2010; Liang et al., 2011), from conversational logs (Artzi and Zettlemoyer, 2011), with distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013b), and from sentences paired with system behavior (Goldwasser and Roth, 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013). Our work adds to these efforts by demonstrating a new approach for learning with latent meaning representations that scales to large databases like Freebase.

Cai and Yates (2013a) present the most closely related work. They applied schema matching techniques to expand a CCG lexicon learned with the UBL algorithm (Kwiatkowski et al., 2010). This approach was one of the first to scale to Freebase, but required labeled logical forms and did not jointly model semantic parsing and ontological reasoning. This method serves as the state of the art for our comparison in Section 9.

We build on a number of existing algorithmic ideas, including using CCGs to build meaning representations (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010, 2011), building derivations to transform the output of the CCG parser based on context (Zettlemoyer and Collins, 2009), and using weakly supervised margin-sensitive parameter updates (Artzi and Zettlemoyer, 2011, 2013). However, we introduce the idea of learning an open-domain CCG semantic parser; all previous methods suffered, to various degrees, from the ontological mismatch problem that motivates our work.

The challenge of ontological mismatch has been previously recognized in many settings. Hobbs (1985) describes the need for ontological promiscuity in general language understanding. Many previous hand-engineered natural language understanding systems (Grosz et al., 1987; Alshawi, 1992; Bos, 2008) are designed to build general meaning representations that are adapted for different domains. Recent efforts to build natural language interfaces to large databases, for example DBpedia (Yahya et al., 2012; Unger et al., 2012), have also used hand-engineered ontology matching techniques. Fader et al. (2013) recently presented a scalable approach to learning an open domain QA system, where ontological mismatches are resolved with learned paraphrases. Finally, the databases research community has a long history of developing schema matching techniques (Doan et al., 2004; Euzenat et al., 2007), which has inspired more recent work on distant supervision for relation extraction with Freebase (Zhang et al., 2012).

4 Background

Semantic Modeling We use the typed lambda calculus to build logical forms that represent the meanings of words, phrases and sentences. Logical forms contain constants, variables, lambda abstractions, and literals. In this paper, we use the term literal to refer to the application of a constant to a sequence of

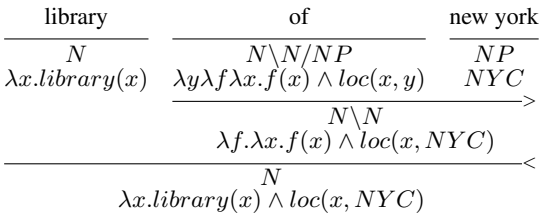


Figure 2: A sample CCG parse.

arguments. We include types for entities e, truth values t, numbers i, events ev, and higher-order functions, such as $\langle e, t\rangle$ and $\langle\langle e, t\rangle, e\rangle$. We use Davidsonian event semantics (Davidson, 1967) to explicitly represent events using event-typed variables and conjunctive modifiers to capture thematic roles.

Combinatory Categorial Grammars (CCG) CCGs are a linguistically-motivated formalism for modeling a wide range of language phenomena (Steedman, 1996, 2000). A CCG is defined by a lexicon and a set of combinators. The lexicon contains entries that pair words or phrases with CCG categories. For example, the lexical entry $\text{library} \vdash N : \lambda x.library(x)$ in Figure 2 pairs the word 'library' with the CCG category that has syntactic category N and meaning $\lambda x.library(x)$. A CCG parse starts from assigning lexical entries to words and phrases. These are then combined using the set of CCG combinators to build a logical form that captures the meaning of the entire sentence. We use the application, composition, and coordination combinators. Figure 2 shows an example parse.

5 Parsing Sentences to Meanings

The function $\text{GEN}(x, \mathcal{O})$ defines the set of possible derivations for an input sentence x. Each derivation $d = \langle \Pi, M\rangle$ builds a logical form y using constants from the ontology $\mathcal{O}$. Π is a CCG parse tree that maps x to an underspecified logical form l_0. M is an ontological match that maps l_0 onto the fully specified logical form y. This section describes, with reference to the example in Figure 3, the operations used by Π and M.

5.1 Domain Independent Parsing

Domain-independent CCG parse trees Π are built using a predefined set of 56 underspecified lexical categories, 49 domain-independent lexical items, and the combinatory rules introduced in Section 4.

An underspecified CCG lexical category has a syntactic category and a logical form containing no constants from the domain ontology $\mathcal{O}$. Instead, the logical form includes underspecified constants that are typed placeholders which will later be replaced during ontology matching. For example, a noun might be assigned the lexical category $\mathsf{N} : \lambda x.p(x)$, where p is an underspecified $\langle e, t\rangle$-type constant.

During parsing, lexical categories are created dynamically. We manually define a set of POS tags for each underspecified lexical category, and use Wiktionary as a tag dictionary to define the possible POS tags for words and phrases. Each phrase is assigned every matching lexical category. For example, the word 'visit' can be either a verb or a noun in Wiktionary. We accordingly assign it all underspecified categories for the classes, including:

$$\mathsf{N}:\lambda x.p(x) \;\;,\;\; \mathsf{S\backslash NP/NP}:\lambda x\lambda y\exists ev.p(y, x, ev)$$

for nouns and transitive verbs respectively.

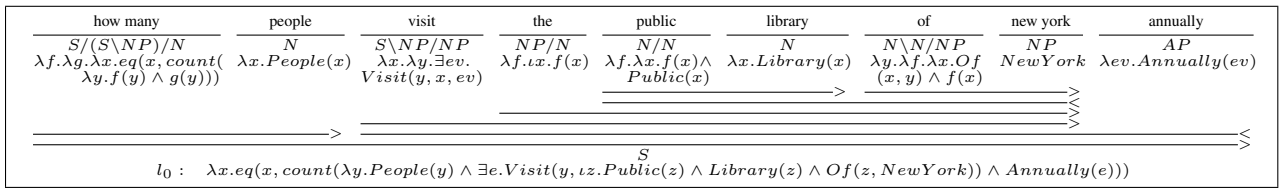
We also define domain-independent lexical items for function words such as 'what,' 'when,' and 'how many,' 'and,' and 'is.' These lexical items pair a word with a lexical category containing only domain-independent constants. For example, $\text{how many} \vdash \mathsf{S/(S\backslash NP)/N} : \lambda f.\lambda g.\lambda x.eq(x, count(\lambda y.f(y) \wedge g(y)))$ contains the function $count$ and the predicate eq.

Figure 3a shows the lexical categories and combinator applications used to construct the underspecified logical form l_0. Underspecified constants in this figure have been labeled with the words that they are associated with for readability.

5.2 Ontological Matching

The second, domain specific, step M maps the underspecified logical form l_0 onto the fully specified logical form y. The mapping from constants in l_0 to constants in y is not one-to-one. For example, in Figure 3, l_0 contains 11 constants but y contains only 2. The ontological match is a sequence of matching operations $M = \langle o_1 \ldots, o_n\rangle$ that can transform the structure of the logical form or replace underspecified constants with constants from $\mathcal{O}$.

(a) **Underspecified CCG parse Π:** Map words onto underspecified lexical categories as described in Section 5.1. Use the CCG combinators to combine lexical categories to give the full underpecified logical form l_0.



(b) **Structure Matching Steps in M:** Use the operators described in Section 5.2.1 and Figure 4 to transform l_0. In each step one of the operators is applied to a subexpression of the existing logical form to generate a modified logical form with a new underspecified constant marked in bold.

$$
\begin{aligned}
l_0 &: \lambda x.eq(x, count(\lambda y.People(y) \wedge \exists e.Visit(y, \iota z.Public(z) \wedge Library(z) \wedge Of(z, NewYork), e) \wedge Annually(e))) \\
l_1 &: \lambda x.eq(x, count(\lambda y.People(y) \wedge \exists e.Visit(y, \textbf{PublicLibraryOfNewYork}, e) \wedge Annually(e))) \\
l_2 &: \lambda x.\textbf{HowManyPeopleVisitAnnually}(x, PublicLibraryOfNewYork)))
\end{aligned}
$$

(c) **Constant Matching Steps in M:** Replace all underspecified constants in the transformed logical form with a similarly typed constant from $\mathcal{O}$, as described in Section 5.2.2. The underspecified constant to be replaced is marked in bold and constants from $\mathcal{O}$ are written in typeset.

$$
\begin{aligned}
& \lambda x.HowManyPeopleVisitAnnually(x, \textbf{PublicLibraryOfNewYork}) \\
l_3 : & \qquad \mapsto \lambda x.HowManyPeopleVisitAnnually(x, \texttt{new\_york\_public\_library}) \\
& \lambda x.\textbf{HowManyPeopleVisitAnnually}(x, \texttt{new\_york\_public\_library}) \\
y : & \qquad \mapsto \lambda x.\texttt{public\_library\_system.annual\_visits}(x, \texttt{new\_york\_public\_library})
\end{aligned}
$$

Figure 3: Example derivation for the query 'how many people visit the public library of new york annually.' Underspecified constants are labelled with the words from the query that they are associated with for readability. Constants from $\mathcal{O}$, written in typeset, are introduced in step (c).

	Operator	Definition and Conditions	Example
a.	Collapse Literal to Constant	$P(a_1, \ldots, a_n) \mapsto c$ s.t. $\texttt{type}(P(a_1, \ldots, a_n)) = \texttt{type}(c)$ $\texttt{type}(c) \in \{e, i\}$ $\texttt{freev}(P(a_1, \ldots, a_n)) = \emptyset$	$\iota z.Public(z) \wedge Library(z) \wedge Of(z, NewYork))$ $\mapsto \quad PublicLibraryOfNewYork$ Input and output have type e. e is allowed in $\mathcal{O}$. Input contains no free variables.
b.	Collapse Literal to Literal	$P(a_1, \ldots, a_n) \mapsto Q(b_1, \ldots, b_m)$ s.t. $\texttt{type}(P(a_1, \ldots, a_n)) = \texttt{type}(Q(b_1, \ldots, b_m))$ $\texttt{type}(Q) \in \{\texttt{type}(c) : c \in \mathcal{O}\}$ $\texttt{freev}(P(a_1, \ldots, a_n)) = \texttt{freev}(Q(b_1, \ldots, b_m))$ $\{b_1, \ldots, b_m\} \in \texttt{subexps}(P(a_1, \ldots, a_n))$	$eq(x, count(\lambda y.People(y) \wedge \exists e.Visit(y,$ $PublicLibraryOfNewYork) \wedge Annually(e)))$ $\mapsto CountPeopleVisitAnnually(x,$ $PublicLibraryOfNewYork)$ Input and output have type t. New constant has type $\langle i, \langle e, t\rangle\rangle$, allowed in $\mathcal{O}$. Input and output contain single free variable x. Arguments of output literal are subexpressions of input.
c.	Split Literal	$P(a_1, \ldots, a_k, x, a_{k+1}, \ldots, a_n)$ $\mapsto Q(b_1, \ldots, x, \ldots b_n) \wedge Q''(c_1, \ldots, x, \ldots c_m)$ s.t. $\texttt{type}(P(\ldots)) = t$ $\{\texttt{type}(Q), \texttt{type}(Q'')\} \in \{\texttt{type}(c) : c \in \mathcal{O}\}$ $\{b_1, \ldots, b_n, c_1, \ldots, c_m\} = \{a_1, \ldots, a_n\}$	$Dedicate(Mozart, Haydn, ev)$ $\mapsto Dedicate(Mozart, ev) \wedge Dedicate''(Haydn, ev)$ Input has type t. This matches output type by definition. New constants have allowed type $\langle e, \langle ev, t\rangle\rangle$. All arguments of input literal are preserved in output.

Figure 4: Definition of the operations used to transform the structure of the underspecified logical form l_0 to match the ontology $\mathcal{O}$. The function $\texttt{type}(c)$ calculates a constant c's type. The function $\texttt{freev}(lf)$ returns the set of variables that are free in lf (not bound by a lambda term or quantifier). The function $\texttt{subexps}(lf)$ generates the set of all subexpressions of the lambda calculus expression lf.

5.2.1 Structure Matching

Three structure matching operators, illustrated in Figure 4, are used to collapse or expand literals in l_0. Collapses merge a subexpression from l_0 to create a new underspecified constant, generating a logical form with fewer constants. Expansions split a subexpression from l_0 to generate a new logical form containing one extra constant.

Collapsing Operators The collapsing operator defined in Figure 4a merges all constants in a literal to generate a single constant of the same type. This operator is used to map $\iota z.Public(z) \wedge Library(z) \wedge Of(z,NewYork)$ to $PublicLibraryOfNewYork$ in Figure 3b. Its operation is limited to entity typed expressions that do not contain free variables.

The operator in Figure 4b, in contrast, can be used to collapse the expression $eq(x,count(\lambda y.People(y) \wedge \exists e.Visit(y,PublicLibraryOfNewYork,e)) \wedge Annually(e)))$, which contains free variable x onto a new expression $CountPeopleVisitAnnually(x,PublicLibraryOfNewYork)$. This is only possible when the type of the newly created constant is allowed in $\mathcal{O}$ and the variable x is free in the output expression. Subsets of conjuncts can be collapsed using the operator in Figure 4b by creating ad-hoc conjunctions that encapsulate them. Disjunctions are treated similarly.

Performing collapses on the underspecified logical form allows non-contiguous phrases to be represented in the collapsed form. In this example, the logical form representing the phrase 'how many people visit' has been merged with the logical form representing the non-adjacent adverb 'annually.' This generates a new underspecified constant that can be mapped onto the Freebase relation `public_library_system_annual_visits` that relates to both phrases.

The collapsing operations preserve semantic type, ensuring that all logical forms generated by the derivation sequence are well typed. The full set of allowed collapses of l_0 is given by the transitive closure of the collapsing operations. The size of this set is limited by the number of constants in l_0, since each collapse removes at least one constant. At each step, the number of possible collapses is polynomial in the number of constants in l_0 and exponential in the arity of the most complex type in $\mathcal{O}$. For domains of interest this arity is unlikely to be high and for triple stores such as Freebase it is 2.

Expansion Operators The fully specified logical form y can contain constants relating to multiple words in x. It can also use multiple constants to represent the meaning of a single word. For example, Freebase does not contain a relation representing the concept 'daughter', instead using two relations representing 'female' and 'child'. The expansion operator in Figure 4c allows a single predicate to be split into a pair of conjoined predicates sharing an argument variable. For example, in Figure 1, the constant for 'dedicate' is split in two to match its representation in Freebase. Underspecified constants from l_0 can be split once. For the experiments in Section 8, we constrain the expansion operator to work on event modifiers but the procedure generalizes to all predicates.

5.2.2 Constant Matching

To build an executable logical form y, all underspecified constants must be replaced with constants from $\mathcal{O}$. This is done through a sequence of constant replacement operations, each of which replaces a single underspecified constant with a constant of the same type from $\mathcal{O}$. Two example replacements are shown in Figure 3c. The output from the last replacement operation is a fully specified logical form.

6 Building and Scoring Derivations

This section introduces a dynamic program used to construct derivations and a linear scoring model.

6.1 Building Derivations

The space of derivations is too large to explicitly enumerate. However, each logical form (both final and interim) can be constructed with many different derivations, and we only need to find the highest scoring one. This allows us to develop a simple dynamic program for our two-stage semantic parser.

We use a CKY style chart parser to calculate the k-best logical forms output by parses of x. We then store each interim logical form generated by an operator in M once in a hyper-graph chart structure. The branching factor of this hypergraph is polynomial in the number of constants in l_0 and linear in the size of $\mathcal{O}$. Subsequently, there are too many possible logical forms to enumerate explicitly; we

prune as follows. We allow the top N scoring ontological matches for each original subexpression in l_0 and remove matches that differ from score from the maximum scoring match by more than a threshold τ. When building derivations, we apply constant matching operators as soon as they are applicable to new underspecified constants created by collapses and expansions. This allows the scoring function used by the pruning strategy to take advantage of all features defined in Section 7.2.

6.2 Ranking Derivations

Given feature vector ϕ and weight vector θ, the score of a derivation $d = \langle \Pi, M \rangle$ is a linear function that decomposes over the parse tree Π and the individual ontology-matching steps o.

$$\begin{aligned} \text{SCORE}(d) &= \phi(d)\theta \\ &= \phi(\Pi)\theta + \sum_{o \in M} \phi(o)\theta \end{aligned} \quad (1)$$

The function $\text{PARSE}(x, \mathcal{O})$ introduced as our goal in Section 2 returns the logical form associated with the highest scoring derivation of x:

$$\text{PARSE}(x, \mathcal{O}) = \arg \max_{d \in \text{GEN}(x, \mathcal{O})} (\text{SCORE}(d))$$

The features and learning algorithm used to estimate θ are defined in the next section.

7 Learning

This section describes an online learning algorithm for question-answering data, along with the domain-independent feature set.

7.1 Learning Model Parameters

Our learning algorithm estimates the parameters θ from a set $\{(x_i, a_i) : i = 1 \ldots n\}$ of questions x_i paired with answers a_i from the knowledge base $\mathcal{K}$. Each derivation d generated by the parser is associated with a fully specified logical form $y = \text{YIELD}(d)$ that can be executed in $\mathcal{K}$. A derivation d of x_i is correct if $\text{EXEC}(\text{YIELD}(d), \mathcal{K}) = a_i$. We use a perceptron to estimate a weight vector θ that support a separation of γ between correct and incorrect answers. Figure 5 presents the learning algorithm.

Input: Q/A pairs $\{(x_i, a_i) : i = 1 \ldots n\}$; Knowledge base $\mathcal{K}$; Ontology $\mathcal{O}$; Function $\text{GEN}(x, \mathcal{O})$ that computes derivations of x; Function $\text{YIELD}(d)$that returns logical form yield of derivation d; Function $\text{EXEC}(y, \mathcal{K})$ that calculates execution of y in $\mathcal{K}$; Margin γ; Number of iterations T.

Output: Linear model parameters θ.

Algorithm:

For $t = 1 \ldots T, i = 1 \ldots n$:

$C = \{d : d \in \text{GEN}(x_i, \mathcal{O}); \text{EXEC}(\text{YIELD}(d), \mathcal{K}) = a_i\}$

$W = \{d : d \in \text{GEN}(x_i, \mathcal{O}); \text{EXEC}(\text{YIELD}(d), \mathcal{K}) \neq a_i\}$

$C^* = \arg\max_{d \in C}(\phi(d)\theta)$

$W^* = \{d : d \in W;\ \exists c \in C^*\ s.t.\ \phi(c)\theta - \phi(d)\theta < \gamma)\}$

If $|C^*| > 0 \wedge |W^*| > 0$:

$\theta = \theta + \frac{1}{|C^*|} \sum_{c \in C^*} \phi(c) - \frac{1}{|W^*|} \sum_{e \in W^*} \phi(e)$

Figure 5: Parameter estimation from Q/A pairs.

7.2 Features

The feature vector $\phi(d)$ introduced in Section 6.2 decomposes over each of the derivation steps in d.

CCG Parse Features Each lexical item in Π has three indicator features. The first indicates the number of times each underspecified category is used. For example, the parse in Figure 3a uses the underspecified category $\mathsf{N} : \lambda x.p(x)$ twice. The second feature indicates (word, category) pairings — e.g. that $\mathsf{N} : \lambda x.p(x)$ is paired with 'library' and 'public' once each in Figure 3a. The final lexical feature indicates (part-of-speech, category) pairings for all parts of speech associated with the word.

Structural Features The structure matching operators (Section 5.2.1) in M generate new underspecified constants that define the types of constants in the output logical form y. These operators are scored using features that indicate the type of each complex-typed constant present in y and the identity of domain-independent functional constants in y. The logical form y generated in Figure 3 contains one complex typed constant with type $\langle i, \langle e, t \rangle \rangle$ and no domain-independent functional constants. Structural features allow the model to adapt to different knowledge bases $\mathcal{K}$. They allow it to determine, for example, whether a numeric quantity such as 'population' is likely to be explicitly listed in $\mathcal{K}$ or if it should be computed with the $count$ function.

Lexical Features Each constant replacement operator (Section 5.2.2) in M replaces an underspec-

ified constant c_u with a constant $c_\mathcal{O}$ from $\mathcal{O}$. The underspecified constant c_u is associated with the sequence of words $\vec{w}_u$ used in the CCG lexical entries that introduced it in Π. We assume that each of the constants c_O in $\mathcal{O}$ is associated with a string label $\vec{w}_O$. This allows us to introduce five domain-independent features that measure the similarity of $\vec{w}_u$ and $\vec{w}_O$.

The feature $\phi_{np}(c_u, c_O)$ signals the replacement of an entity-typed constant c_u with entity c_O that has label $\vec{w}_u$. For the second example in Figure 1 this feature indicates the replacement of the underspecified constant associated with the word 'mozart' with the Freebase entity `mozart`. Stem and synonymy features $\phi_{stem}(c_u, c_O)$ and $\phi_{syn}(c_u, c_O)$ signal the existence of words $w_u \in \vec{w_u}$ and $w_u \in \vec{w_O}$ that share a stem or synonym respectively. Stems are computed with the Porter stemmer and synonyms are extracted from Wiktionary. A single Freebase specific feature $\phi_{fp:stem}(c_u, c_O)$ indicates a word stem match between $w_u \in \vec{w_u}$ and the word filling the most specific position in $\vec{w_u}$ under Freebase's hierarchical naming schema.

A final feature $\phi_{gl}(c_u, c_O)$ calculates the overlap between Wiktionary definitions for $\vec{w_u}$ and $\vec{w_O}$. Let $\texttt{gl}(w)$ be the Wiktionary definition for w. Then:

$$\phi_{gl}(c_u, c_O) = \sum_{w_u \in \vec{w_u}; w_O \in \vec{w_O}} \frac{2 \cdot |\texttt{gl}(w_O) \cap \texttt{gl}(w_c)|}{|\vec{w_O}| \cdot |\vec{w_u}| \cdot |\texttt{gl}(w_O)| + |\texttt{gl}(w_c)|}$$

Domain-indepedent lexical features allow the model to reason about the meaning of unseen words. In small domains, however, the majority of word usages may be covered by training data. We make use of this fact in the GeoQuery domain with features $\phi_m(c_u, c_O)$ that indicate the pairing of $\vec{w_u}$ with c_O.

Knowledge Base Features Guided by the observation that we generally want to create queries y which have answers in knowledge base $\mathcal{K}$, we define features to signal whether each operation could build a logical form y with an answer in $\mathcal{K}$.

If a predicate-argument relation in y does not exist in $\mathcal{K}$, then the execution of y against $\mathcal{K}$ will not return an answer. Two features indicate whether predicate-argument relations in y exist in $\mathcal{K}$. $\phi_{direct}(y, \mathcal{K})$ indicates predicate-argument applications in y that exists in $\mathcal{K}$. For example, if the application of `dedicated_by` to `mozart` in Figure 1 exists in Freebase, $\phi_{direct}(y, \mathcal{K})$ will fire. $\phi_{join}(y, \mathcal{K})$ indicates entities separated from a predicate by one join in y, such as `mozart` and `dedicated_to` in Figure 1, that exist in the same relationship in $\mathcal{K}$.

If two predicates that share a variable in y do not share an argument in that position in $\mathcal{K}$ then the execution of y against $\mathcal{K}$ will fail. The predicate-predicate $\phi_{pp}(y, \mathcal{K})$ feature indicates pairs of predicates that share a variable in y but cannot occur in this relationship in $\mathcal{K}$. For example, since the subject of the Freebase property `date_of_birth` does not take arguments of type `location`, $\phi_{pp}(y, \mathcal{K})$ will fire if y contains the logical form $\lambda x \lambda y.\texttt{date\_of\_birth}(x, y) \wedge \texttt{location}(x)$.

Both the predicate-argument and predicate-predicate features operate on subexpressions of y. We also define the execution features: $\phi_{emp}(y, \mathcal{K})$ to signal an empty answer for y in $\mathcal{K}$; $\phi_0(y, \mathcal{K})$ to signal a zero-valued answer created by counting over an empty set; and $\phi_1(y, \mathcal{K})$ to signal a one-valued answer created by counting over a singleton set.

As with the lexical cues, we use knowledge base features as soft constraints since it is possible for natural language queries to refer to concepts that do not exist in $\mathcal{K}$.

8 Experimental Setup

Data We evaluate performance on the benchmark GeoQuery dataset (Zelle and Mooney, 1996), and a newly introduced Freebase Query (FQ) dataset (Cai and Yates, 2013a). FQ contains 917 questions labeled with logical form meaning representations for querying Freebase. We gathered question answer labels by executing the logical forms against Freebase, and manually correcting any inconsistencies.

Freebase (Bollacker et al., 2008) is a large, collaboratively authored database containing almost 40 million entities and two billion facts, covering more than 100 domains. We filter Freebase to cover the domains contained in the FQ dataset resulting in a database containing 18 million entities, 2072 relations, 635 types, 135 million facts and 81 domains, including for example film, sports, and business. We use this schema to define our target domain, allowing for a wider variety of queries than could be encoded with the 635 collapsed relations previously used to label the FQ data.

We report two different experiments on the FQ data: test results on the existing 642/275 train/test split and domain adaptation results where the data is split three ways, partitioning the topics so that the logical meaning expressions do not share any symbols across folds. We report on the standard 600/280 training/test split for GeoQuery.

Parameter Initialization and Training We initialize weights for ϕ_{np} and ϕ_{direct} to 10, and weights for ϕ_{stem} and ϕ_{join} to 5. This promotes the use of entities and relations named in sentences. We initialize weights for ϕ_{pp} and ϕ_{emp} to -1 to favour logical forms that have an interpretation in the knowledge base $\mathcal{K}$. All other feature weights are initialized to 0. We run the training algorithm for one iteration on the Freebase data, at which point performance on the development set had converged. This fast convergence is due to the very small number of matching parameters used (5 lexical features and 8 $\mathcal{K}$ features). For GeoQuery, we include the larger domain specific feature set introduced in Section 7.2 and train for 10 iterations. We set the pruning parameters from Section 6.1 as follows: $k = 5$ for Freebase, $k = 30$ for GeoQuery, $N = 50$, $\tau = 10$.

Comparison Systems We compare performance to state-of-the-art systems in both domains. On GeoQuery, we report results from DCS (Liang et al., 2011) without special initialization (DCS) and with an small hand-engineered lexicon (DCS with L^+). We also include results for the FUBL algorithm (Kwiatkowski et al., 2011), the CCG learning approach that is most closely related to our work. On FQ, we compare to Cai and Yates (2013a) (CY13).

Evaluation We evaluate by comparing the produced question answers to the labeled ones, with no partial credit. Because the parser can fail to produce a complete query, we report recall, the percent of total questions answered correctly, and precision, the percentage of produced queries with correct answers. CY13 and FUBL report fully correct logical forms, which is a close proxy to our numbers.

9 Results

Quantitative Analysis For FQ, we report results on the test set and in the cross-domain setting, as defined in Section 8. Figure 6 shows both results. Our approach outperforms the previous state of the art, achieving a nine point improvement in test recall, while not requiring labeled logical forms in training. We also see consistent improvements on both scenarios, indicating that our approach is generalizing well across topic domains. The learned ontology matching model is able to reason about previously unseen ontological subdomains as well as if it was provided explicit, in-domain training data.

Setting	System	R	P	F1
Test	Our Approach	68.0	76.7	72.1
	CY13	59	67	63
Cross Domain	Our Approach	67.9	73.5	71.5
	CY13	60	69	65

Figure 6: Results on the FQ dataset.

	R	P	F1
All Features	68.6	72.0	70.3
Without Wiktionary	67.2	70.7	68.9
Without $\mathcal{K}$ Features	61.8	62.5	62.1

Figure 7: Ablation Results

We also performed feature ablations with 5-fold cross validation on the training set, as seen in Figure 7. Both the Wiktionary features and knowledge base features were helpful. Without the Wiktionary features, the model must rely on word stem matches which, in combination with graph constraints, can still recover many of the correct queries. However, without the knowledge base constraints, the model produces many queries that return empty answers, and significantly impacts overall performance.

For GeoQuery, we report test results in Figure 8. Our approach outperforms the most closely related CCG model (FUBL) and DCS without initialization, but falls short of DCS with a small hand-built initial lexicon. Given the small size of the test set, it is fair to say that all algorithms are performing at state-of-the-art levels. This result demonstrates that our ap-

	Recall
FUBL	88.6
DCS	87.9
DCS with L^+	91.1
Our Approach	89.0

Figure 8: GeoQuery Results

Parse Failures (20%)		
1.	Query	in what year did motorola have the most revenue
2	Query	on how many projects was james walker a design engineer
Structural Matching Failure (30%)		
3.	Query	how many children does jerry seinfeld have
	Labeled	λx.eq(x, count(λy.people.person.children(jerry_seinfeld, y)))
	Predicted	λx.eq(x, count(λy.people.person.children(y, jerry_seinfeld)))
Incomplete Database (10%)		
4.	Query	how many countries participated in the 2006 winter olympics
	Labeled	λy.olympics.olympic_games.number_of_countries(2006_winter_olympics, y)
	Predicted	λy.eq(y, count(λy.olympic_participation_country.olympics_participated_in(x, 2006_winter_olympics)))
5.	Query	what programming languages were used for aol instant messenger
	Labeled	λy.computer.software.languages_used(aol_instant_messenger, y)
	Predicted	λy.computer.software.languages_used(aol_instant_messenger, y) $\wedge$ computer.programming_language(y)
Lexical Ambiguity (35%)		
6.	Query	when was the frida kahlo exhibit at the philadelphia art museum
	Labeled	λy.$\exists$x.exhibition_run.exhibition(x, frida_kahlo)$\wedge$ exhibition_venue.exhibitions_at(philadelphia_art_museum, x) $\wedge$ exhibition_run.opened_on(x, y)
	Predicted	λy.$\exists$x.exhibition_run.exhibition(x, frida_kahlo)$\wedge$ exhibition_venue.exhibitions_at(philadelphia_art_museum, x) $\wedge$ exhibition_run.closed_on(x, y)

Figure 9: Example error cases, with associated frequencies, illustrating system output and gold standard references. 5% of the cases were miscellaneous or otherwise difficult to categorize.

proach can handle the high degree of lexical ambiguity in the FQ data, without sacrificing the ability to understanding the rich, compositional phenomena that are common in the GeoQuery data.

Qualitative Analysis We also did a qualitative analysis of errors in the FQ domain. The model learns to correctly produce complex forms that join multiple relations. However, there are a number of systematic error cases, grouped into four categories as seen in Figure 9.

The first and second examples show parse failures, where the underspecified CCG grammar did not have sufficient coverage. The third shows a failed structural match, where all of the correct logical constants are selected, but the argument order is reversed for one of the literals. The fourth and fifth examples demonstrate a failures due to database incompleteness. In both cases, the predicted queries would have returned the same answers as the gold-truth ones if Freebase contained all of the required facts. Developing models that are robust to database incompleteness is a challenging problem for future work. Finally, the last example demonstrates a lexical ambiguity, where the system was unable to determine if the query should include the opening date or the closing date for the exhibit.

10 Conclusion

We considered the problem of learning domain-independent semantic parsers, with application to QA against large knowledge bases. We introduced a new approach for learning a two-stage semantic parser that enables scalable, on-the-fly ontological matching. Experiments demonstrated state-of-the-art performance on benchmark datasets, including effective generalization to previously unseen words.

We would like to investigate more nuanced notions of semantic correctness, for example to support many of the essentially equivalent meaning representations we found in the error analysis. Although we focused exclusively on QA applications, the general two-stage analysis approach should allow for the reuse of learned grammars across a number of different domains, including robotics or dialog applications, where data is more challenging to gather.

11 Acknowledgements

This research was supported in part by DARPA under the DEFT program through the AFRL (FA8750-13-2-0019) and the CSSG (N11AP20020), the ARO (W911NF-12-1-0197), the NSF (IIS-1115966), and by a gift from Google. The authors thank Anthony Fader, Nicholas FitzGerald, Adrienne Wang, Daniel Weld, and the anonymous reviewers for their helpful comments and feedback.

References

Alshawi, H. (1992). *The core language engine*. The MIT Press.

Artzi, Y. and Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In *Proceedings of the Conference on Semantics in Text Processing*.

Cai, Q. and Yates, A. (2013a). Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Cai, Q. and Yates, A. (2013b). Semantic parsing freebase: Towards open-domain semantic parsing. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Chen, D. and Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Clark, S. and Curran, J. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Davidson, D. (1967). The logical form of action sentences. *Essays on actions and events*, pages 105–148.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2004). Ontology matching: A machine learning approach. In *Handbook on ontologies*. Springer.

Euzenat, J., Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*. Springer.

Fader, A., Zettlemoyer, L., and Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Goldwasser, D. and Roth, D. (2011). Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Grosz, B. J., Appelt, D. E., Martin, P. A., and Pereira, F. (1987). TEAM: An experiment in the design of transportable natural language interfaces. *Artificial Intelligence*, 32(2):173–243.

Hobbs, J. R. (1985). Ontological promiscuity. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*.

Jones, B. K., Johnson, M., and Goldwater, S. (2012). Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association of Computational Linguistics*.

Kate, R. and Mooney, R. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1(2).

Krishnamurthy, J. and Mitchell, T. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Kushman, N. and Barzilay, R. (2013). Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Kwiatkowski, T., Goldwater, S., Zettlemoyer, L., and Steedman, M. (2012). A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. *Proceedings of the Conference of the European Chapter of the Association of Computational Linguistics*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012). A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning*.

Muresan, S. (2011). Learning for deep language understanding. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press.

Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A., Gerber, D., and Cimiano, P. (2012). Template-based question answering over RDF data. In *Proceedings of the International Conference on World Wide Web*.

Wong, Y. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., and Weikum, G. (2012). Natural language questions for the web of data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, L. and Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Zhang, C., Hoffmann, R., and Weld, D. S. (2012). Ontological smoothing for relation extraction with minimal supervision. In *Proceeds of the Conference on Artificial Intelligence*.

Classifying Message Board Posts with an Extracted Lexicon of Patient Attributes

Ruihong Huang and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
{huangrh, riloff}@cs.utah.edu

Abstract

The goal of our research is to distinguish veterinary message board posts that describe a case involving a specific patient from posts that ask a general question. We create a text classifier that incorporates automatically generated attribute lists for veterinary patients to tackle this problem. Using a small amount of annotated data, we train an information extraction (IE) system to identify veterinary patient attributes. We then apply the IE system to a large collection of unannotated texts to produce a lexicon of veterinary patient attribute terms. Our experimental results show that using the learned attribute lists to encode patient information in the text classifier yields improved performance on this task.

1 Introduction

Our research focuses on the problem of classifying message board posts in the domain of veterinary medicine. Most of the posts in our corpus discuss a case involving a specific patient, which we will call *patient-specific* posts. But there are also posts that ask a general question, for example to seek advice about different medications, information about new procedures, or how to perform a test. Our goal is to distinguish the patient-specific posts from general posts so that they can be automatically routed to different message board folders.

Distinguishing patient-specific posts from general posts is a challenging problem for two reasons. First, virtually any medical topic can appear in either type of post, so the vocabulary is very similar. Second, a highly skewed distribution exists between patient-specific posts and general posts. Almost 90% of the posts in our data are about specific patients.

With such a highly skewed distribution, it would seem logical to focus on recognizing instances of the minority class. But the distinguishing characteristic of a general post is the *absence* of a patient. Two nearly identical posts belong in different categories if one mentions a patient and the other does not. Consequently, our aim is to create features that identify references to a specific patient and use these to more accurately distinguish the two types of posts.

Our research explores the use of information extraction (IE) techniques to automatically identify common attributes of veterinary patients, which we use to encode patient information in a text classifier. Our approach involves three phases. First, we train a conditional random fields (CRF) tagger to identify seven common types of attributes that are often ascribed to veterinary patients: SPECIES/BREED, NAME, AGE, GENDER, WEIGHT, POSSESSOR, and DISEASE/SYMPTOM. Second, we apply the CRF tagger to a large set of unannotated message board posts, collect its extractions, and harvest the most frequently extracted terms to create a *Veterinary Patient Attribute (VPA) Lexicon*.

Finally, we define three types of features that exploit the harvested VPA lexicon. These features represent the patient attribute terms, types, and combinations of them to help the classifier determine whether a post is discussing a specific patient. We conduct experiments which show that the extracted patient attribute information improves text classification performance on this task.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1557–1562,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Our work demonstrates the use of information extraction techniques to benefit a text classification application. There has been a great deal of research on text classification (e.g., (Borko and Bernick, 1963; Hoyle, 1973; Joachims, 1998; Nigam et al., 2000; Sebastiani, 2002)), which most commonly has used bag-of-word features. Researchers have also investigated clustering (Baker and McCallum, 1998), Latent Semantic Indexing (LSI) (Zelikovitz and Hirsh, 2001), Latent Dirichlet Allocation (LDA) (Br et al., 2008) and string kernels (Lodhi et al., 2001). Information extraction techniques have been used previously to create richer features for event-based text classification (Riloff and Lehnert, 1994) and web page classification (Furnkranz et al., 1998). Semantic information has also been incorporated for text classification. However, most previous work relies on existing semantic resources, such as Wordnet (Scott and Stan, 1998; Bloehdorn and Hotho, 2006) or Wikipedia (Wang et al., 2009).

There is also a rich history of automatic lexicon induction from text corpora (e.g., (Roark and Charniak, 1998; Riloff and Jones, 1999; McIntosh and Curran, 2009)), Wikipedia (e.g., (Vyas and Pantel, 2009)), and the Web (e.g., (Etzioni et al., 2005; Kozareva et al., 2008; Carlson et al., 2010)). The novel aspects of our work are in using an IE tagger to harvest a domain-specific lexicon from unannotated texts, and using the induced lexicon to encode domain-specific features for text classification.

3 Text Classification with Extracted Patient Attributes

This resesarch studies message board posts from the Veterinary Information Network (VIN), which is a web site (www.vin.com) for professionals in veterinary medicine. VIN hosts forums where veterinarians discuss medical issues, challenging cases, etc. We observed that patient-specific veterinary posts almost always include some basic facts about the patient, such as the animal's breed, age, or gender. It is also common to mention the patient's owner (e.g., *"a new client's cat"*) or a disease or symptom that the patient has (e.g., *"a diabetic cat"*). General posts almost never contain this information.

Although some of these terms can be found in existing resources such as Wordnet (Miller, 1990), our veterinary message board posts are filled with informal and unconventional vocabulary. For example, one might naively assume that *"male"* and *"female"* are sufficient to identify gender. But the gender of animals is often revealed by describing their spayed/neutered status, often indicated with shorthand notations. For example, *"m/n"* means male and neutered, *"fs"* means female spayed, *"castrated"* means neutered and implies male. Shorthand terms and informal jargon are also frequently used for breeds (e.g., *"doxy"* for dachsund, *"labx"* for labrador cross, *"gshep"* for German Shepherd) and ages (e.g., *"3-yr-old"*, *"3yo"*, *"3mo"*). A particularly creative age expression describes an animal as (say) *"a 1999 model"* (i.e., born in 1999). To recognize the idiosyncratic vocabulary in these texts, we use information extraction techniques to identify terms corresponding to seven attributes of veterinary patients: SPECIES/BREED, NAME, AGE, WEIGHT, GENDER, POSSESSOR, and DISEASE/SYMPTOM.

Figure 1 illustrates our overall approach, which consists of three steps. First, we train a sequential IE tagger to label veterinary patient attributes using supervised learning. Second, we apply the tagger to 10,000 unannotated message board posts to automatically create a Veterinary Patient Attribute (VPA) Lexicon. Third, we use the VPA Lexicon to encode patient attribute features in a document classifier.

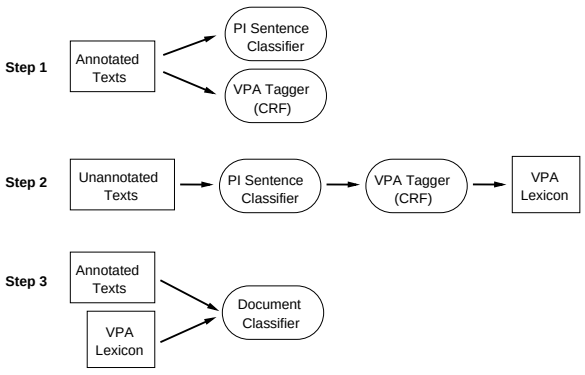


Figure 1: Flowchart for Creating a Patient-Specific vs. General Document Classifier

3.1 Patient Attribute Tagger

The first component of our system is a tagger that labels veterinary patient attributes. To train the tagger, we need texts labeled with patient attributes.

The message board posts can be long and tedious to read (i.e., they are often filled with medical history and test results), so manually annotating every word would be arduous. However, the patient is usually described at the beginning of a post, most commonly in 1-2 "introductory" sentences. Therefore we adopted a two stage process, both for manual and automatic tagging of patient attributes.

First, we created annotation guidelines to identify "patient introductory" (PI) sentences, which we defined as sentences that introduce a patient to the reader by providing a general (non-medical) description of the animal (e.g., *"I was presented with a m/n Siamese cat that is lethargic."*) We randomly selected 300 posts from our text collection and asked two human annotators to manually identify the PI sentences. We measured their inter-annotator agreement using Cohen's kappa (κ) and their agreement was κ=.93. The two annotators then adjudicated their differences to create our gold standard set of PI sentence annotations. 269 of the 300 posts contained at least one PI sentence , indicating that 89.7% of the posts mention a specific patient. The remaining 31 posts (10.3%) are general in nature.

Second, the annotators manually labeled the words in these PI sentences with respect to the 7 veterinary patient attributes. On 50 randomly selected texts, the annotators achieved an inter-annotator agreement of κ = .89. The remaining 250 posts were then annotated with patient attributes (in the PI sentences), providing us with gold standard attribute annotations for all 300 posts. To illustrate, the sentence below would have the following labels:

$$\text{Daisy}_{name} \text{ is a } \text{10yr}_{age} \text{ old}_{age} \text{ lab}_{species}$$

We used these 300 annotated posts to train both a PI sentence classifier and a patient attribute tagger. The PI sentence classifier is a support vector machine (SVM) with a linear kernel (Keerthi and DeCoste, 2005), unigram and bigram features, and binary feature values. The PI sentences are the positive training instances, and the sentences in the general posts are negative training instances.

For the tagger, we trained a single conditional random fields (CRF) model to label all 7 types of patient attributes using the CRF++ package (Lafferty et al., 2001). We defined features for the word string and the part-of-speech tags of the targeted word, two words on its left, and two words on its right.

Given new texts to process, we first apply the PI sentence classifier to identify sentences that introduce a patient. These sentences are given to the patient attribute tagger, which labels the words in those sentences for the 7 patient attribute categories.

To evaluate the performance of the patient attribute tagger, we randomly sampled 200 of the 300 annotated documents to use as training data and used the remaining 100 documents for testing. For this experiment, we only applied the CRF tagger to the gold standard PI sentences, to eliminate any confounding factors from the PI sentence classifier. Table 1 shows the performance of the CRF tagger in terms of Recall (%), Precision (%), and F Score (%). Its precision is consistently high, averaging 91% across all seven attributes. But the average recall is only 47%, with only one attribute (AGE) achieving recall $\geq$ 80%. Nevertheless, the CRF's high precision justifies our plan to use the CRF tagger to harvest additional attribute terms from a large collection of unannotated texts. As we will see in Section 4, the additional terms harvested from the unannotated texts provide substantially more attribute information for the document classifier to use.

Attribute	Rec	Prec	F
SPECIES/BREED	59	93	72
NAME	62	100	76
POSSESSOR	12	100	21
AGE	80	91	85
GENDER	59	81	68
WEIGHT	19	100	32
DISEASE/SYMPTOM	35	73	47
Average	47	91	62

Table 1: Patient Attribute Tagger Evaluation

3.2 Creating a Veterinary Patient Attribute (VPA) Lexicon

The patient attribute tagger was trained with supervised learning, so its ability to recognize important words is limited by the scope of its training set. Since we had an additional 10,000 unannotated veterinary message board posts, we used the tagger to acquire a large lexicon of patient attribute terms.

We applied the PI sentence classifier to all 10,000 texts and then applied the patient attribute tagger to each PI sentence. The patient attribute tagger is not

perfect, so we assumed that words tagged with the same attribute value at least five times[1] are most likely to be correct and harvested them to create a veterinary patient attribute (VPA) lexicon. This produced a VPA lexicon of 592 words. Table 2 shows examples of learned terms for each attribute, with the total number of learned words in parentheses.

Species/Breed (177): DSH, Schnauzer, kitty, Bengal, pug, Labrador, siamese, Shep, miniature, golden, lab, Spaniel, Westie, springer, Chow, cat, Beagle, Mix, ...
Name (53): Lucky, Shadow, Toby, Ginger, Boo, Max, Baby, Buddy, Tucker, Gracie, Maggie, Willie, Tiger, Sasha, Rusty, Beau, Kiki, Oscar, Harley, Scooter, ...
Age (59): #-year, adult, young, YO, y/o, model, wk, y.o., yr-old, yrs, y, #-yr, #-month, #m, mo, mth, ...
Gender (39): F/s, speyed, neutered, spayed, N/M, FN, CM, F, mc, mn, SF, male, fs, M/N, Female, S, S/F, m/n, m/c, intact, M, NM, castrated, ...
Weight (5): lb, lbs, pound, pounds, kg
Possessor (7): my, owner, client, technician, ...
Disease/Symptom (252): abscess, fever, edema, hepatic, inappetance, sneezing, blindness, pain, persistent, mass, insufficiency, acute, poor, ...

Table 2: Examples from the Induced VPA Lexicon

3.3 Text Classification with Patient Attributes

Our ultimate goal is to incorporate patient attribute information into a text classifier to help it distinguish between patient-specific posts and general posts. We designed three sets of features:

Attribute Types: We create one feature for each attribute type, indicating whether a word of that attribute type appeared or not.

Attribute Types with Neighbor: For each word labeled as a patient attribute, we create two features by pairing its Attribute Type with a preceding or following word. For example, given the sentence: *"The tiny Siamese kitten was lethargic."*, if "Siamese" has attribute type SPECIES then we create two features: <*tiny*, SPECIES> and <SPECIES, *kitten*>.

Attribute Pairs: We create features for all pairs of patient attribute words that occur in the same sentence. For each pair, we create one feature representing the words themselves and one feature representing the attribute types of the words.

4 Evaluation

To create a blind test set for evaluation, our annotators labeled an additional 500 posts as *patient-specific* or *general*. Specifically, they labeled those 500 posts with PI sentences. The absence of a PI sentence meant that the post was general. Of the 500 texts, 48 (9.6%) were labeled as general posts. We evaluated the performance of the PI sentence classifier on this test set and found that it achieved 88% accuracy at identifying patient introductory sentences.

We then conducted a series of experiments for the document classification task: distinguishing patient-specific message board posts from general posts. All of our experiments used support vector machine (SVM) classifiers with a linear kernel, and ran 10-fold cross validation on our blind test set of 500 posts. We report Recall (%), Precision (%), and F score (%) results for the patient-specific posts and general posts separately, and for the macro-averaged score across both classes. For the sake of completeness, we also show overall Accuracy (%) results. However, we will focus attention on the results for the general posts, since our main goal is to improve performance at recognizing this minority class.

As a baseline, we created SVM classifiers using unigram features.[2] We tried binary, frequency, and tf-idf feature values. The first three rows of Table 3 show that binary feature values performed the best, yielding a macro-averaged F score of 81% but identifying only 54% of the general posts.

The middle section of Table 3 shows the performance of SVM classifiers using our patient attribute features. We conducted three experiments: applying the CRF tagger to PI sentences (per its design), and labeling words with the VPA lexicon either on all sentences or only on PI sentences (as identified by the PI sentence classifier). The CRF features produced extremely low recall and precision on the general posts. The VPA lexicon performed best when applied only to PI sentences and produced much higher recall than all of the other classifiers, although with lower precision than the two

[1] After our text classification experiments were done, we reran the experiments with the unigrams+lexicon classifier using thresholds ranging from 1 to 10 for lexicon creation, just to see how much difference this threshold made. We found that values ≥ 5 produced nearly identical classification results.

[2] We also tried unigrams + bigrams, but they did not perform better.

Method	Patient-Specific Posts			General Posts			Macro Avg			
	Rec	Prec	F	Rec	Prec	F	Rec	Prec	F	Acc
Unigram Features										
Unigrams (freq)	96	96	96	58	60	59	77	76	77	92
Unigrams (tf-idf)	99	93	96	33	84	48	66	89	76	93
Unigrams (binary)	98	95	97	54	79	64	76	87	81	94
Patient Attribute Features										
CRF Features (PI Sents)	99	91	95	02	25	04	51	58	54	90
VPA Lexicon Features (All Sents)	96	96	96	60	63	62	78	79	79	93
VPA Lexicon Features (PI Sents)	96	98	97	81	66	73	88	82	85	94
Unigram & Patient Attribute Features										
CRF Features (PI Sents)	97	96	97	60	71	65	79	83	81	94
VPA Lexicon Features (PI Sents)	98	98	**98**	79	78	**78**	88	88	**88**	**96**

Table 3: Experimental Results

best unigram-based SVMs.

The bottom section of Table 3 shows results for classifiers with both unigrams (binary) and patient attribute features. Using the CRF features increases recall on the general posts from 54 → 60, but decreases precision from 79 → 71. Using the patient attribute features from the VPA lexicon yields a substantial improvement. Recall improves from 54 → 79 and precision is just one point lower. Overall, the macro-averaged F score across the two categories jumps from 81% to 88%.

We performed paired bootstrap testing (Berg-Kirkpatrick et al., 2012)) to determine whether the SVM with unigrams and VPA lexicon features is statistically significantly better than the best SVM with only unigram features (binary). The SVM with unigrams and VPA lexicon features produces significantly better F scores at the $p < 0.05$ level for general post classification as well as the macro average. The F score for patient-specific classification and overall accuracy are statistically significant at the $p < 0.10$ level.

Attribute	CRF Tagger	VPA Lexicon
SPECIES/BREED	270	1045
NAME	36	43
POSSESSOR	12	233
AGE	545	1773
GENDER	153	338
WEIGHT	27	83
DISEASE/SYMPTOM	220	2673

Table 4: Number of Attributes Labeled in Test Set

Finally, we did an analysis to understand why the VPA lexicon was so much more effective than the CRF tagger when used to create features for text classification. Table 4 shows the number of words in PI sentences (identified by the classifier) of the test set that were labeled as patient attributes by the CRF tagger or the VPA lexicon. The VPA lexicon clearly labeled many more terms, and the additional coverage made a big difference for the text classifier.

5 Conclusions

This work demonstrated how annotated data can be leveraged to automatically harvest a domain-specific lexicon from a large collection of unannotated texts. Our induced VPA lexicon was then used to create patient attribute features that improved the ability of a document classifier to distinguish between patient-specific message board posts and general posts. We believe that this approach could also be used to create specialized lexicons for many other domains and applications. A key benefit of inducing lexicons from unannotated texts is that they provide additional vocabulary coverage beyond the terms found in annotated data sets, which are usually small.

6 Acknowledgements

This material is based upon work supported by the National Science Foundation under grant IIS-1018314. We are very grateful to the Veterinary Information Network for providing us with samples of their data.

References

D. Baker and A. McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*.

T. Berg-Kirkpatrick, D. Burkett, and D. Klein. 2012. An Empirical Investigation of Statistical Significance in NLP. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*.

S. Bloehdorn and A. Hotho. 2006. Boosting for text classification with semantic features. In *Advances in Web mining and Web usage Analysis*.

H. Borko and M. Bernick. 1963. Automatic Document Classification. *J. ACM*, 10(2):151–162.

I. Br, J. Szab, and A. Benczr. 2008. Latent dirichlet allocation in web spam filtering. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, R. Estevam, J. Hruschka, and T. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*.

O. Etzioni, M. Cafarella, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165(1):91–134.

J. Furnkranz, T. Mitchell, and E. Riloff. 1998. A Case Study in Using Linguistic Phrases for Text Categorization from the WWW. In *Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization*.

W. Hoyle. 1973. Automatic Indexing and Generation of Classification Systems by Algorithm. *Information Storage and Retrieval*, 9(4):233–242.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*.

S. Keerthi and D. DeCoste. 2005. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *Journal of Machine Learning Research*.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08)*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

H. Lodhi, J. Shawe-Taylor, N. Christianini, and C. Watkins. 2001. Text classification using string kernels. In *Advances in Neural Information Processing Systems (NIPS)*.

T. McIntosh and J. Curran. 2009. Reducing Semantic Drift with Bagging and Distributional Similarity. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*.

G. Miller. 1990. Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4).

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2-3):103–134, May.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

E. Riloff and W. Lehnert. 1994. Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems*, 12(3):296–333, July.

B. Roark and E. Charniak. 1998. Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.

S. Scott and M. Stan. 1998. Text classification using WordNet hypernyms. In *In Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*.

F. Sebastiani. 2002. Machine learning in automated text categorization. In *ACM computing surveys (CSUR)*.

V. Vyas and P. Pantel. 2009. Semi-automatic entity set refinement. In *Proceedings of North American Association for Computational Linguistics / Human Language Technology (NAACL/HLT-09)*.

P. Wang, J. Hu, H. Zeng, and Z. Chen. 2009. Using Wikipedia knowledge to improve text classification. In *Knowledge and Information Systems*.

S. Zelikovitz and H. Hirsh. 2001. Using LSI for text classication in the presence of background text. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)*.

Lexical Chain Based Cohesion Models for Document-Level Statistical Machine Translation

Deyi Xiong[1], Yang Ding[2], Min Zhang[1*] and Chew Lim Tan[2]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China 215006
{dyxiong, minzhang}@suda.edu.cn
[2]School of Computing, National University of Singapore, Singapore 117417
{a0082379, tancl}@comp.nus.edu.sg

Abstract

Lexical chains provide a representation of the lexical cohesion structure of a text. In this paper, we propose two lexical chain based cohesion models to incorporate lexical cohesion into document-level statistical machine translation: 1) a count cohesion model that rewards a hypothesis whenever a chain word occurs in the hypothesis, 2) and a probability cohesion model that further takes chain word translation probabilities into account. We compute lexical chains for each source document to be translated and generate target lexical chains based on the computed source chains via maximum entropy classifiers. We then use the generated target chains to provide constraints for word selection in document-level machine translation through the two proposed lexical chain based cohesion models. We verify the effectiveness of the two models using a hierarchical phrase-based translation system. Experiments on large-scale training data show that they can substantially improve translation quality in terms of BLEU and that the probability cohesion model outperforms previous models based on lexical cohesion devices.

1 Introduction

Given a source document, traditionally most statistical machine translation (SMT) systems translate the document sentence by sentence. In such a translation scheme, sentences are translated independent of any other sentences. However, a text is normally written cohesively, in which sentences are connected to each other via syntactic and lexical devices. This linguistic phenomenon is called as *textual cohesion* (Halliday and Hasan, 1976).

Cohesion is a surface-level property of well-formed texts. It deals with five categories of relationships between text units, namely co-reference, ellipsis, substitution, conjunction and lexical cohesion that is realized via semantically related words. The former four cohesion relations can be grouped as grammatical cohesion. Generally speaking, grammatical cohesion is less common and harder to identify than lexical cohesion (Barzilay and Elhadad, 1997).

As most SMT systems translate a text in a sentence-by-sentence fashion, they tend to build less lexical cohesion than human translators (Wong and Kit, 2012). We therefore study lexical cohesion for document-level translation. We use lexical chains (Morris and Hirst, 1991) to capture lexical cohesion in a text. Lexical chains are connected graphs that represent the lexical cohesion structure of a text. They have been successfully used for information retrieval (Stairmand, 1996), document summarization (Barzilay and Elhadad, 1997) and so on. In this paper, we investigate how lexical chains can be used to incorporate lexical cohesion into document-level translation.

Our basic assumption is that the lexical chains of a target document are direct correspondences of the lexical chains of its counterpart source document. This assumption is reasonable as the target document translation should be faithful to the source document in terms of both text meaning and structure. Based on this assumption, we propose a framework

*Corresponding author

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1563–1573,
Seattle, Washington, USA, 18-21 October 2013.

to incorporate lexical cohesion into target document translation via lexical chains, which works as follows.

- Compute lexical chains for each source document that is to be translated;
- Project the computed source lexical chains onto the corresponding target document by translating source chain words into target chain words using maximum entropy classifiers;
- Incorporate lexical cohesion into the target document translation via cohesion models built on the projected target lexical chains .

We build two lexical chain based cohesion models. The first model is a count model that rewards a hypothesis whenever a word in the projected target lexical chains occur in the hypothesis. As a source chain word may be translated into many different target words, we further extend the count model to a second cohesion model: a probability model that takes chain word translation probabilities into account.

We test the two lexical chain based cohesion models on a hierarchical phrase-based SMT system that is trained with large-scale Chinese-English bilingual data. Experiment results show that our lexical chain based cohesion models can achieve substantial improvements over the baseline. Furthermore, the probability cohesion model is better than the count model and it also outperforms previous cohesion models based on lexical cohesion devices (Xiong et al., 2013).

To the best of our knowledge, this is the first attempt to explore lexical chains for statistical machine translation. The remainder of this paper is organized as follows. Section 2 discusses related work and highlights the differences between our method and previous work. Section 3 briefly introduces lexical chains and algorithms that compute lexical chains. Section 4 elaborates the proposed lexical chain based framework, including details on source lexical chain computation, target lexical chain generation and the two lexical chain based cohesion models. Section 5 presents our large-scale experiments and results. Finally, we conclude with future directions in Section 6.

2 Related Work

Recent years have witnessed growing research interests in document-level statistical machine translation. Such research efforts can be roughly divided into two groups: 1) general document-level machine translation that does not explore or explores very little linguistic discourse information; 2) linguistically-motivated document-level machine translation that incorporates discourse information such as cohesion and coherence into SMT. Recent studies (Guillou, 2013; Beigman Klebanov and Flor, 2013) show that this discourse information is very important for document-level machine translation.

General Document-Level Machine Translation Tiedemann (2010) propose cache-based language and translation models for document-level machine translation. These models are built on recently translated sentences. Following this cache-based approach, Gong et al. (2011) further introduce two additional caches. They use a static cache to store bilingual phrases extracted from documents in training data that are similar to the document being translated. They also adopt a topic cache with target language topic words. Xiao et al. (2011) study the translation consistency issue in document-level machine translation. They use a hard constraint to consistently translate ambiguous source words into the most frequent translation options. Ture et al. (2012) soften this consistency constraint by integrating three counting features into decoder.

Using Lexical Cohesion Devices in Document-Level SMT Lexical cohesion devices are semantically related words, including word repetition, synonyms/near-synonyms, hyponyms and so on. They are also the cohesion-building elements in lexical chains.

Wong and Kit (2012) use lexical cohesion device based metrics to improve machine translation evaluation at the document level. These metrics measure the proportion of content words that are used as lexical cohesion devices in machine-generated translations. Hardmeier et al. (2012) propose a document-wide phrase-based decoder and integrate a semantic language model into the decoder. They argue that their semantic language model can capture lexical cohesion by exploring n-grams that cross sentence

boundaries.

Most recently Xiong et al. (2013) integrate three categories of lexical cohesion devices into document-level machine translation. They define three cohesion models based on lexical cohesion devices: a direct reward model, a conditional probability model and a mutual information trigger model. The latter two models measure the strength of lexical cohesion relation between two lexical items. They are incorporated into SMT to calculate how appropriately lexical cohesion devices are used in document translation. As lexical chains capture lexical cohesion relations among sequences of related words rather than those only between two words, experiments in Section 5 show that our lexical chain based probability cohesion model is better than the lexical cohesion device based trigger model, which is the best among the three cohesion models proposed by Xiong et al. (2013).

Modeling Coherence in Document-Level SMT In discourse analysis, cohesion is often studied together with *coherence* which is another dimension of the linguistic structure of a text (Barzilay and Elhadad, 1997). Cohesion is related to the surface structure of a text while coherence is concerned with the underlying meaning connectedness in a text (Vasconcellos, 1989). Compared with cohesion, coherence is not easy to be detected. Even so, various models have been proposed to explore coherence for document summarization and generation (Barzilay and Lapata, 2008; Louis and Nenkova, 2012). Following this line, Xiong and Zhang (2013) integrate a topic-based coherence model into document-level machine translation, where coherence is defined as a continuous sentence topic transition.

Our lexical chain based cohesion models are also related to previous work on using word and phrase sense disambiguation for lexical choice in SMT (Carpuat and Wu, 2007b; Carpuat and Wu, 2007a; Chan et al., 2007). The difference is that we use document-wide lexical chains to build our cohesion models rather than sentence-level context features. In our framework, lexical choice is performed to make the selected words consistent with the lexical cohesion structure of a document.

Carpuat (2009) explores the principle of one sense per discourse (Gale et al., 1992) in the context of SMT and imposes the constraint of one translation per discourse on document translation. We also use the one sense per discourse principle to perform word sense disambiguation on the source side in our lexical chaining algorithm (See Section 4.1).

3 Background: Lexical Chain and Chain Computation

Lexical chains are sequences of semantically related words (Morris and Hirst, 1991). They represent the lexical cohesion structure of a text. Figure 2 displays six lexical chains computed from the Chinese news article shown in Figure 1. Words in these lexical chains have lexical cohesion relations such as repetition, synonym, which may range over the entire text. For example, in the lexical chain LC^1 of Figure 2, the same word "déguó" (Germany) repeats 9 times. In the lexical chain LC^3, the two words "zǒngcái" (president) and "zhǔxí" (chairman) are synonym words. Generally, a text can have many different lexical chains, each of which represents a thread of cohesion through the text.

Several lexical chaining algorithms have been proposed to compute lexical chains from texts. Normally they need an ontology to obtain semantic relations between words. Word sense disambiguation (WSD) is also used to determine the sense of each word in a text. Generally a lexical chain computation algorithm completes the following three subtasks:

- Building a representation of a text with a set of candidate words and assigning semantic relations between the candidate words according to the ontology;
- Choosing the right sense for each candidate word via WSD;
- Building chains over the semantically related and disambiguated candidate words.

These three sub-tasks can be done separately or simultaneously.

Morris and Hirst (Morris and Hirst, 1991) define the first lexical chain computation algorithm that adopts a greedy strategy to immediately disambiguate a word at its first occurrence. This algorithm runs in linear time but suffers from inaccurate disambiguation. Barzilay and Elhadad (Barzilay and Elhadad, 1997) significantly improve WSD

déguó diànxìn gōngsī zǒngcái suǒmò cízhí
déguó diànxìn gōngsī xuānbù , qián jiānshìhuì zhǔxí qīshíèr suì de xīlèěr jiāng dānrèn gāi gōngsī de línshí zǒngcái , wéiqī liù gè yuè , zhídào suǒmò de jìrèn rénxuǎn jiērèn wéizhǐ 。
(fǎxīnshè bō áng diàn) déguó diànxìn gōngsī zǒngcái suǒmò jīntiān cíqù tā de zhíwù , tā shuō , yóuyú tā xiǎnrán bú zài shòudào déguó diànxìn gōngsī jiānshìhuì de chōngfèn xìnrèn , cízhí shì tā wéiyī de xuǎnzé 。
tóuzīrén huānyíng zhèxiàng xuānbù , déguó diànxìn gōngsī de gǔpiào yīncǐ zài fǎlánkèfú gǔpiào jiāoyì shìchǎng shàng zhǎng bǎifènzhīshíyī yǐshàng 。
suǒmò zài déguó diànxìn gōngsī bō áng zǒngbù zhàokāi jiānshìhuì tèbié huìyì zhōng fābiǎo yī xiàng shēngmíng , tā shuō : 「 wǒ yǐ yàoqiú jiānshìhuì jiěchú wǒ de zhíwù 。 」
yóuyú liǎng gè yuè hòu jiāng jǔxíng dàxuǎn , dàn liánhézhèngfǔ zài mín diào zhōng shēngwàng luòhòu , déguó zǒnglǐ shī ruòdé sìhū xīwàng zài gǔjià xiàcuò zhì xīn dī shí , déguó diànxìn gōngsī shùbǎiwàn míng xiǎo gǔdōng de zījīn bìng wèi xiāoshī , ér zhīchí tā 。
déguó diànxìn gǔjià hòuláihuí wěn , yǐ shíyīdiǎnyībā ōuyuán zuò shōu , shàngzhǎng bǎifènzhībādiǎnwǔsì 。
déguó cáizhèngbù huānyíng suǒmò cízhí de juédìng 。

Figure 1: An example of a Chinese news article (written in pinyin).

LC^1: {déguó, déguó, bō, déguó, déguó, déguó, déguó, bō, déguó, déguó, déguó}
LC^2:{jiānshìhuì, fǎxīnshè, jiānshìhuì, zǒngbù, jiānshìhuì, jiānshìhuì}
LC^3: {zǒngcái, zhǔxí, zǒngcái, zǒnglǐ}
LC^4: {cízhí, cíqù, cízhí, cízhí}
LC^5: {zhǎng, xiàcuò, shàngzhǎng}
LC^6: {xuānbù, xuānbù, fābiǎo}

Figure 2: Six lexical chains from the example in Figure 1.

accuracy by processing all possible combinations of word senses in a text to disambiguate words. Unfortunately, their algorithm runs slowly in quadratic time. Galley and Mckeown (2003) present an algorithm that are better than the former two algorithms both in terms of running efficiency and WSD accuracy. They separate the WSD sub-task from the task of lexical chain building and impose a "one sense per discourse" constraint in the WSD step.

4 Translating Documents Using Lexical Chains

In this section, we describe how we incorporate lexical cohesion into document-level machine translation using lexical chains. We divide the lexical chain based document-level machine translation process into three steps: (1) computing lexical chains for source documents with a source language ontology, (2) generating target lexical chains from the computed source lexical chains, and finally (3) incorporating lexical cohesion encoded in the generated target lexical chains into document-level translation via lexical chain based cohesion models. The remainder of this section will elaborate these three steps.

4.1 Source Lexical Chains Computation

We follow the chain computation algorithm introduced by Galley and McKeown (2003) to build lexical chains on source (Chinese) documents. In the algorithm, the chaining process includes three steps: choosing candidate words to build a disambiguation graph (Galley and McKeown, 2003) for each document, disambiguating the candidate words and finally building lexical chains over the disambiguated candidate words.

The disambiguation graph can be considered as a representation of all possible interpretations of its corresponding text. In the graph, nodes are candidate words with different senses and edges between word senses are weighted according to their semantic relations, such as synonym, hypernym and so on. We use an extended version of a Chinese thesaurus *Tongyici Cilin* (Cilin for short) to define word senses and semantic relations between senses. The ex-

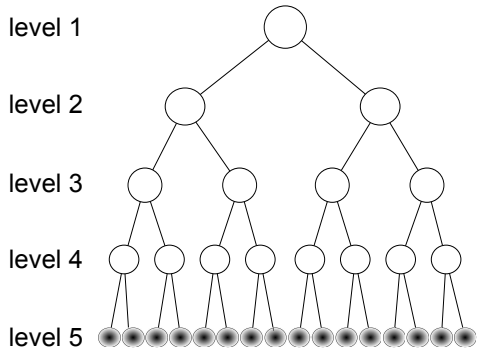


Figure 3: The architecture of the extended Cilin. For simplicity, we only draw a binary tree to represent the hierarchical structure of Cilin. This doesn't mean that each semantic class at level i has only two sub-classes at level $i+1$. Actually, they have multiple sub-classes.

tended Cilin contains 77,343 Chinese words, which are organized in a hierarchical structure containing 5 levels as shown in Figure 3. In the 5th level, each node represents an atomic concept which consists of a set of synonyms. These atomic concepts are just like *synsets* in WordNet. We use them to represent senses of words in the disambiguation graph.

We select nouns, verbs, abbreviations and idioms as candidate words for the disambiguation graph. These words are identified by a Chinese part-to-speech tagger LTP (Che et al., 2010) in a preprocessing step. In order to build the disambiguation graph, we first build an array indexed by the atomic concepts of Cilin, then insert a copy of each candidate word into its all concept (sense) entries in the array. After that, we create all semantic links among senses of different candidate words in the disambiguation graph following Galley and McKeown (2003).

In the second step, we use the principle of one sense per discourse to perform WSD for each candidate word in the disambiguation graph. We sum the weights of all semantic links under the different senses of the candidate word in question. The sense with the highest sum of weights is considered as the most probable sense for this word. We then assign this sense to all occurrences of the word in the document by adopting the constraint of one sense per discourse.

Once all candidate words are disambiguated, we can build lexical chains over these words by removing all semantic links that connect those unselected word senses. The six lexical chains shown in Figure 2 are computed from the Chinese document in Figure 1 exactly following the algorithm of Galley and McKeown (2003). The only difference is that we use Cilin rather than WordNet as the ontology.

4.2 Target Lexical Chains Generation

Since a faithful target document translation should follow the same cohesion structure as that in its corresponding source document, we generate target lexical chains from the computed source lexical chains. Given a source lexical chain $LC_s = \{s_i^j\}$ where the ith chain word s_i^j is from the jth sentence of the source document D_s, we generate a target lexical chain $LC_t = \{t_i^j\}$ using maximum entropy (MaxEnt) classifiers. Particularly, we translate a word s_i^j in the source lexical chain into a target word t_i^j in the target lexical chain using a corresponding MaxEnt classifier as follows[1].

$$P(t_i^j|\mathcal{C}(s_i^j)) = \frac{exp(\sum_k \theta_k f_k(t_i^j, \mathcal{C}(s_i^j)))}{\sum_t exp(\sum_k \theta_k f_k(t, \mathcal{C}(s_i^j)))} \quad (1)$$

where f_k are binary features, θ_k are weights of these features, and $\mathcal{C}(s_i^j)$ is the surrounding context of chain word s_i^j.

We train one MaxEnt classifier per unique source chain word. For each classifier, we define two groups of binary features: 1) the preceding and succeeding two words of s_i^j in the jth sentence ($\{w_{-2}, w_{-1}, s_i^j, w_{+1}, w_{+2}\}$); 2) the preceding and succeeding one word of s_i^j in the lexical chain LC_s ($\{s_{i-1}^p, s_i^j, s_{i+1}^q\}$). All features are in the following binary form.

$$f(t_i^j, \mathcal{C}(s_i^j)) = \begin{cases} 1, & \text{if } t_i^j = \clubsuit \text{ and } \mathcal{C}(s_i^j).\heartsuit = \spadesuit \\ 0, & \text{else} \end{cases} \quad (2)$$

where the symbol $\clubsuit$ is a placeholder for a possible target word, the symbol $\heartsuit$ indicates a contextual element for the chain word s_i^j (e.g., the preceding word in the jth sentence or the succeeding word in the lexical chain LC_s), and the symbol $\spadesuit$ represents the value of $\heartsuit$.

Given a source document D_s and its N lexical chains $\{LC_s^k\}_{k=1}^N$ computed from the document as

[1]We collect training instances from word-aligned bilingual data to train the MaxEnt classifier.

described in Section 4.1, we can generate the N target lexical chains $\{LC_t^k\}_{k=1}^N$ using our MaxEnt classifiers. Each target word t_i^j in the target lexical chain LC_t^k is the translation of its corresponding source word s_i^j in the source lexical chain LC_s^k with the highest probability $P(t_i^j|\mathcal{C}(s_i^j))$ according to Eq. (1).

As we know, the MaxEnt classifier can generate multiple translations for each source word. In order to incorporate these multiple chain word translations, we can generate a *super target lexical chain* ${}_\epsilon LC_t$ from a source lexical chain LC_s, where ϵ is a pre-defined threshold used to select multiple translations. For example, given a source lexical chain $LC_s = \{a, b, c\}$, we can have the corresponding super target lexical chain ${}_\epsilon LC_t = \{\{a_t^1, a_t^2...\}, \{b_t^1, b_t^2...\}, \{c_t^1, c_t^2...\}\}$, where x_t^i is the translation of x with a translation probability $P(x_t^i|\mathcal{C}(x)) \geq \epsilon$ according to Eq. (1). Integrating multiple translations for each source chain word, we can reduce the error propagation of the MaxEnt classifier to some extent. Our experiments also confirm that the super target lexical chains with multiple translation options for each chain word are better than the target lexical chains with only one translation per chain word. Therefore we build our cohesion models based on the super target lexical chains, which will be described in the next section.

4.3 Lexical Chain Based Cohesion Models

Once we generate the super target lexical chains $\{{}_\epsilon LC_t^k\}_{k=1}^N$ for the target document D_t, we can use them to provide constraints for the target document translation. Our key interest is to make the target document translation T_{D_t} as cohesive as possible. We therefore propose lexical chain based cohesion models to measure the cohesion of the target document translation. The basic idea is to reward a translation hypothesis if a word from the super target lexical chains occurs in the hypothesis. According to the difference in the reward strategy, we have two cohesion models: a *count cohesion model* and a *probability cohesion model*.

Count Cohesion Model $M_c(T_{D_t}, \{{}_\epsilon LC_t^k\}_{k=1}^N)$: This model rewards a translation hypothesis of the jth sentence in the document whenever a lexical chain word t_i^j occurs in the hypothesis. The model maintains a counter and accumulates the counter when necessary. It is factorized into the sentence cohesion metric $M_c(T_j, \{{}_\epsilon LC_t^k\}_{k=1}^N)$, where T_j is the translation of the jth sentence in the target document. $M_c(T_j, \{{}_\epsilon LC_t^k\}_{k=1}^N)$ is formulated as follows.

$$M_c(T_j, \{{}_\epsilon LC_t^k\}_{k=1}^N) = \prod_{w \in T_j} \prod_{t_i^j \in \mathbf{C}} e^{\delta(w, t_i^j)} \quad (3)$$

where $\mathbf{C}$ represents $\{{}_\epsilon LC_t^k\}_{k=1}^N$, and the δ function is defined as follows.

$$\delta(w, t_i^j) = \begin{cases} 1, & \text{if } t_i^j = w \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Probability Cohesion Model $M_p(T_{D_t}, \{{}_\epsilon LC_t^k\}_{k=1}^N)$: This model rewards a translation hypothesis according to the translation probability of a chain word that occurs in the hypothesis. The translation probability is computed by Eq. (1). The model is also factorized into the sentence cohesion metric $M_p(T_j, \{{}_\epsilon LC_t^k\}_{k=1}^N)$ which is formulated as follows.

$$M_p(T_j, \{{}_\epsilon LC_t^k\}_{k=1}^M) = \prod_{w \in T_j} \prod_{t_i^j \in \mathbf{C}} e^{\delta(w, t_i^j)} \times P(t_i^j|\mathcal{C}(s_i^j)) \quad (5)$$

where $P(t_i^j|\mathcal{C}(s_i^j)$ is the translation probability computed according to Eq. (1).

4.4 Decoding

The proposed lexical chain based cohesion models are integrated into the log-linear translation framework of SMT as a cohesion feature. Before translating a source document, we compute lexical chains for the source document as described in Section 4.1. We then generate the super target lexical chains. In order to efficiently calculate our lexical chain based cohesion models, we reorganize words in the super target lexical chains into vectors. We associate each source sentence S_j a vector to store target lexical chain words that are to occur in the corresponding target sentence T_j.

Although we still translate a source document sentence by sentence, we capture the global cohesion structure of the document via lexical chains and use the lexical chain based cohesion models to constrain word selection in document translation. Figure 4 shows the architecture of an SMT system with the lexical chain based cohesion model.

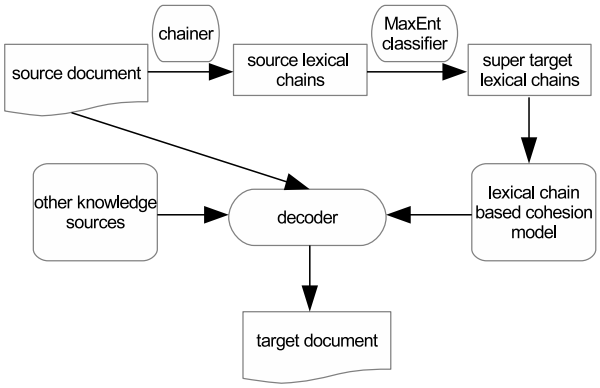


Figure 4: Architecture of an SMT system with the lexical chain based cohesion model.

5 Experiments

In this section, we conducted a series of experiments to validate the effectiveness of the proposed lexical chain based cohesion models for Chinese-to-English document-level machine translation. We used a hierarchical phrased-based SMT system (Chiang, 2007) trained on large-scale data. In particular, we aim at:

- Measuring the impact of the threshold ϵ on the probability cohesion model and selecting the best threshold on a development test set.
- Investigating the effect of the two lexical-chain based cohesion models.
- Comparing our lexical chain based cohesion models against the previous lexical cohesion device based models (Xiong et al., 2013).

5.1 Setup

We collected our bilingual training data from LDC, which includes the corpus LDC2002E18, LDC2003E07, LDC2003E14, LDC2004E12, LDC2004T07, LDC2004T08 (Only Hong Kong News), LDC2005T06 and LDC2005T10. The collected bilingual training data contains 3.8M sentence pairs with 96.9M Chinese words and 109.5M English words. We trained a 4-gram language model on the Xinhua portion of the English Gigaword corpus (306 million words) via the SRILM toolkit (Stolcke, 2002) with Kneser-Ney smoothing.

	Training	**MT05**	**MT06**	**MT08**
#Doc	103,236	100	79	109
#Sent	2.80M	1,082	1,664	1,357
#Chain	3.52M	1700	2172	1693
#AvgC	35.72	17	27.49	15.53
#AvgW	14.81	5.89	6.89	5.63

Table 1: Statistics of the training, development and test sets, which show the number of documents (#Doc) and sentences (#Sent), the number of lexical chains extracted from the source documents (#Chain), the average number of lexical chains per document (#AvgC) and the average number of words per lexical chain (#AvgW).

In order to build the lexical chain based cohesion models, we selected corpora with document boundaries explicitly provided from the bilingual training data together with the whole Hong Kong parallel text corpus as the cohesion model training data[2]. We show the statistics of these selected corpora in Table 1. They contain 103,236 documents and 2.80M sentences. Averagely, each document consists of 28.4 sentences. From the source documents of the selected corpora, we extract 3.52M lexical chains. On average, there are 35.72 lexical chains per document and 14.81 words per lexical chain.

We used the off-the-shelf MaxEnt toolkit[3] to train one MaxEnt classifier per unique source lexical chain word (61,121 different source chain words in total). We performed 100 iterations of the L-BFGS algorithm implemented in the training toolkit for each chain word with both Gaussian prior and event cutoff set to 1 to avoid overfitting. After event cutoff, we have an average of 17.75 different classes (target translations) per source chain word.

We used the NIST MT05 as the tuning set for the minimum error rate training (MERT) [Och, 2003], the NIST MT06 as the development test set and the MT08 as the final test set. The numbers of documents/sentences in the NIST MT05, MT06 and MT08 are 100/1082, 79/1664 and 109/1357 respectively. They contain 17, 27.49 and 15.53 lexical chains per document respectively.

We used the case-insensitive BLEU-4 (Papineni

[2]The training data includes LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2004T08 (Hong Kong Hansards/Laws/News).

[3]Available at: http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

ϵ	**MT06**
0.05	30.53
0.1	**31.64**
0.2	31.45
0.3	30.73
0.4	31.01

Table 2: BLEU scores of the probability cohesion model $M_p(T_{D_t}, \{_\epsilon LC_t^k\}_{k=1}^N)$ with different values for the threshold ϵ.

System	**MT06**	**MT08**	**Avg**
Baseline	30.43	23.32	26.88
LexChainCount(top 1)	30.46	23.52	26.99
LexChainCount	30.79	23.34	27.07
LexChainProb	**31.64**	**24.54**	**28.09**

Table 3: Effects of the lexical chain based count and probability cohesion models. LexChainCount: the count model defined in Eq. (3). LexChainProb: the probability model defined in Eq. (5).

et al., 2002) as our evaluation metric. As MERT is normally instable, we ran the tuning process three times for all our experiments and presented the average BLEU scores on the three MERT runs as suggested by Clark et al (2011).

5.2 Setting the Threshold ϵ

As the two lexical chain based cohesion models are built on the super target lexical chains that are associated with a parameter ϵ, we need to tune the threshold parameter ϵ on the development test set NIST MT06. We conducted a group of experiments using the probability cohesion model defined in Eq. (5) to find the best threshold. Experiment results are shown in Table 2.

If we set the threshold too small (e.g., 0.05), the super target lexical chains may contain too many noisy words that are not the translations of source lexical chain words, which may jeopardise the quality of the super target lexical chains. The cohesion model built on these noisy super target lexical chains may select incorrect words rather than the proper lexical chain words. On the other hand, if we set the threshold too large (e.g., 0.3 or 0.4), we may take the risk of not selecting the appropriate chain word translations into the super target lexical chains. It seems that the best threshold is 0.1 as we obtained the highest BLEU score 31.64 on the NIST MT06 with this threshold. Therefore we set the threshold ϵ to 0.1 in all experiments thereafter.

5.3 Effect of the Count and Probability Cohesion Model

After we found the best threshold, we carried out experiments to test the effect of the two lexical chain based cohesion models: the count and probability cohesion model. We compared them against the baseline system that does not integrate any lexical chain information. We also compared the count cohesion model (LexChainCount(top1)) built on the target lexical chains where each target chain word is the best translation of its corresponding source lexical chain word according to Eq. (1). Experiment results are shown in Table 3.

From Table 3, we can observe that

- Our lexical chain based cohesion models are able to substantially improve the translation quality in terms of BLEU score. We achieve an average improvement of up to 1.21 BLEU points over the baseline on the two test sets MT06 and MT08.
- The count cohesion model built on the super target lexical chains is better than that based on the target lexical chains only with top one translations (27.07 vs. 26.99). This shows the advantage of the super target lexical chains $\{_\epsilon LC_t^k\}_{k=1}^N$ over the standard target lexical chians $\{LC_t^k\}_{k=1}^N$.
- Finally, the probability cohesion model is much better than the count cohesion model (28.09 vs. 27.07). This suggests that we should take into account chain word translation probabilities when we reward hypotheses where target lexical chain words occur.

5.4 Lexical Chains vs. Lexical Cohesion Devices

As we have mentioned in Section 2, lexical cohesion devices can be also used to build lexical cohesion models to capture lexical cohesion relations in a text. We therefore want to compare our lexical chain based cohesion models with the lexical cohesion device based cohesion models.

System	MT06	MT08	Avg
Baseline	30.43	23.32	26.88
LexDeviceTrigger	31.35	24.11	27.73
LexChainProb	31.64	24.54	28.09

Table 4: The lexical chain based probability cohesion model (LexChainProb) vs. the lexical cohesion device based trigger model (LexDeviceTrigger).

We re-implemented the mutual information trigger model that is the best lexical cohesion model based on lexical cohesion devices among the three models proposed by Xiong et al. (2013). The mutual information trigger model measures the association strength of two lexical cohesion items x and y in a lexical cohesion relation xRy. In the model, it is required that x occurs in a sentence preceding the sentence where y occurs and that the two items have a lexical cohesion relation such as word repetition, synonym. The model treats x as the trigger and y as the triggered item. The mutual information between the trigger x and the triggered item y estimates how possible y will occur given x is mentioned in a text.

The comparison results are reported in Table 4. Our lexical chain based probability cohesion model outperforms the lexical cohesion device based trigger model by 0.36 BLEU points. The reason for this superiority of our cohesion model over the trigger model may be that the former model captures lexical cohesion relations among sequences of words through lexical chains while the latter model captures lexical cohesion relations only between two related words.

6 Conclusions

We have presented two lexical chain based cohesion models that incorporate the lexical cohesion structure of a text into document-level machine translation. We project the lexical chains of a source document to the corresponding target document by translating each word in each source lexical chain into their counterparts via MaxEnt classifiers. The projected target lexical chains provide a representation of the lexical cohesion structure of the target document that is to be generated. We build two cohesion models based on the projected target lexical chains: a count model that rewards a hypothesis according to the time of occurrence of target lexical chain words in the hypothesis and a probability model that further takes translation probabilities into account when rewarding hypotheses. These two cohesion models are used to constrain word selection for document translation so that the generated document is consistent with the projected lexical cohesion structure.

We have integrated the two proposed cohesion models into a hierarchical phrase-based SMT system. Experiment results on large-scale data validate that

- The lexical chain based cohesion models are able to substantially improve translation quality in terms of BLEU.
- The probability cohesion model is better than the count cohesion model.
- The lexical chain based probability cohesion model is better than the previous mutual information trigger model that adopts lexical cohesion devices to capture lexical cohesion relations between two related words.

As we mentioned in Section 2, cohesion is closely connected to coherence. It provides a surface indicator for coherence identification (Barzilay and Elhadad, 1997). In the future, we would like to use lexical chains to identify coherence and incorporate both cohesion and coherence into document-level machine translation.

References

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. In *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Beata Beigman Klebanov and Michael Flor. 2013. Associative texture is lost in translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 27–32, Sofia, Bulgaria, August. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2007a. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. In *Proceedings of the 11th Conference on Theoretical and*

Methodological Issues in Machine Translation, pages 43–52.

Marine Carpuat and Dekai Wu. 2007b. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, Prague, Czech Republic, June. Association for Computational Linguistics.

Marine Carpuat. 2009. One translation per discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 19–27, Boulder, Colorado, June. Association for Computational Linguistics.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June. Association for Computational Linguistics.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: a chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, COLING '10, pages 13–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, Harriman, NY, February.

Michel Galley and Kathleen McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 1486–1488, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 909–919, Edinburgh, Scotland, UK., July.

Liane Guillou. 2013. Analysing lexical consistency in translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 10–18, Sofia, Bulgaria, August. Association for Computational Linguistics.

M.A.K Halliday and Ruqayia Hasan. 1976. *Cohesion in English*. London: Longman.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1179–1190, Jeju Island, Korea, July.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1157–1168, Jeju Island, Korea, July. Association for Computational Linguistics.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Comput. Linguist.*, 17(1):21–48, March.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

M.A. Stairmand. 1996. *A computational analysis of lexical cohesion with applications in information retrieval*. UMIST.

Andreas Stolcke. 2002. Srilm–an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado, USA, September.

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15, Uppsala, Sweden, July.

Ferhan Ture, Douglas W. Oard, and Philip Resnik. 2012. Encouraging consistent translation choices. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 417–426, Montréal, Canada, June.

Muriel Vasconcellos. 1989. Cohesion and coherence in the presentation of machine translation products. In James E.Alatis, editor, *Geogetown University Round Table on Languages and Linguistics 1989*, pages 89–105, Washington, D.C. Georgetown University Press.

Billy T. M. Wong and Chunyu Kit. 2012. Extending machine translation evaluation metrics with lexical cohesion to document level. In *Proceedings of the*

2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1060–1068, Jeju Island, Korea, July.

Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. 2011. Document-level consistency verification in machine translation. In *Proceedings of the 2011 MT summit XIII*, pages 131–138, Xiamen, China, September.

Deyi Xiong and Min Zhang. 2013. A topic-based coherence model for statistical machine translation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13)*, Bellevue, Washington, USA, July.

Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lv, and Qun Liu. 2013. Modeling lexical cohesion for document-level machine translation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-13)*, Beijing, China, August.

A Convex Alternative to IBM Model 2

Andrei Simion
Columbia University
IEOR Department
New York, NY, 10027
aas2148@columbia.edu

Michael Collins
Columbia University
Computer Science
New York, NY, 10027
mc3354@columbia.edu

Clifford Stein
Columbia University
IEOR Department
New York, NY, 10027
cs2035@columbia.edu

Abstract

The IBM translation models have been hugely influential in statistical machine translation; they are the basis of the alignment models used in modern translation systems. Excluding IBM Model 1, the IBM translation models, and practically all variants proposed in the literature, have relied on the optimization of likelihood functions or similar functions that are non-convex, and hence have multiple local optima. In this paper we introduce a convex relaxation of IBM Model 2, and describe an optimization algorithm for the relaxation based on a subgradient method combined with exponentiated-gradient updates. Our approach gives the same level of alignment accuracy as IBM Model 2.

1 Introduction

The IBM translation models (Brown et al., 1993) have been tremendously important in statistical machine translation (SMT). The IBM models were the first generation of SMT systems; in recent work, they play a central role in deriving alignments used within many modern SMT approaches, for example phrase-based translation models (Koehn, 2008) and syntax-based translation systems (e.g., (Chiang, 2005; Marcu et al., 2006)). Since the original IBM paper, there has been a large amount of research exploring the original IBM models and modern variants (e.g., (Moore, 2004; Liang et al., 2006; Toutanova and Galley, 2011; Riley and Gildea, 2012; Vogel et al., 1996)).

Excluding IBM Model 1, the IBM translation models, and practically all variants proposed in the literature, have relied on the optimization of likelihood functions or similar functions that are non-convex. Unfortunately, non-convex objective functions have multiple local optima, and finding a global optimum of a non-convex function is typically a computationally intractible problem. Typically, an EM algorithm is used, which often runs in a reasonable amount of time, but with no guarantees of finding a global optima (or for that matter, even a near-optimal solution).

In this paper we make the following contributions:

- We introduce a convex relaxation of IBM Model 2. At a very high level, the relaxation is derived by replacing the product $t(f_j|e_i) \times d(i|j)$ with a relaxation that is commonly used in the linear programming literature (e.g., see (Bertsimas, 1997; Bertsimas and Tsitsiklis, 1997; Martins et al., 2010)). (Here $t(f|e)$ are the translation parameters of the model, and $d(i|j)$ are the distortion parameters; the product is non-linear, effectively introducing non-convexity into the problem.)
- We describe an optimization algorithm for the relaxed objective, based on a combination of stochastic subgradient methods with the exponentiated-gradient (EG) algorithm (Kivinen and Warmuth, 1997; Beck and Teboulle, 2003).
- We describe experiments with the method on standard alignment datasets, showing that the EG algorithm converges in only a few passes over the data, and that our method achieves accuracies that are very similar to those of IBM Model 2.

Framing the unsupervised learning of alignment models as a convex optimization problem, with guaranteed convergence to a global optimum, has several clear advantages. First, the method is easier to analyze, as the objective function is being truly maximized. Second, there is no need for initialization heuristics with the approach, given that the method will always converge to a global optimum. Finally, we expect that our convexity-based approach may facilitate the further development of more convex models. There has been a rich

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1574–1583,
Seattle, Washington, USA, 18-21 October 2013.

interplay between convex and non-convex methods in machine learning: as one example consider the literature on classification problems, with early work on the perceptron (linear/convex), then work on neural networks with back-propagation (non-linear/non-convex), then the introduction of support vector machines (non-linear/convex), and finally recent work on deep belief networks (non-linear/non-convex). In view of these developments, the lack of convex methods in translation alignment models has been noticeable, and we hope that our work will open up new directions and lead to further progress in this area.

Notation. Throughout this paper, for any integer N, we use $[N]$ to denote $\{1 \dots N\}$ and $[N]_0$ to denote $\{0 \dots N\}$.

2 Related Work

(Brown et al., 1993) introduced IBM Models 1 through 5, and optimization methods for these models based on the EM algorithm. While the models were originally introduced for full translation, they are now mainly used to derive alignments which are then used by phrase-based and other modern SMT systems. Since the original IBM models were introduced, many variants have been introduced in the literature. (Vogel et al., 1996) introduced a model, sometimes referred to as IBM 2.5, which uses a parameterization that is similar to a hidden Markov model, and which allows the value of each alignment variable to be conditioned on a previous alignment variable. (Liang et al., 2006) describe a method that explicitly incorporates agreement preferences during training. (Och and Ney, 2003) give a systematic comparison of several alignment models in the literature. (Moore, 2004) gives a detailed study of IBM Model 1, showing various steps that can be used to improve its performance. (Ganchev et al., 2010) describes a method based on posterior regularization that incorporates additional constraints within the EM algorithm for estimation of IBM models. All of these approaches are unsupervised, in that they do not require labeled alignment data; however several authors have considered supervised models (e.g., see (Lacoste-Julien et al., 2006; Taskar et al., 2005; Haghighi et al., 2009)). The focus of the current paper is on unsupervised learning; the unsupervised variants described above all make use of non-convex objective functions during training, with the usual problems with multiple local maxima.

3 The IBM Model 1 and Model 2 Optimization Problems

In this section we give a brief review of IBM Models 1 and 2, and the optimization problems arising from these models. The standard approach for optimization within these models is the EM algorithm.

Throughout this section, and the remainder of the paper, we assume that our set of training examples is $(e^{(k)}, f^{(k)})$ for $k = 1 \dots n$, where $e^{(k)}$ is the k'th English sentence and $f^{(k)}$ is the k'th French sentence. Following standard convention, we assume the task is to translate from *French* (the "source" language) into *English* (the "target" language). We use E to denote the English vocabulary (set of possible English words), and F to denote the French vocabulary. The k'th English sentence is a sequence of words $e_1^{(k)} \dots e_{l_k}^{(k)}$ where l_k is the length of the k'th English sentence, and each $e_i^{(k)} \in E$; similarly the k'th French sentence is a sequence $f_1^{(k)} \dots f_{m_k}^{(k)}$ where each $f_j^{(k)} \in F$. We define $e_0^{(k)}$ for $k = 1 \dots n$ to be a special NULL word (note that E contains the NULL word). Finally, we define $L = \max_{k=1}^{n} l_k$ and $M = \max_{k=1}^{n} m_k$.

For each English word $e \in E$, we will assume that $D(e)$ is a *dictionary* specifying the set of possible French words that can be translations of e. The set $D(e)$ is a subset of F. In practice, $D(e)$ can be derived in various ways; in our experiments we simply define $D(e)$ to include all French words f such that e and f are seen in a translation pair.

Given these definitions, the IBM model 2 optimization problem is given in Figure 1. The parameters in this problem are $t(f|e)$ and $d(i|j)$. The $t(f|e)$ parameters are *translation* parameters specifying the probability of English word e being translated as French word f. The *distortion* parameters $d(i|j)$ specify the probability of the j'th French word in a sentence being aligned to the i'th English word. We use a variant of IBM Model 2 where the distortion variables are shared across all sentence lengths (similar variants have been used in (Liang et al., 2006) and (Koehn, 2008)). The objective function is then

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3.

Parameters:
• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
• A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \quad (1)$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \quad (2)$$

$$\forall i \in [L]_0, j \in [M], \quad d(i|j) \geq 0 \quad (3)$$

$$\forall j \in [M], \quad \sum_{i \in [L]_0} d(i|j) = 1 \quad (4)$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) d(i|j) \quad (5)$$

with respect to the $t(f|e)$ and $d(i|j)$ parameters.

Figure 1: The IBM Model 2 Optimization Problem.

the log-likelihood of the training data (see Eq. 5):

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log p(f_j^{(k)}|e^{(k)}) \ ,$$

where

$$p(f_j^{(k)}|e^{(k)}) = \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) d(i|j) \ .$$

Crucially, while the constraints in the IBM Model 2 optimization problem are linear, the objective function in Eq. 5 is non-convex. Therefore, optimization methods for IBM Model 2, in particular the EM algorithm, are typically only guaranteed to reach a local maximum of the objective function.

For completeness, Figure 2 shows the optimization problem for IBM Model 1. In IBM Model 1 the distortion parameters $d(i|j)$ are all fixed to be the uniform distribution (i.e., $1/(L+1)$). The objective function for IBM Model 1 is actually convex, so the EM algorithm will converge to a global maximum. However IBM Model 1 is much weaker than model 2, and typically gives far worse performance.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3.

Parameters:
• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \quad (6)$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \quad (7)$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)} \quad (8)$$

with respect to the $t(f|e)$ parameters.

Figure 2: The IBM Model 1 Optimization Problem.

A common heuristic is to initialize the $t(f|e)$ parameters in EM optimization of IBM Model 2 using the output from IBM Model 1. The intuition behind this heuristic is that the IBM Model 1 values for $t(f|e)$ will be a reasonable starting point, and the EM algorithm will climb to a "good" local optimum. We are not aware of any guarantees for this initialization heuristic, however.

4 A Convex Relaxation of IBM Model 2

We now introduce a convex optimization problem, the I2CR (IBM 2 Convex Relaxation) problem. As its name suggests, this optimization problem is closely related to IBM Model 2, but is convex. Because of this it will be relatively easy to derive an optimization algorithm that is guaranteed to converge to a global optimum. Our experiments show that the relaxation gives very similar performance to the original IBM 2 optimization problem, as described in the previous section.

We first describe an optimization problem, I2CR-1, that illustrates the basic idea behind the convex relaxation. We then describe a refined relaxation, I2CR-2, that introduces a couple of modifications, and which performs well in experiments.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.
- A parameter $q(i, j, k)$ for each $k \in [n]$, $i \in [l_k]_0$, $j \in [m_k]$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \tag{9}$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \tag{10}$$

$$\forall i \in [L]_0, j \in [M], \quad d(i|j) \geq 0 \tag{11}$$

$$\forall j \in [M], \quad \sum_{i \in [L]_0} d(i|j) = 1 \tag{12}$$

$$\forall i, j, k, \quad q(i, j, k) \geq 0 \tag{13}$$

$$\forall i, j, k, \quad q(i, j, k) \leq d(i|j) \tag{14}$$

$$\forall i, j, k, \quad q(i, j, k) \leq t(f_j^{(k)}|e_i^{(k)}) \tag{15}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} q(i, j, k) \tag{16}$$

with respect to the $q(i, j, k)$, $t(f|e)$ and $d(i|j)$ parameters.

Figure 3: The I2CR-1 (IBM 2 Convex Relaxation) Problem, version 1.

4.1 The I2CR-1 Problem

The I2CR-1 problem is shown in Figure 3. A first key idea is to introduce a new variable $q(i, j, k)$ for each $k \in [n]$, $i \in [l_k]_0$, $j \in [m_k]$: that is, a new variable for each triple (i, j, k) specifying a sentence pair, and a specific English and French position in that sentence. Each q variable must satisfy the constraints in Eqs. 13-15, repeated here for convenience:

$$\begin{aligned} &\forall i, j, k, \quad q(i, j, k) \geq 0 \, , \\ &\forall i, j, k, \quad q(i, j, k) \leq d(i|j) \, , \\ &\forall i, j, k, \quad q(i, j, k) \leq t(f_j^{(k)}|e_i^{(k)}) \, . \end{aligned}$$

The objective function is

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} q(i, j, k)$$

which is similar to the objective function in Figure 1, but where $t(f_j^{(k)}|e_i^{(k)}) \times d(i|j)$ has been replaced by $q(i, j, k)$. The intuition behind the new problem is as follows. If, instead of the constraints in Eqs. 13-15, we had the constraint

$$q(i, j, k) = t(f_j^{(k)}|e_i^{(k)}) \times d(i|j) \, , \tag{17}$$

then the I2CR-1 problem would clearly be identical to the IBM Model 2 optimization problem. We have used a standard relaxation of the non-linear constraint $x = y \times z$ where x, y, z are all variables in the range $[0, 1]$, namely

$$\begin{aligned} x &\leq y \, , \\ x &\leq z \, , \\ x &\geq y + z - 1 \, . \end{aligned}$$

These inequalites are a relaxation in the sense that any (x, y, z) triple that satisfies $x = y \times z$ also satisfies these constraints. Applying this relaxation to Eq. 17 gives

$$\begin{aligned} q(i, j, k) &\leq t(f_j^{(k)}|e_i^{(k)}) \, , \\ q(i, j, k) &\leq d(i|j) \, , \\ q(i, j, k) &\geq t(f_j^{(k)}|e_i^{(k)}) + d(i|j) - 1 \, . \end{aligned} \tag{18}$$

The final thing to note is that the constraint in Eq. 18 can be omitted in the I2CR-1 problem. This is because the task is to maximize the objective with respect to the q variables and the objective is strictly increasing as the q values increase—thus lower bounds on their values are redundant in the I2CR-1 problem.

It is easily verified that the constraints in the I2CR-1 problem are linear, and that the objective function is convex. In Section 5 of this paper we describe an optimization method for the problem.

Note that because the objective function is being maximized, and the objective increases monotonically as the q values increase, at the global optimum[1]

[1] More precisely, at *any* global optimum: the objective function may not be strictly convex, in which case there will be multiple global optima.

Input: Same as in I2CR-1 (Figure 4).
Parameters: Same as in I2CR-1 (Figure 4).

Constraints: Same as in I2CR-1 (Figure 4).
Objective: Maximize

$$\frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}q(i,j,k) + \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}\frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)}$$

with respect to the $q(i,j,k)$, $t(f|e)$ and $d(i|j)$ parameters.

Figure 4: The I2CR-2 (IBM 2 Convex Relaxation) Problem, version 2. The problem is identical to the I2CR-1 problem, but it also includes a term in the objective function that is identical to the IBM Model 1 objective. We define $\log'(z) = \log(z+\lambda)$ where λ is a small positive constant.

we have

$$q(i,j,k) = \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\}\ ,$$

where $\min\{x,y\}$ returns the minimum of the two values x and y. Thus, we could actually eliminate the q variables and write an optimization problem that is identical to the IBM Model 2 optimization problem, but with the objective function

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k}\min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\}\ .$$

It will turn out that both views of the I2CR-1 problem—with and without the q variables—are helpful, so we have included both in this paper.

4.2 The I2CR-2 Problem

Figure 4 shows the refined optimization problem, which we call I2CR-2. The problem incorporates two modifications. First, we modify the objective function to be

$$\frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}q(i,j,k) + \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}\frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)}\ .$$

Thus the objective function includes a second term that is identical to the objective function for IBM Model 1 (see Figure 2). In preliminary experiments with the I2CR-1 optimization problem, we found that the I2CR-1 objective was not sufficiently dependent on the t parameters: intuitively, if the d parameters achieve the $\min$ on many training examples, the values for the t variables become unimportant. The addition of the IBM Model 1 objective fixed this problem by introducing a term that depends on the t values alone.

Second, we replace $\log$ by $\log'$, where $\log'(z) = \log(z+\lambda)$, and λ is a small positive constant (in our experiments we used $\lambda = 0.001$). Under this definition the derivatives of $\log'$ are upper-bounded by $1/\lambda$, in contrast to $\log$, where the derivatives can diverge to infinity. The optimization methods we use are gradient-based methods (or more precisely, *subgradient*-based methods), and we have found them to be considerably more stable when the values for gradients do not diverge to infinity.

The modified objective remains convex.

5 A Stochastic Exponentiated-Gradient Algorithm for Optimization

We now describe an algorithm for optimizing the I2CR-2 problem in Figure 4. The algorithm is closely related to stochastic gradient ascent, but with two modifications:

- First, because the $t(f|e)$ and $d(i|j)$ parameters have simplex constraints (see Figure 1), we use *exponentiated gradient* (EG) updates. EG algorithms are gradient-based methods that maintain simplex constraints; see for example: (Kivinen and Warmuth, 1997; Beck and Teboulle, 2003; Collins et al., 2008).

- Second, the objective function in the I2CR-2 problem is convex, but is not differentiable (the gradient may not exist at all points). For this reason we use *subgradients* in the place of gradients. In spite of the non-differentiability of the objective function, subgradient methods still have strong convergence guarantees when combined with EG updates (e.g., the convergence proofs in (Beck and Teboulle, 2003)

go through with minor modifications; see also (Bertsekas, 1999)).

To derive the updates, recall that we are maximizing the following objective function:

$$\begin{aligned} & h(t,d) \\ = & \frac{1}{2|\mathcal{T}|} \sum_{k \in \mathcal{T}} \sum_{j=1}^{m_k} \log' \sum_{i=0}^{l_k} \min \left\{ t(f_j^{(k)} | e_i^{(k)}), d(i|j) \right\} \\ + & \frac{1}{2|\mathcal{T}|} \sum_{k \in \mathcal{T}} \sum_{j=1}^{m_k} \log' \sum_{i=0}^{l_k} \frac{t(f_j^{(k)} | e_i^{(k)})}{(L+1)} \, . \end{aligned} \quad (19)$$

Here we use $\mathcal{T}$ to denote the set $\{1 \ldots n\}$; we will see shortly why this notation is convenient. We use t and d to refer to the full set of t and d parameters respectively; $h(t,d)$ is the function to be maximized. Recall that $\log'(z) = \log(z + \lambda)$ where λ is a small positive parameter.

Given a concave function $f(x)$ where $x \in \mathbb{R}^d$, a subgradient of $f(x)$ at x is any vector $g(x) \in \mathbb{R}^d$ such that for any $y \in \mathbb{R}^d$,

$$f(y) \leq f(x) + g(x) \cdot (y - x) \, ,$$

where $u \cdot v$ is the inner product between vectors u and v. Subgradients are similar to gradients for differentiable concave functions, in that gradients satisfy the above property. Subgradients can be used in the place of gradients in many optimization algorithms (see for example (Bertsekas, 1999)).

The subgradients for the objective function in Eq. 19 take a simple form. First, define

$$\begin{aligned} R(j,k) &= \lambda + \sum_{i=0}^{l_k} t(f_j^{(k)} | e_i^{(k)}) \, , \\ Q(j,k) &= \lambda + \sum_{i=0}^{l_k} \min\{ t(f_j^{(k)} | e_i^{(k)}), d(i|j) \} \, , \end{aligned}$$

and

$$I(i,j,k) = \begin{cases} 1 & \text{if } t(f_j^{(k)} | e_i^{(k)}) \leq d(i|j) \\ 0 & \text{otherwise} \, . \end{cases}$$

Then the subgradients[2] are

$$\nabla t(f|e) = \frac{1}{2|\mathcal{T}|} \sum_{\substack{i,j,k: \\ f_j^{(k)} = f \\ e_i^{(k)} = e}} \left(\frac{1}{R(j,k)} + \frac{I(i,j,k)}{Q(j,k)} \right)$$

and

$$\nabla d(i|j) = \frac{1}{2|\mathcal{T}|} \sum_{k: i \leq l_k, j \leq m_k} \frac{1 - I(i,j,k)}{Q(j,k)} \, .$$

Exponentiated-gradient updates then take the following form:

$$t(f|e) \leftarrow \frac{t(f|e) \times \exp\{\gamma \times \nabla t(f|e)\}}{\sum_f t(f|e) \times \exp\{\gamma \times \nabla t(f|e)\}} \quad (20)$$

and

$$d(i|j) \leftarrow \frac{d(i|j) \times \exp\{\gamma \times \nabla d(i|j)\}}{\sum_i d(i|j) \times \exp\{\gamma \times \nabla d(i|j)\}} \, , \quad (21)$$

where $\gamma > 0$ is a constant step size in the algorithm. Note that the EG updates make use of subgradients, but maintain the simplex constraints on the t and d variables.

The method just described is a *batch* gradient method, where the entire training set $\mathcal{T} = \{1 \ldots n\}$ is used to derive the subgradients before the updates in Eqs. 20 and 21 are made. Many results in machine learning and NLP have shown that *stochastic gradient methods*, where a subset of the training examples is used before each gradient-based update, can converge much more quickly than batch gradient methods. In our notation, this simply involves replacing $\mathcal{T}$ by some subset $\mathcal{T}'$ of the training examples in the above definitions, where $|\mathcal{T}'|$ is typically much smaller than $|\mathcal{T}|$.

Figure 5 shows our final algorithm, a stochastic version of the exponentiated-gradient method. The method takes S passes over the data. For each pass, it randomly partitions the training set into mini-batches $\mathcal{T}_1 \ldots \mathcal{T}_K$ of size B, where B is an integer specifying the size of each mini-batch (in our experiments we used $B = 125$ or $B = 250$). The algorithm then performs EG updates using each mini-batch $\mathcal{T}_1 \ldots \mathcal{T}_K$ in turn. As can be seen in Table 3, our experiments show that the algorithm makes very significant progress in the first pass over the data, and takes very few iterations to converge to a good solution even though we initialized with uniform parameter values.

6 Experiments

In this section we describe experiments using the I2CR-2 optimization problem combined with the

[2] We set $\nabla t(f|e)$ and $\nabla d(i|j)$ as the subgradients for the objective function in Eq. 19 with respect to $t(f|e)$ and $d(i|j)$ respectively.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3. An integer B specifying the batch size. An integer S specifying the number of passes over the data. A step size $\gamma > 0$. A parameter $\lambda > 0$ used in the definition of $\log'$.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.

Definitions:

$$R(j,k) = \lambda + \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)})$$

$$Q(j,k) = \lambda + \sum_{i=0}^{l_k} \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\}$$

Initialization:

- $\forall e \in E,\ f \in D(e),\ t(f|e) = 1/|D(e)|$
- $\forall j \in [M],\ i \in [L]_0,\ d(i|j) = 1/(L+1)$

Algorithm:

for all $s = 1$ to S **do**
  Randomly partition $[n]$ into subsets $\mathcal{T}_1 \ldots \mathcal{T}_K$ of size B where $K = n/B$.
  for all $b = 1$ to K **do**
    $\forall e \in E,\ f \in D(e),\ \alpha(e,f) = 0$
    $\forall j \in [M],\ i \in [L]_0,\ \beta(i,j) = 0$
    for all $k \in \mathcal{T}_b$ **do**
      for all $j = 1$ to m_k **do**
        for all $i = 0$ to l_k **do**
          $\alpha(e_i^{(k)}, f_j^{(k)})$ += $1/(2R(j,k))$
          if $t(f_j^{(k)}|e_i^{(k)}) \leq d(i|j)$ **then**
            $\alpha(e_i^{(k)}, f_j^{(k)})$ += $1/(2Q(j,k))$
          else
            $\beta(i,j)$ += $1/(2Q(j,k))$
    $\forall e, f,\ t(f|e) = t(f|e) \exp(\gamma \times \alpha(e,f)/B)$
    $\forall i, j, d(i|j) = d(i|j) \exp(\gamma \times \beta(i,j)/B)$
    Renormalize t and d parameters to satisfy $\sum_f t(f|e) = 1$ and $\sum_i d(i|j) = 1$.

Output: t and d parameters.

Figure 5: The stochastic exponentiated-gradient algorithm for optimization of I2CR-2.

stochastic EG algorithm for parameter estimation. We first describe the data sets we use, and then describe experiments with the method, comparing our approach to results from IBM Model 2. We compare the various algorithms in terms of their accuracy in recovering alignments, using metrics such as F-measure and AER.

6.1 Data Sets

We use data from the bilingual word alignment workshop held at HLT-NAACL 2003 (Michalcea and Pederson, 2003). As a first dataset, we use the Canadian Hansards bilingual corpus, with 247,878 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data (note that we use a randomly chosen subset of the original training set of 1.1 million sentences, similar to the setting used in (Moore, 2004)). The development and test data have been manually aligned at the word level, annotating alignments between source and target words in the corpus as either "sure" (S) or "possible" (P) alignments, as described in (Och and Ney, 2003).

As a second data set, we used the Romanian-English data from the HLT-NAACL 2003 workshop. This consisted of a training set of 48,706 Romanian-English sentence-pairs, a development set of 17 sentence pairs, and a test set of 248 sentence pairs.

6.2 Methodology

For each of the models—IBM Model 1, IBM Model 2, and I2CR-2—we follow convention in applying the following methodology: first, we estimate the t and d parameters using models in both source-target and target-source directions; second, we find the most likely alignment for each development or test data sentence in each direction; third, we take the intersection of the two alignments as the final output from the model.

For the EG algorithm we use a batch size $B = 250$ and step size $\gamma = 0.5$ on the Hansards data, and $B = 125$ and $\gamma = 0.5$ for the Romanian-English data.

We report the performance of the models in terms of *Precision*, *Recall*, *AER*, and *F-Measure* as defined by (Och and Ney, 2003). If A is the set of alignments produced by an algorithm, S is the set of sure alignments as annotated in test data, and P is the set of possible alignments, then these quantities are defined as

$$\text{Recall} = \frac{|A \cap S|}{|S|} ,$$

$$\text{Precision} = \frac{|A \cap S|}{|A|} ,$$

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} ,$$

$$\text{F-Measure} = \frac{1}{\frac{.5}{\text{Recall}} + \frac{.5}{\text{Precision}}} .$$

Note that we report results in both AER and F-measure; however there is evidence (Fraser and Marcu, 2004) that F-measure is better correlated with translation quality when the alignments are used in a full system.

In training IBM Model 1 we follow (Moore, 2004) in running EM for 15 iterations. In training IBM Model 2 we first train IBM Model 1 for 15 iterations to initialize the t parameters, then train IBM Model 2 for a further 10 iterations. For the EG algorithm, we use 10 iterations over the training data for the Hansards data, and 15 iterations on the Romanian-English data (on the latter dataset results on the trial data showed that the method took slightly longer to converge). We report F-measure and AER results for each of the iterations under the IBM Model 2 and I2CR-2 models. See Table 1 for the results on the Hansards data, and Table 2 for the results on the English-Romanian dataset. It can be seen that both I2CR-2 and IBM Model 2 converge to a fairly stable result after 2-3 iterations. The two models give very similar levels of performance, for example after 10 iterations on the Hansard data IBM Model 2 gives 14.22 AER and 0.7516 F-Measure versus 14.60 AER and 0.7506 F-Measure for I2CR-2.

On the right, Table 3 shows the values of the objective function at each iteration when using the EG algorithm to optimize the I2CR-2 objective. The method makes a large amount of progress on the first iteration and then continues to improve. Finally, we note that the memory requirements for I2CR-2 and IBM2 are about the same, but that the time for one iteration of I2CR-2 on the Hansards data is approximately one hour, while the time for one iteration of IBM2 was approximately 10 minutes.

7 Conclusions and Future Work

We have introduced the first convex model for unsupervised learning of alignments in statistical machine translation with performance comparable to

Iteration	IBM2 AER	I2CR-2 AER	IBM2 F-Measure	I2CR-2 F-Measure
Test Set Statistics				
1	0.1491	0.1556	0.7530	0.7369
2	0.1477	0.1489	0.7519	0.7456
3	0.1451	0.1476	0.7527	0.7467
4	0.1426	0.1488	0.7536	0.7449
5	0.1422	0.1495	0.7535	0.7472
6	0.1431	0.1476	0.7511	0.7478
7	0.1434	0.1506	0.7506	0.7456
8	0.1437	0.1495	0.7501	0.7470
9	0.1434	0.1494	0.7501	0.7468
10	0.1422	0.1460	0.7516	0.7506
Development Set Statistics				
1	0.1871	0.1971	0.6823	.6676
2	0.1896	0.1760	0.6758	.6827
3	0.1964	0.1860	0.6648	.6739
4	0.1912	0.1835	0.6713	.6775
5	0.1884	0.1813	0.6740	.06773
6	0.1836	0.1851	0.6767	0.6811
7	0.1831	0.1806	0.6749	0.6765
8	0.1842	0.1843	0.6739	0.6775
9	0.1864	0.1928	0.6694	0.6640
10	0.1845	0.1829	0.6703	.6721

Table 1: Results on the Hansards data for IBM Model 2 and the I2CR-2 method.

Iteration	IBM2 AER	I2CR-2 AER	IBM2 F-Measure	I2CR-2 F-Measure
Test Set Statistics				
1	0.4041	0.5354	0.5959	0.4646
2	0.4010	0.4764	0.5990	0.5256
3	0.4020	0.4543	0.5980	0.5457
4	0.4012	0.4384	0.5988	0.5617
5	0.4003	0.4277	0.5997	0.5723
6	0.3990	0.4266	0.6010	0.5834
7	0.4000	0.4162	0.6000	0.5838
8	0.4023	0.4114	0.5977	0.5886
9	0.4022	0.4081	0.5978	0.5919
10	0.4027	0.4043	0.5973	0.5957
11	0.4031	0.4040	0.5969	0.5960
12	0.4042	0.4027	0.5958	0.5973
13	0.4043	0.4021	0.5957	0.5979
14	0.4062	0.4007	0.5938	0.5993
15	0.4057	0.4014	0.5943	0.5986
Development Set Statistics				
1	0.4074	0.5841	0.5926	0.4159
2	0.3911	0.4938	0.6089	0.5062
3	0.3888	0.4673	0.6112	0.5327
4	0.3904	0.4596	0.6096	0.5404
5	0.3881	0.4463	0.6119	0.5537
6	0.3904	0.4306	0.6096	0.5694
7	0.3936	0.4175	0.6094	0.5826
8	0.3897	0.4060	0.6103	0.5940
9	0.3961	0.4014	0.6039	0.5986
10	0.3970	0.4072	0.6030	0.5928
11	0.4018	0.3956	0.5982	0.6044
12	0.4035	0.3931	0.5965	0.6069
13	0.4035	0.3862	0.5965	0.6138
14	0.4014	0.3908	0.5986	0.6092
15	0.4063	0.3858	0.5937	0.6142

Table 2: Results on the English-Romanian data for IBM Model 2 and the I2CR-2 method.

Iteration	EF Objective	FE Objective
0	-99.6053	-79.5566
1	-32.4528	-27.4925
2	-31.1641	-26.262
3	-30.6311	-25.7093
4	-30.3367	-25.3714
5	-30.1428	-25.1456
6	-30.0000	-24.992
7	-29.8736	-24.8605
8	-29.8093	-24.7551
9	-29.7326	-24.684
10	-29.6771	-24.6099

Table 3: Objective values for the EG algorithm optimization of I2CR-2 at each iteration. "EF Objective" corresponds to training a model with $t(e|f)$ parameters, "FE Objective" corresponds to the reverse direction, with $t(f|e)$ parameters. Iteration 0 corresponds to the objective value under the initial, uniform parameter values.

the commonly-used IBM Model 2. We believe that introducing convexity without sacrificing performance will open the door to further improvements in this area. Future work will consider ways to speed up our algorithm and extensions of the method to more complex alignment models.

Acknowledgments

Michael Collins is partly supported by NSF grant IIS-1161814. Cliff Stein is partly supported by NSF grant CCF-0915681. The authors thank Sasha Rush for his help with implementation questions. We also thank the anonymous reviewers for many useful comments; we hope to pursue the comments we were not able to address in a followup paper.

References

Peter L. Bartlett, Ben Taskar, Michael Collins and David Mcallester. 2004. Exponentiated Gradient Algorithms for Large-Margin Structured Classification. *In Proceedings of NIPS*.

Amir Beck and Marc Teboulle. 2003. Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Optimization. *Operations Research Letters*, 31:167-175.

Dimitris Bertsimas and John N. Tsitsiklis. 1997. Introduction to Linear Programming. Athena Scientific.

Dimitris Bertsimas. 2005. Optimization Over Integers. Dynamic Ideas.

Dimitri P. Bertsekas. 1999. Nonlinear Optimization. Athena Press.

Steven Boyd and Lieven Vandenberghe. 2004. Convex Optimization. Cambridge University Press.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263-311.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. *In Proceedings of the ACL.*

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras and Peter L. Bartlett. 2008. Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks. *Journal Machine Learning,* 9(Aug): 1775-1822.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood From Incomplete Data via the EM Algorithm. *Journal of the royal statistical society, series B*, 39(1):1-38.

Alexander Fraser and Daniel Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Journal Computational Linguistics*, 33(3): 293-303.

Kuzman Ganchev, Joao V. Graca, Jennifer Gillenwater, Ben Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning*, 11(July): 2001-2049.

Joao V. Graca, Kuzman Ganchev and Ben Taskar. 2007. Expectation Maximization and Posterior Constraints. *In Proceedings of NIPS.*

Aria Haghighi, John Blitzer, John DeNero and Dan Klein. 2009. Better Word Alignments with Supervised ITG Models. *In Proceedings of the ACL.*

Darcey Riley and Daniel Gildea. 2012. Improving the IBM Alignment Models Using Variational Bayes. *In Proceedings of the ACL.*

Yuhong Guo and Dale Schuurmans. 2007. Convex Relaxations of Latent Variable Training. *In NIPS.*

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael Jordan. 2008. Word Alignment via Quadratic Assignment. *In Proceedings of the HLT-NAACL.*

Phillip Koehn. 2008. Statistical Machine Translation. Cambridge University Press.

Kivinen, J., Warmuth, M. 1997. Exponentiated Gradient Versus Gradient Descent for Linear Predictors. *Information and Computation*, 132, 1-63.

Percy Liang, Ben Taskar and Dan Klein. 2006. Alignment by Agreement. *In Proceedings of NAACL.*

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. *In Proceedings of the EMNLP.*

Andre F. T. Martins, Noah A. Smith and Eric P. Xing. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. *In Proceedings of the EMNLP*.

Rada Michalcea and Ted Pederson. 2003. An Evaluation Exercise in Word Alignment. *HLT-NAACL 2003: Workshop in building and using Parallel Texts: Data Driven Machine Translation and Beyond*.

Robert C. Moore. 2004. Improving IBM Word-Alignment Model 1. *In Proceedings of the ACL*.

Stephan Vogel, Hermann Ney and Christoph Tillman. 1996. HMM-Based Word Alignment in Statistical Translation. *In Proceedings of COLING*.

Franz Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational-Linguistics*, 29(1): 19-52.

Libin Shen, Jinxi Xu and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. *In Proceedings of the ACL-HLT*.

Ben Taskar, Simon Lacoste-Julien and Dan Klein. 2005. A Discriminative Matching Approach to Word Alignment. *In Proceedings of the EMNLP*.

Kristina Toutanova and Michel Galley. 2011. Why Initialization Matters for IBM Model 1: Multiple Optima and Non-Strict Convexity. *In Proceedings of the ACL*.

Kenji Yamada and Kevin Knight. 2001. A Syntax-Based Statistical Translation Model. *In Proceedings of the ACL*.

Kenji Yamada and Kevin Knight. 2002. A Decoder for Syntax-Based Statistical Machine Translation. *In Proceedings of the ACL*.

Ashish Vaswani, Liang Huang and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the $L0$-norm. *In Proceedings of the ACL*.

Pair Language Models for Deriving Alternative Pronunciations and Spellings from Pronunciation Dictionaries

Russell Beckley
Oregon Health and Science University
beckleyr@ohsu.com

Brian Roark
Google Inc.
roarkbr@gmail.com

Abstract

Pronunciation dictionaries provide a readily available parallel corpus for learning to transduce between character strings and phoneme strings or vice versa. Translation models can be used to derive character-level paraphrases on either side of this transduction, allowing for the automatic derivation of alternative pronunciations or spellings. We examine finite-state and SMT-based methods for these related tasks, and demonstrate that the tasks have different characteristics – finding alternative spellings is harder than alternative pronunciations and benefits from round-trip algorithms when the other does not. We also show that we can increase accuracy by modeling syllable stress.

1 Introduction

Robust processing of speech and language requires dealing with variation in language production, either in terms of pronunciation in the spoken domain or spelling in the written domain. Predicting the intended words of an acoustic or textual sequence is an important recognition task, often required for downstream processing such as spoken language understanding or knowledge extraction. Informal text genres, such as those found in social media, share some characteristics with speech; in fact such text is often informed by pronunciation variation. For example, consider the following tweet:

> He aint gotta question my loyalty, cuz he knw wen sh!t get real. Ill be right here!

where several tokens (e.g. "cuz", "wen") represent spelling alternations related to pronunciation. Work in text normalization and spelling correction – e.g., Toutanova and Moore (2002); Li and Liu (2012) – has included pronunciation information to improve recognition of the intended word, via grapheme to phoneme (g2p) conversion modeling derived from pronunciation dictionaries.

Pronunciation dictionaries provide natural parallel corpora, with strings of characters paired to strings of phones. Thus, standard lexicons have been used in recent years with machine translation systems such as Moses (Koehn et al., 2007), to train g2p systems (Laurent et al., 2009; Gerosa and Federico, 2009). Further, other algorithms using such dictionaries also use translation phrase tables, but not for translation tasks. For example, data-driven paraphrasing methods (Bannard and Callison-Burch, 2005) use translation phrase-tables as a "pivot" to learn sets of phrases which translated to the same target phrase. In a similar manner, with a pronunciation dictionary instead of a phrsetable, pivoting can be used to learn alternative pronunciations (Karanasou and Lamel, 2010), i.e., direct phoneme-to-phoneme (p2p) "translation" systems that yield alternative pronunciations. Alternatively, round-trip translation could be used, e.g., to map from letter strings to phone strings in one step, then from the resulting phone strings to letter strings in a second step, as the means to find alternative spellings (Li and Liu, 2012).

In this study, we explore dictionary-derived models to find either alternative pronunciations or alternative spellings, using either direct (p2p or g2g) or round-trip algorithms (p2g2p or g2p2g). We compare methods based on weighted finite-state transducers (WFST) with phrase-based models trained with Moses. Our main interest is to evaluate Karanasou and Lamel (2010) methods – shown to be useful for deriving alternative pronunciations – for deriving alternative spellings, and thus to determine the relative difficulty of these two tasks. We also examine when, if ever, round-trip processing yields benefits over direct transduction. Our results indicate that real alternative pronunciations are substantially easier to find than real alternative spellings, partic-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1584–1589,
Seattle, Washington, USA, 18-21 October 2013.

ularly when pronunciation features such as syllable stress are available. Second, round trip translation yields no gain (and some loss) over direct transduction for finding alternative pronunciations, yet yields some modest gains for finding alternative spellings. Further, WFST methods perform as well as or better than Moses trained models. Finally, combining the methods yields further gains, indicating that the models are learning complementary sets of patterns.

The primary contribution of this work is to introduce a competitive method of building and using pair language model WFSTs for generating alternative spellings and pronunciations which reflect real-world variability. This could improve results for downstream processes, e.g., epidemiological studies (Chew and Eysenbach, 2010) or sentiment analysis (Barbosa and Feng, 2010) derived from social media text. Further, we present a controlled comparison between the two tasks, and demonstrate that they differ in terms of task difficulty

2 Related work

Text normalization has been a major focus in text-to-speech (TTS) research for many years. Notably, Sproat et al. (2001) deemed it a problem in itself, rather than *ad hoc* preparatory work, and defined many of the issues involved, as well as offering a variety of initial solutions. Similar approaches apply to automatic spelling correction, where Toutanova and Moore (2002) extended the noisy channel spelling correction method of Brill and Moore (2000), by modeling pronunciation alternations to infer from misspellings to correct spellings. Similarly, Li and Liu (2012) extended the character-based translation approach to text normalization of Pennell and Liu (2011), by adding an additional round-trip translation to-and-from pronunciations. Karanasou and Lamel (2010) used Moses to generate alternative pronunciations from an English dictionary, using both direct and round-trip methods. They validated their systems on a set of words with multiple pronunciations, measuring the degree to which alternative pronunciations are generated from one of the given pronunciations. Our task and method of evaluation is similar to theirs, though we also look at alternative spellings.

3 Methods

To generate alternative spellings and pronunciations, we built phrase-based translation and finite-state transduction models from a parallel corpus. When pronunciations were part of the model – i.e., not direct grapheme-to-grapheme – we included conditions with and without vowel stress.

3.1 Corpus

Our training corpus is the CMU Pronouncing Dictionary[1], which contains nearly 130k entries. From this corpus, we identified homophone sets, i.e., sets of multiple spellings sharing the same pronunciation, such as "colonel" and "kernel". We found 9,977 such sets, and randomly selected 1000 for testing; the rest we used for training. Each set had, on average, 2.46 members. We also identified homograph sets, i.e., sets of multiple pronunciations all spelled the same, such as potato (/potato/ and /pəteto/). We found 8,216 such homograph sets, and randomly selected 1000 for testing; the rest we used for training. These sets averaged 2.13 members.

We construct seven parallel training corpora from the lexicon, each disjoint from its relevant test set. For round-trip models, the parallel corpus is each grapheme string in the lexicon aligned with its phoneme string, if neither the grapheme string nor phoneme string appear in the test set. There are four such corpora, corresponding to these options: stress or no stress, and g2p2g or p2g2p. The g2p2g and p2g2g conditions require different corpora because they are differently partitioned for testing. For direct grapheme-to-grapheme training sets, non-homophone words are self-aligned; for homophones, from each homophone set, each possible pair of spellings are aligned. For example, for a pronunciation with four spellings—a, b, c, and d—there would be six alignments: a:b, a:c, a:d, b:c, b:d, c:d. Similarly for direct phoneme-to-phoneme training sets, non-homograph words are self-aligned; words from the training homograph sets are pairwise aligned in all pairings. There are two direct p2p corpora: with and without stress.

3.2 Phrase-based translation models

As a baseline system, we used the Moses statistical machine translation package (Koehn et al., 2007) to build grapheme-based and phoneme-based translation systems, using a bigram language model.[2] These are trained on the parallel corpus resulting from the homophone or homograph sets detailed in

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[2] Higher order language models yielded no improvements.

the previous section for the direct methods. For this paper, we did not perform round-trip translation with Moses, rather present it as a baseline for the direct approach.

3.3 Pair language models

Our weighted finite-state transducer approach is based on pair language models (Bisani and Ney, 2008; Deligne and Bimbot, 1997; Ghoshal et al., 2009), or, more recently, (Sagae et al., 2012).) The basic idea in a pair LM is to align strings, then train a language model over sequences whose symbols are the input:output pairs of the alignment. This language model can then be converted to transducers. For a g2g example, homophones "their" and "there" are aligned via the standard Levenshtein edit distance algorithm as "t:t h:h e:e i:ϵ r:r ϵ:e". A trigram model over these x:y strings would use standard n-gram modeling to estimate, for example, P(ϵ:e | i:ϵ r:r); i.e., the probability of a silent "r" in a given context.

Building the pair language model transducers requires two phases. In the first phase we create new corpora by aligning the elements of the parallel corpora outlined above. In the second phase we use these corpora of string alignments to build a pair language model.

3.3.1 Alignment and Corpora Building

We use extensions to the Levenshtein edit distance algorithm to align g2g, p2p and g2p strings, with substitution matrices created to provide useful alignments (Wagner and Fischer, 1974). As in Brill and Moore (2000), we allow for certain multi-symbol strings to be substituted with a single cost, e.g., substituting 'th' with /θ/ in g2p alignment. For g2g alignment, our substitution cost is 0 for identity and 2 for a few pairs of commonly interchangeable graphemes, such as 'c' and 'k'. Other substitutions are not permitted, and delete and insertion have cost 10. For p2p alignment there are two conditions, with and without stress. Without vowel stress, no substitutions other than identity are allowed; with vowel stress, substitution cost is 2.5 for the same vowel with differing stress; and 5.0 if substituting a vowel with another vowel. Other substitutions are not permitted, and, again, delete and insertion have cost 10.

For training round-trip models, we have to perform g2p and p2g alignment, with differing alphabets on the input and output of the alignment. We begin with a basic substitution table that allows graphemes and their most likely phonemes to align. We then re-estimate the substitution costs based on relative frequency estimation (-logP), and also aggregate sequences of consecutively deleted graphemes so that they collectively map to a single phoneme. For example, given the alignment 'o:/a/ u:/ϵ/ g:/ϵ/ h:/ϵ/ t:/t/', ('ought', /at/), we make a new rule: ough:/a/, and give it a cost based on its relative frequency. Grapheme strings that appear sufficiently often with a given phoneme will thus accumulate sufficient probability mass to compete.

Each alignment produced as described above is a string in a training corpus for creating a pair language model. As such, each alignment pair (e.g. a:/ə/) is a token.

3.3.2 From Corpora to WFSTs

We use the open source OpenGrm NGram library (Roark et al., 2012) to build 5-gram language models from the strings of input:output pairs. These langauge models are encoded as weighted finite-state acceptors in the OpenFst format (Allauzen et al., 2007). We shrink the models with the `ngramshrink` command, using the relative entropy method (Stolcke, 1998), with the "theta" threshold set at $1.0e-6$. These finite state acceptors are then converted into transducers by modifying the arcs: split the labels of each arc, x:y, making x the input label for that arc, and y the output label. Thus traversing such an arc will consume an x a return a y. Such pair language models we use for all WFST methods discussed here.

3.4 Producing k-best output

Each tested input string, spelling or pronunciation, is encoded as a cost-free linear chain WFST and composed with a pair language model transducer described in the previous section. The resulting lattice is converted to an acceptor by projecting onto its output labels, i.e., for each arc, the input label is set to the value of the output label. Epsilons are then removed and the result is determinized. The k-best paths are extracted using the shortest path algorithm in the OpenFst library.

For direct models (g2g and p2p), the k-best output from this first transduction is our result, ranked according the probability of each path. For round-trip methods (e.g. g2p2g), however, we do a second transduction in the other direction. For example, for

g2p2g, the first transduction would have transduced from a spelling to a set of candidate pronunciations; the second transduction will transduce from pronunciations to spellings. For this second transduction, we take each string s from the k-best list from the first transduction, and process them as we did in the first transduction, now using the inverse transducer. So, for each s in the first k-best list, we now have a k-best list from the second transduction. Thus, for the original input string, we have up to k^2 alternatives. Finally, we score each alternative by combining their scores from both transductions.

Let $\bar{p}$ represent a phoneme string, and $\bar{g}$ a grapheme string. If we perform a transduction from $\bar{p}$ to $\bar{g}$, the weights from the transducer provide the (negative log) joint probability $\mathrm{P}(\bar{p}, \bar{g})$. By performing a soft-max normalization over the k-best list output, we obtain the (negative log) conditional probability $\mathrm{P}(\bar{g} \mid \bar{p})$. For round-trip methods, we take the product of the conditional probability in each direction, and marginalize out the intermediate grapheme sequence, i.e.,

$$\mathrm{P}(\bar{p}_2 \mid \bar{p}_1) \;=\; \sum_{\bar{g}} \mathrm{P}(\bar{p}_2 \mid \bar{g})\, \mathrm{P}(\bar{g} \mid \bar{p}_1).$$

4 Experimental results

For evaluation purposes, we reserved a set of 1000 test homophone sets and 1000 test homograph sets, as described in Section 3.1. From each set, we generate alternatives from the longest set member (ties broken alphabetically) and examine the resulting k-best list for presence of other members of the set. Note that the input string itself is not a target, and, before evaluation, is removed from the k-best list. Recall is the proportion of the k-best list returned by the system:

$$\text{Recall}(\{\text{k-best}\}) \;=\; \frac{\mid \{\text{k-best}\} \cap \{\text{gold-list}\} \mid}{\mid \{\text{gold-list}\} \mid}$$

.

Results for generating alternative pronunciations are listed in Table 1; those for generating alternative spellings are in Table 2. For alternative spellings, we also present results that combine the outputs of direct, round-trip (no stress) and Moses into a single list using a simple ranked voting scheme (simple Borda count).

A noteworthy result is the apparent usefulness of stress modeling for predicting pronunciation variation using WFSTs with the direct method; this is seen in the first two data columns of 1. This suggests that stress has an effect on phoneme alteration, something we discuss in more detail in Section 5.

Recall: Alternative Pronunciations						
k-best size	pair language model				Moses	
	Direct		Roundtrip		Direct	
	stress	none	stress	none	stress	none
1	0.43	0.54	0.38	0.37	0.44	0.46
3	0.77	0.71	0.59	0.58	0.60	0.62
5	0.82	0.77	0.66	0.66	0.64	0.65
10	0.86	0.80	0.73	0.76	0.68	0.69

Table 1: Recall for generating alternative pronunciations

However, while providing a large gain in the p2p condition, pronunciation modeling gives small or negative effects elsewhere. In the round trip methods, the effects of stress are lost: stress has little influence of how a particular phoneme is spelled. Thus, graphemes do not retain much stress information, hence any pass through the orthographic domain will shed it.

Recall is higher for alternative pronunciations than for alternative spellings. One reason for this is that spellings in our test set average eight letters, whereas the pronunciations average around five phonemes. Furthermore, the average Levenshtein distance between original spellings and their target alternatives, is 2.6, while for pronunciations, it is 2.2. Combining these factors, we see that, for spellings, more edit operations are required, and there are more symbols to which to apply them. Therefore, for spellings, there are more incorrect candidates.

The results also show gains resulting from the roundtrip method when applied to finding alternative spellings, but no such gains when roundtrip methods are applied to alternative pronunciations. Suppose, when seeking alternatives for some spelling, we alter grapheme g_1 to g_2. With a direct method, we must have instances of g_1 mapping to g_2 in the training set. The roundtrip method, however, is less constrained: there must exist some phoneme p_1 in the training set such that g_1 maps to p_1, and p_1 maps to g_2; thus, the set of possible alternations at testing are $\{g_1 \rightarrow p_1\} \times \{p_1 \rightarrow g_2\}$. This argument also applies to finding alternative pronunciations. Thus the roundtrip method offers more possible mappings. These extra possible mappings may be helpful or harmful, depending on how likely they are compared to the possible mappings they displace. Why are they helpful for alternative spellings, but not for al-

Recall: Alternative Spellings					
k-best size	pair language model			Moses	Comb.
	Direct	Roundtrip		Direct	Direct
	none	stress	none	none	none
1	0.19	0.19	0.19	0.20	0.30
3	0.36	0.38	0.37	0.39	0.52
5	0.45	0.49	0.48	0.48	0.60
10	0.55	0.63	0.62	0.60	0.69

Table 2: Recall for generating alternative spellings

ternative pronunciations? We discuss one possible explanation in Section 5.

Comparing Moses to the pair language model methods, Moses does slightly better for smaller n ($n = 1, 3$), and slightly worse for larger n ($n = 10$). Our only partial explanation for this is that Moses does well at weighing alternatives but, possibly, does not generate a large number of viable alternatives. System combination yields solid gains in finding alternative spellings, demonstrating that these different systems are coming up with diverse options.

Finally, we note that many of the false positive pronunciations given by the WFST system are plausibly correct although they are not included in the CMU dictionary. For example, for the spelling, *adequate*, the CMU dictionary provides two pronunciations: /ædəkwət/ and /ædəkwet/. Meanwhile, the p2p WFST system (with stress modeling) produces /ædəkwIt/. This suggests that we can learn from CMU dictionary to predict actual pronunciations that CMU dictionary does not itself list.

5 Discussion and Summary

The experimental results demonstrated the utility of stress modeling for generating alternative pronunciations, which we suggested was due to the impact of stress on phoneme alternation. To examine this more closely, we looked at each phoneme, stress class, (ph, s)—e.g. (/ə/, primary)—and determined how likely is an occurrence of (ph, s) to have an alternative phoneme in a homograph set. We found that primary and secondary stressed vowels had an alteration probability of 0.017, while non-stressed vowels had an alteration probability of 0.036. This difference should be picked up in the transition probabilities of our WFSTs, resulting in a preference for alterations of unstressed vowels. This is analogous to results found in (Greenberg et al., 2002) for spontaneous American English discourse. A further analysis of the system output might shed more light on relationships between stress and phoneme choice.

Why are round-trip methods useful for finding alternative spellings but not for finding alternative pronunciations? One possible explanation is that the variety of orthographic alternations is greater than that of pronunciation alternations. Thus, the training set for spelling may provide less relative coverage of the alternations in its test set than the training set for pronunciation provides for *its* test set. This is supported by the fact that pronunciation recall exceeds spelling recall. The roundtrip method allows for finding mappings not seen in training. These extra mappings might be no better for spelling than they are for pronunciation, but for spelling, the mappings they replace in the k-best list are worse, so they yield an improvement. For pronunciation, the mappings they replace in the k-best list are better, so they yield a loss. Further research is required to validate this explanation.

Ultimately, we would like to apply these methods to the normalization of social media text, especially to find alternative spellings based on alternative pronunciations. To apply such methods to, say, Twitter normalization requires a sizable corpus mapping canonical spellings to non-standard spellings. To assess domain portability, we applied a model built from the CMU dictionary to just over 100 alternative spellings observed in a small Twitter collection. Using the direct g2g method, we generated alternative spellings from the canonical spelling of each term, and measured the recall of the output, i.e., whether the observed alternatives were present in the k-best list. Recall was extremely low (less than 5%), suggesting that the type of orthographic alterations that are found in dictionary pronunciations are very different from the orthographic variations found on Twitter, and that those differences have a profound effect on our ability to recover alternatives.

In sum, we have presented a small study of the utility of pronunciation dictionaries for finding spelling and pronunciation alternatives, demonstrating key differences between these tasks.

Acknowledgments

This work was supported in part by NSF grant #BCS-1049308. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Twelfth International Conference on Implementation and Application of Automata (CIAA 2007), Lecture Notes in Computer Science*, volume 4793, pages 11–23.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434 – 451.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293.

Cynthia Chew and Gunther Eysenbach. 2010. Pandemics in the age of twitter: content analysis of tweets during the 2009 h1n1 outbreak. *PloS one*, 5(11):e14118.

Sabine Deligne and Frdric Bimbot. 1997. Inference of variable-length linguistic and acoustic units by multigrams. *Speech Communication*, 23(3):223 – 241.

Matteo Gerosa and Marcello Federico. 2009. Coping with out-of-vocabulary words: open versus huge vocabulary asr. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4313–4316.

A. Ghoshal, M. Jansche, S. Khudanpur, M. Riley, and M. Ulinski. 2009. Web-derived pronunciations. In *Proc. ICASSP*.

Steven Greenberg, Hannah Carvey, and Leah Hitchcock. 2002. The relation between stress accent and pronunciation variation in spontaneous american english discourse. In *In Proceedings of ISCA Workshop on Prosody in Speech Processing (Speech Prosody 2002), Aix-enProvence*.

Panagiota Karanasou and Lori Lamel. 2010. Comparing SMT methods for automatic generation of pronunciation variants. In *Advances in Natural Language Processing*, pages 167–178. Springer.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Antoine Laurent, Paul Deléglise, Sylvain Meignier, and France Spécinov-Trélazé. 2009. Grapheme to phoneme conversion using an smt system. In *Proceedings of Interspeech*, pages 708–711.

Chen Li and Yang Liu. 2012. Normalization of text messages using character-and phone-based machine translation approaches. In *Proceedings of Interspeech*.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of IJCNLP*, pages 974–982.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66.

Kenji Sagae, Maider Lehr, Emily Tucker Prud'hommeaux, Puyang Xu, Nathan Glenn, Damianos Karakos, Sanjeev Khudanpur, Brian Roark, Murat Saraclar, Izhak Shafran, Daniel M. Bikel, Chris Callison-Burch, Yuan Cao, Keith Hall, Eva Hasler, Philipp Koehn, Adam Lopez, Matt Post, and Darcey Riley. 2012. Hallucinated n-best lists for discriminative language modeling. In *ICASSP*, pages 5001–5004. IEEE.

Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Prior Disambiguation of Word Tensors for Constructing Sentence Vectors

Dimitri Kartsaklis
University of Oxford
Department of
Computer Science
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
dimitri.kartsaklis@cs.ox.ac.uk

Mehrnoosh Sadrzadeh
Queen Mary University of London
School of Electronic Engineering
and Computer Science
Mile End Road
London, E1 4NS, UK
mehrs@eecs.qmul.ac.uk

Abstract

Recent work has shown that compositional-distributional models using element-wise operations on contextual word vectors benefit from the introduction of a prior disambiguation step. The purpose of this paper is to generalise these ideas to tensor-based models, where relational words such as verbs and adjectives are represented by linear maps (higher order tensors) acting on a number of arguments (vectors). We propose disambiguation algorithms for a number of tensor-based models, which we then test on a variety of tasks. The results show that disambiguation can provide better compositional representation even for the case of tensor-based models. Furthermore, we confirm previous findings regarding the positive effect of disambiguation on vector mixture models, and we compare the effectiveness of the two approaches.

1 Introduction

Distributional models of meaning have been proved extremely useful for a number of natural language processing tasks, ranging from thesaurus extraction (Curran, 2004) to topic modelling (Landauer and Dumais, 1997) and information retrieval (Manning et al., 2008), to name just a few. These models are based on the *distributional hypothesis* of Harris (1968), which states that the meaning of a word depends on its context. This idea allows the words to be represented by vectors of statistics collected from a sufficiently large corpus of text; each element of the vector reflects how many times a word co-occurs in the same context with another word of the vocabulary. However, due to the generative power of natural language, which is able to produce infinite new structures from a finite set of resources (words), no text corpus, regardless of its size, can provide reliable distributional representations for anything longer than single words or perhaps very short phrases consisting of two words; in other words, this technique cannot scale up to the phrase or sentence level.

Much research activity has been recently dedicated to provide a solution to this problem: although the direct construction of a sentence vector is not possible, we might still be able to synthetically create such a vectorial representation by somehow *composing* the vectors of the words that comprise the sentence. Towards this goal, researchers have employed a variety of approaches that roughly fall into two general categories. Following an influential work (Mitchell and Lapata, 2008), the models in the first category compute a sentence vector as a mixture of the original word vectors, using simple operations such as element-wise multiplication and addition; we refer to these models as *vector mixtures*. The main characteristic of these models is that they do not distinguish between the type-logical identities of the different words: an intransitive verb, for example, is of the same order as its subject (a noun), and both will contribute equally to the composite sentence vector.

However, this symmetric treatment of composition seems unjustified from a formal semantics point of view. Words with special meanings, such as verbs and adjectives, are usually seen as functions acting on, hence modifying, a number of arguments rather than lexical units of the same order as them; an adjective, for example, is a function that returns a modified version of its input noun. Inspired from

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1590–1601,
Seattle, Washington, USA, 18-21 October 2013.

this more-aligned-to-formal-semantics view, a second research direction aims to represent relational words as linear maps (tensors of various orders) that can be applied to one or more arguments (vectors). Baroni and Zamparelli (2010), for example, model adjectives as matrices which, when matrix-multiplied with a noun vector, will produce a vectorial representation of the specific adjective-noun compound. The notion of a framework where relational words are entities living in vector spaces of higher order than nouns, which are simple vectors, has been formalized by Coecke et al. (2010) in the context of the abstract mathematical framework of compact closed categories. We refer to this class of models as *tensor-based*.

Regardless of the way they approach the representation of relational words and their composition operation, however, most current compositional-distributional models do share a common feature: they all rely on ambiguous vector representations, where all the senses of a polysemous word, such as the verb 'file' (which can mean *register* or *smooth*), are merged into the same vector or tensor. At least for the vector mixture approach, this practice has been proved suboptimal: Reddy et al. (2011) and Kartsaklis et al. (2013) test a number of simple multiplicative and additive models using disambiguated vector representations on various tasks, showing that the introduction of a disambiguation step prior to actual composition can indeed increase the quality of the composite vectors. However, the fact that disambiguation can be beneficial for models based on vector mixtures is not very surprising. Both additive and multiplicative compositions are but a kind of average of the vectors of the words in the sentence, hence can directly benefit from the provision of more accurate starting points. Perhaps a more interesting question, and one that the current paper aims to address, is to what extent disambiguation can also provide benefits for tensor-based approaches, which in general constitute more powerful models for natural language (see discussion in Section 2).

Specifically, this paper aims to: (a) propose disambiguation algorithms for a number of tensor-based distributional models; (b) examine the effect of disambiguation on tensors for relational words; and (c) meaningfully compare the effectiveness of tensor-based against vector mixture models in a number of tasks. Based on the generic procedure of Schütze (1998), we propose algorithms for a number of tensor-based models, where the composition is modelled as the application of linear maps (tensor contractions). Following Mitchell and Lapata (2008) and many others, we test our models on two disambiguation tasks similar to that of Kintsch (2001), and on the phrase similarity task introduced in (Mitchell and Lapata, 2010). In almost every case, the results show that disambiguation can make a great difference in the case of tensor-based models; they also reconfirm previous findings regarding the effectiveness of the method for simple vector mixture models.

2 Vectors vs tensors

The simple models of Mitchell and Lapata (2008) constitute the easiest and perhaps the most intuitive way of composing two or more vectors: each element of the resulting vector is computed as the sum or the product of the corresponding elements in the input vectors (left part in Figure 1). In the case of addition, the components of the output vector are simply the cumulative scores of the corresponding input components. So in a sense the output element embraces both input elements, resembling a *union* of the input features. On the other hand, the element-wise multiplication of two vectors can be seen as the *intersection* of their features: a zero element in one of the input vectors will eliminate the corresponding feature in the output, no matter how high the other input component was. In addition to failing to identify the special roles of words in a sentence, vector mixture models disregard grammar in another way: the commutativity of operators make them a bag-of-words approach, where the meaning of sentence 'dog bites man' is equated to that of 'man bites dog'.

On the contrary to the above element-wise treatment, a compositional approach based on linear maps computes each element of the resulting vec-

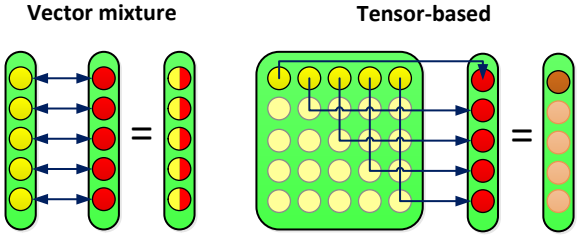


Figure 1: Vector mixture and tensor-based models for composition. In the latter approach, the ith element of the output vector is the linear combination of the input vector with the ith row of the matrix.

tor via a linear combination of *all* the elements of the input vector (right part of Figure 1); in other words, possible interdependencies between different features are also taken into account, offering (in principle) more power. Furthermore, by design, the bag-of-words problem is not present here. Overall, tensor-based models offer a more complete and linguistically motivated solution to the problem of composition. For example, one can consider building linear maps for prepositions and logical words, rather than treating them as noise and discard them, as commonly done in the vector mixture models.

3 Disambiguation in vector mixtures

For a compositional model based on vector mixtures, polysemy of words can be a critical factor. Pulman (2013) and Kartsaklis et al. (2013) point out that the element-wise combination of "ambiguous" vectors produces results that are hard to interpret; the composed vector is not a purely compositional representation but a product of two tasks that take place in parallel: composition and *some* amount of disambiguation that emerges as a side-effect of the compositional process, leaving the resulting vector in an intermediate state.

This effect is demonstrated in Figure 2, which shows the composition of the ambiguous verb 'run' (with meanings *moving fast* and *dissolving*) with the subject 'horse'. The first three components of our toy vector space are related to the *dissolving* meaning, while the last three of them to the *moving fast* meaning. An ambiguous vector for 'run' will have non-zero values for every component. On the other hand, we would expect the vector for 'horse' to have high values for the 'race', 'gallop', and 'move' components, and very low values (but not necessarily zero) for the dissolving-related ones—it is always possible for the word 'horse' to appear in the same context with the word 'painting', for example. The left part of Figure 2 shows what happens when the ambiguous 'run' vector is used; the multiplication with the 'horse' vector will produce an impure result, half affected by composition and half by disambiguation. However, what we really want is a vector where all the dissolving-related components will be eliminated, since they are irrelevant to the way the word 'run' is used in the sentence. In order to achieve this, we have to introduce a disambiguation step prior to composition (right part of Figure 2).

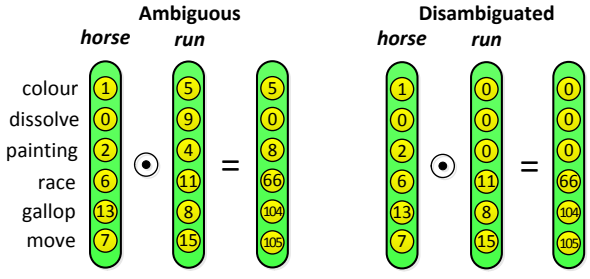


Figure 2: The effect of disambiguation on vector composition. The numbers are (artificial) co-occurrence counts of each target word with the 6 basis words on the left.

These ideas are experimentally verified in the works of Reddy et al. (2011) and Kartsaklis et al. (2013); Pulman (2013) also presents a comprehensive analysis of the problem. What remains to be seen is if disambiguation can also provide benefits for the linguistically motivated setting of tensor-based models, the principles of which are shortly discussed in the next section.

4 Tensors as multilinear maps

A *tensor* is a geometric object that can be seen as the generalization of the familiar notion of a vector in higher dimensions. The *order* of a tensor is the number of its dimensions; in other words, the number of indices we need to fully describe a random element of the tensor. Hence, a vector is a tensor of order 1, a matrix is a tensor of order 2, and so on. Tensors and multilinear maps stand in one-to-one correspondence, as stated by the following well-known "map-state" isomorphism (Bourbaki, 1989):

$$f : V_1 \rightarrow \ldots \rightarrow V_j \rightarrow V_k \cong V_k \otimes V_j \otimes \ldots \otimes V_1 \quad (1)$$

This offers an elegant way to adopt a formal semantics view of natural language in vector spaces. Let nouns live in a basic vector space $N \in \mathbb{R}^D$; returning to our previous example, an adjective then can be seen as a map $f : N \rightarrow N$ which is isomorphic to $N \otimes N$ (that is, to a matrix). In general, the order of the tensor is equal to the number of arguments plus one dimension that carries the result; so a unary function (e.g. adjectives, intransitive verbs) is represented by a tensor of order 2 (a matrix), a binary function (e.g. a transitive verb) as an order 3 tensor, and so on. Due to the above isomorphism, function application (and hence our compositional operation) becomes a generalisation of matrix multiplication, formalised in terms of the inner product. In the case of a unary relational word, such as an adjective, this is nothing more than the usual notion

of matrix multiplication between a matrix and a vector. The generalization of this process to tensors of higher order is known as *tensor contraction*. Given two tensors of orders n and m, the tensor contraction operation will always produce a tensor of order $n + m - 2$.

Let us see an example of how this works for a simple transitive sentence. Let $\mathcal{V} \in \mathbb{R}^{I \times J \times K}$ be the tensor of order 3 for the verb and $\mathcal{S} \in \mathbb{R}^I$, $\mathcal{O} \in \mathbb{R}^K$ the tensors of order 1 (vectors) for the subject and the object of the verb, respectively. Then $\mathcal{V} \times \mathcal{O}$ will return a new tensor living in $\mathbb{R}^{I \times J}$ (i.e. a matrix)[1]; a further interaction of this result with the subject will return a vector for the whole transitive sentence living in $\mathbb{R}^J$. We should note that the order in which the verb is applied to its arguments is not important; so in general the meaning of a transitive sentence is given by:

$$(\mathcal{V} \times \mathcal{O})^{\mathsf{T}} \times \mathcal{S} = (\mathcal{V}^{\mathsf{T}} \times \mathcal{S}) \times \mathcal{O} \qquad (2)$$

where T denotes a transpose and makes indices match, since subject precedes the verb.

5 Creating verb tensors

In this section we review a number of proposals regarding concrete methods of constructing tensors for relational words in the context of the frameworks of Coecke et al. (2010) and Baroni and Zamparelli (2010), which both comply to the setting of Section 4.[2]

Relational Following ideas from the set-theoretic view of formal semantics, Grefenstette and Sadrzadeh (2011a) suggest that the meaning of a relational word should be represented as the sum of its arguments. The meaning of adjective 'red', for example, becomes the sum of the vectors of all the nouns that 'red' modifies in the corpus; so $\overrightarrow{red} = \sum_i \overrightarrow{noun_i}$, where i iterates through all the occurrences of 'red'. This can be generalised to relational words of any arity, by summing the tensor product of their arguments. So for a transitive verb we have:

$$\overline{verb}^2 = \sum_i (\overrightarrow{subj}_i \otimes \overrightarrow{obj}_i) \qquad (3)$$

where i again iterates over all occurrences of the specific verb in the corpus and the superscript denotes the order of the tensor.

In order to achieve a more expressive representation for the sentences, the authors used the convention that the arity of the head word in a sentence will also determine the order of the sentence space; that is, the space of intransitive sentences will be of order 1, of transitive ones will be of order 2, and so on. Recall from Section 4 that for the transitive case this increases the order of the verb tensor to 4 (2 dimensions for the arguments plus another 2 for the result). In spite of this, however, note that the method of Equation 3 produces a matrix. The other two dimensions of the tensor remain empty (filled with zeros), a fact that simplifies the calculations but also considerably weakens the expressive power of the model. This simplification transforms Equation 2 to the following:

$$\overline{subj\ verb\ obj}^2 = (\overrightarrow{subj} \otimes \overrightarrow{obj}) \odot \overline{verb}^2 \qquad (4)$$

where $\otimes$ denotes the tensor product and $\odot$ element-wise multiplication.

Kronecker In a subsequent work (Grefenstette and Sadrzadeh, 2011b), the same team proposes the creation of a verb matrix as the Kronecker product of the verb's contextual vector with itself:

$$\overline{verb}^2 = \overrightarrow{verb} \otimes \overrightarrow{verb} \qquad (5)$$

Again in this model the sentence space is of order 2, and the meaning of a transitive sentence is calculated using Equation 4.

Frobenius The previous models bring the important limitation that only sentences of the same structure can be meaningfully compared; it is not possible, for example, to compare an intransitive sentence (e.g. 'kids play') with a transitive one ('children play football'), since the former is a vector and the latter a matrix. Using Frobenius algebras, Kartsaklis et al. (2012) provide a unified sentence space for every sentence regardless of its type. These models turn the matrix of Equation 3 to a tensor of order 3 (as required by the type-logical identities) by copying one of the existing dimensions. When the dimension of rows (corresponding to subjects) is copied, the calculation of a vector for a transitive sentence becomes:

$$\overrightarrow{subj\ verb\ obj} = \overrightarrow{subj} \odot (\overline{verb}^2 \times \overrightarrow{obj}) \qquad (6)$$

[1] The symbol $\times$ denotes tensor contraction.

[2] In what follows we use the case of a transitive verb as an example; however the descriptions apply to any relational word of any arity. A vector (an order-1 tensor) is denoted as $\overrightarrow{x}$; tensors of order $n > 1$ are shown as $\overline{x}^n$ for clarity.

Copying the column dimension (objects) gives:

$$\overrightarrow{subj\ verb\ obj} = \overrightarrow{obj} \odot \left((\overrightarrow{verb}^2)^{\mathsf{T}} \times \overrightarrow{subj} \right) \quad (7)$$

Linear regression None of the above models create tensors that are fully populated: one or more dimensions will always remain empty. Following an idea first introduced by Baroni and Zamparelli (2010) for the creation of adjective matrices, Grefenstette et al. (2013) use linear regression in order to learn full tensors of order 3 for transitive verbs. Linear regression is a supervised method of learning, so it needs a number of exemplar data points. In the case of the adjective 'red', for example, we would need a set of the form $\langle \overrightarrow{car}, \overrightarrow{red\ car} \rangle, \langle \overrightarrow{shirt}, \overrightarrow{red\ shirt} \rangle, \langle \overrightarrow{shoe}, \overrightarrow{red\ shoe} \rangle$ and so on, where the second vector in each pair is the contextual vector of the whole phrase created exactly as if it were a single word. The goal of the learning process is to find the parameters $\overline{adj}^2$ and $\overrightarrow{b}$ such that:

$$\overrightarrow{adj\ noun} \approx \overline{adj}^2 \times \overrightarrow{noun} + \overrightarrow{b} \quad (8)$$

for all nouns modified by the specific adjective. In practice, the bias $\overrightarrow{b}$ is embedded in $\overline{adj}^2$, hence the above procedure provides us with a matrix for the adjective. One can generalize this procedure to tensors of higher order by proceeding step-wise, as done by Grefenstette et al. (2013). For the case of a transitive verb, they first use exemplar pairs of the form $\langle \overrightarrow{subj}, \overrightarrow{subj\ verb\ obj} \rangle$ to learn a matrix $\overline{verb\ obj}^2$ for the verb phrase; then, they perform a new training session with exemplars of the form $\langle \overrightarrow{obj}, \overline{verb\ obj}^2 \rangle$, the result of which is an order 3 tensor for the verb.

6 Generic context-based disambiguation

In all of the models of Section 5, the training of a relational word tensor is based on the set of contexts where this word occurs. Hence, in these models the problem of creating disambiguated versions of tensors can be recasted to that of further breaking the set of contexts in a way that each subset reflects a different sense of the word in the corpus. If, for example, S is the whole set of sentences for a word w that occurs in the corpus under n different senses, then the goal is to create n subsets $S_1, \ldots S_n$ such that S_1 contains the sentences where w appears under the first sense, S_2 the sentences where w occurs under the second sense, and so on. Each one of these subsets can then be used to train a tensor for a specific sense of the target relational word.

Towards this purpose we use a variation of the effective procedure of Schütze (1998): first, each context for a target word w_t is represented by a *context vector* of the form $\frac{1}{n}(\overrightarrow{w_1} + \ldots + \overrightarrow{w_n})$, where $\overrightarrow{w_i}$ is the lexical vector of some other word $w_i \neq w_t$ in the same context. Next, we apply a clustering method on this set of vectors in order to discover the latent senses of w_t. The assumption is that the contexts of w_t will vary according to the specific sense this word is used: 'bank' as a financial institution should appear in quite different contexts than as land.

The above procedure will give us a number of clusters, each consisting of context vectors; we use the centroid of each cluster as a vectorial representation of the corresponding sense. So in our model each word w is initially represented by a tuple $\langle \overrightarrow{w}, S \rangle$, where $\overrightarrow{w}$ is the lexical vector of the word as created by the usual distributional practice, and S is a set of sense vectors (centroids of context vector clusters) produced by the above procedure. The disambiguation of a new word w under a context C can now be accomplished as follows: we create a context vector $\overrightarrow{c}$ for C as above, and we compare it with every sense vector of w; the word is assigned to the sense corresponding to the closest sense vector. Specifically, if S_w is the set of sense vectors for w, $\overrightarrow{c}$ the context vector for C, and $d(\overrightarrow{v}, \overrightarrow{u})$ our vector distance measure, the preferred sense $\hat{s}$ is given by:

$$\hat{s} = \underset{\overrightarrow{v_s} \in S_w}{\arg\min}\ d(\overrightarrow{v_s}, \overrightarrow{c}) \quad (9)$$

For the actual clustering step we follow the setting of Kartsaklis et al. (2013), which worked well in tasks very similar to ours. Specifically, we perform hierarchical agglomerative clustering (HAC) using Ward's method as the inter-cluster distance, while the distance between vectors is measured with Pearson's correlation.[3] In the above work, this configuration has been found to return the highest V-measure (Rosenberg and Hirschberg, 2007) on the noun set of SEMEVAL 2010 Word Sense Induction & Disambiguation Task (Manandhar et al., 2010). As context for a word, we consider the sentence in which this word occurs. The output of HAC is a *dendrogram* embedding all the possible partitionings of the

[3] Informal experimentation with more robust probabilistic techniques, such as Dirichlet process gaussian mixture models, revealed no significant benefits for our setting.

data. In order to select the optimal partitioning, we rely on the Caliński/Harabasz index (Caliński and Harabasz, 1974), also known as variance ratio criterion (VRC). VRC is calculated as the ratio of the sum of the inter-cluster variances over the sum of the intra-cluster variances, bearing the intuition that the optimal partitioning should be the one that results in the most compact and maximally separated clusters. We compute the VRC for a range of different partitionings (from 2 to 10 clusters) and keep the partitioning with the highest score.

7 Constructing unambiguous verb tensors

The procedure described in Section 6 provides us with n clusters of context vectors for a target word. Since in our case each context vector corresponds to a distinct sentence, the output of the clustering scheme can also be seen as n subsets of sentences, where the word appears under different senses. It is now quite straightforward to use this partitioning of the training corpus in order to learn unambiguous versions of verb tensors, as detailed below.

Relational/Frobenius Both the Relational and the Frobenius models use the same way of creating an initial verb matrix (Equation 3) which then they expand to a higher order tensor. Let $S_1 \dots S_n$ be the sets of sentences returned by the clustering step for a verb. Then, the verb tensor for the ith sense is:

$$\overline{verb}_i^2 = \sum_{s \in S_i} (\overrightarrow{subj_s} \otimes \overrightarrow{obj_s}) \tag{10}$$

where $subj_s$ and obj_s refer to the subject/object pair that occurred with the verb in sentence s. This can be generalized to any arity n as follows:

$$\overline{word}_i^n = \sum_{s \in S_i} \bigotimes_{k=1}^{n} \overrightarrow{arg_{k,s}} \tag{11}$$

where $arg_{k,s}$ denotes the kth argument of the target word in sentence s.

Kronecker For a given verb v in a context C, let $\overrightarrow{v_i}$ be the sense vector of v given C corresponding to the sense i returned by Equation 9. Then we have:

$$\overline{verb}_i^2 = \overrightarrow{v_i} \otimes \overrightarrow{v_i} \tag{12}$$

The generalized version to arity n is given by:

$$\overline{word}_i^n = \bigotimes_{k=1}^{n} \overrightarrow{v_i} \tag{13}$$

Linear regression Creating unambiguous full tensors using linear regression is also quite straightforward. Let us assume again that the clustering step for a verb v returns n sets of sentences $S_1 \dots S_n$, where each sentence set corresponds to a different sense. Then, we have n different regression problems, each one of which will be trained on exemplar pairs derived exclusively from the sentences of the corresponding set. This will result in n verb tensors, which will correspond to the different senses of the verb. Generalization to higher arities is a straightforward extension of the step-wise process in Section 5 for transitive verbs.

8 Experiments

In this section we will test the effect of disambiguation on the models of Section 5 in a variety of tasks. Due to the significant methodological differences of the linear regression model from the other approaches and the variety of its set of parameters, we decided that it would be better if this was left as the subject of a distinct work.

Experimental setting We train our vectors using ukWaC (Ferraresi et al., 2008), a corpus of English text with 2 billion words (100m sentences). We use 2000 dimensions, with weights calculated as the ratio of the probability of the context word given the target word to the probability of the context word overall. The context here is a 5-word window on both sides of the target word. The vectors are disambiguated both syntactically and semantically: first, separate vectors have been created for different syntactic usages of the same word in the corpus; for example, the word 'book' has two vectors, one for its noun sense and one for its verb sense. Furthermore, each word is semantically disambiguated according to the method of Section 6.

Models We compare the tensor-based models of Section 5 with the multiplicative and additive models of Mitchell and Lapata (2008), reporting results for both ambiguous and disambiguated versions. For all the disambiguated models, the best sense for each word in the sentence or phrase is first selected by applying the procedure of Section 6 and Equation 9. If the model is based on a vector mixture, the sense vectors corresponding to these senses are multiplied or added to form the composite representation for the sentence or phrase. For the tensor-based models, the composite meanings are calculated ac-

cording to the equations of Section 5, using verb tensors created by the procedures of Section 7. The semantic similarity of two phrases or sentences is measured as the cosine distance between their composite vectors. For models that return a matrix (e.g. Relational, Kronecker), the distance is based on the Frobenius inner product.

Implementation details Our code is mainly written in Python and C++, and for the actual clustering step we use the Python interface of the efficient FASTCLUSTER library (Müllner, 2013). In a shared 24-core Xeon machine with 72 GB of memory, and with a fair amount of parallelism applied, the average processing time per word was about 4 minutes; this is roughly translated to 12-13 hours of training on average per dataset.

8.1 Verb disambiguation task

Perhaps surprisingly, one of the most popular tasks for testing compositionality in distributional models is based on disambiguation. This task, originally introduced by Kintsch (2001), has been adopted by Mitchell and Lapata (2008) and others for evaluating the quality of composition in vector spaces. Given an ambiguous verb such as 'file', the goal is to find out to what extent the presence of an appropriate context will disambiguate its intended meaning. The context (e.g. a subject/object pair) is composed with two landmark verbs corresponding to the different senses ('smooth' and 'register') to create simple sentences. The assumption is that a good compositional model should be able to reflect that 'woman files application' is closer to 'woman registers application' than to 'woman smooths application'.

In this paper we test our models on two different datasets of transitive sentences, that of Grefenstette and Sadrzadeh (2011a) and Kartsaklis et al. (2013)[4]. Specific details about the creation of the datasets can be found in the above papers; for the purposes of the current work it is sufficient to mention that their main difference is that in the former the verbs and their alternative meanings have been selected automatically using the JCN metric of semantic similarity (Jiang and Conrath, 1997), while in the latter the selection was based on human judgements from the work of Pickering and Frisson (2001). So, while in the first dataset many verbs cannot be considered as genuinely ambiguous (e.g. 'say' with meanings *state* and *allege* or 'write' with meanings *publish* and *spell*), the landmarks in the second dataset correspond to clearly separated senses (e.g. 'file' with meanings *register* and *smooth* or 'charge' with meanings *accuse* and *bill*). Furthermore, subjects and objects of this latter case are modified by appropriate adjectives, overall creating a richer and more linguistically balanced dataset.

In both cases the evaluation methodology is the same: each entry of the dataset has the form ⟨subject, verb, object, high-sim landmark, low-sim landmark⟩. The context is combined with the verb and the two landmarks, creating three simple transitive sentences. The main-verb sentence is paired with both the landmark sentences, and these pairs are randomly presented to human evaluators, the duty of which is to evaluate the similarity of the sentences within a pair in a scale from 1 to 7. The scores of the compositional models are the cosine distances (or the Frobenius inner products, in the case of matrices) between the composite representations of the sentences of each pair. As an overall score for each model, we report its Spearman's ρ correlation with the human judgements. Both datasets consist of 200 pairs of sentences (10 main verbs $\times$ 2 landmarks $\times$ 10 contexts).

Results The results for the G&S dataset are shown in Table 1.[5] The verbs-only model (BL) refers to a non-compositional evaluation, where the similarity between two sentences is solely based on the distance between the two verbs, without applying any compositional step with subject and object.

The tensor-based models present much better performance than the vector mixture ones, with the disambiguated version of the copy-object model significantly higher than the relational model. By design, the copy-object model retains more information about the objects; so this result confirms previous findings, that in this certain dataset the role of objects is more important than that of subjects (Kartsaklis et al., 2012). In general, the disambiguation step improves the results of all the tensor-based models except Kronecker; the effect is reversed for the vector mixture models, where the disambiguated versions present much worse performance (these

[4]This dataset has been created by Mehrnoosh Sadrzadeh in collaboration with Edward Grefenstette, but remained unpublished until (Kartsaklis et al., 2013).

[5]For all tables in this section, $\ll$ and $\gg$ denote highly statistically significant differences with $p < 0.001$.

	Model	Ambig.		Disamb.
BL	Verbs only	0.198	$\gg$	0.132
M1	Multiplicative	0.137	$\gg$	0.044
M2	Additive	0.127	$\gg$	0.047
T1	Relational	0.219	$<$	0.223
T2	Kronecker	0.207	$\gg$	0.061
T3	Copy-subject	0.070	$\ll$	0.122
T4	**Copy-object**	**0.241**	$\ll$	**0.262**
	Human agreement	0.599		

Difference between T4 and T1 is s.s. with $p < 0.001$

Table 1: Results for the G&S dataset.

	Model	Ambig.		Disamb.
BL	**Verbs only**	**0.151**	$\ll$	**0.217**
M1	Multiplicative	0.131	$<$	0.137
M2	Additive	0.085	$\ll$	0.193
T1	Relational	0.036	$\ll$	0.121
T2	Kronecker	0.159	$<$	0.166
T3	Copy-subject	0.035	$\ll$	0.117
T4	Copy-object	0.033	$\ll$	0.095
	Human agreement	0.383		

Difference between BL and M2 is s.s. with $p < 0.001$

Table 2: Results for the Kartsaklis et al. dataset.

findings are further discussed in Section 9).

The result of disambiguation is clearer for the dataset of Kartsaklis et al. (Table 2). The longer context in combination with genuinely ambiguous verbs produces two effects: first, disambiguation is now helpful for all models, either vector mixtures or tensor-based; second, the disambiguation of just the verb (verbs-only model), without any interaction with the context, is sufficient to provide the best score (0.22) with a difference statistically significant from the second model (0.19 for disambiguated additive). In fact, further composition of the verb with the context decreases performance, confirming the results reported by Kartsaklis et al. (2013) for vectors trained using BNC. Given the nature of the specific task, which is designed around the ambiguity of the verb, this result is not surprising: a direct disambiguation of the verb based on the rest of the context should naturally constitute the best method to achieve top performance—no composition is necessary for this task to be successful.

However, when one *does* use a task like this in order to evaluate compositional models (as we do here and as is commonly the case), they implicitly correlate the strength of the disambiguation effect that takes place during the composition with the quality of composition, essentially assuming that the stronger the disambiguation, the better the compositional model that produced this side-effect. Unfortunately, the extent to which this assumption is valid or not is still not quite clear; the subject is addressed in more detail in (Kartsaklis et al., 2013). Keeping a note of this observation, we now proceed to examine the performance of our models in a task that does not use disambiguation as a criterion of composition.

8.2 Phrase/sentence similarity task

Our second set of experiments is based on the phrase similarity task of Mitchell and Lapata (2010). On the contrary with the task of Section 8.1, this one does not involve any assumptions about disambiguation, and thus it seems like a more genuine test of models aiming to provide appropriate phrasal or sentential semantic representations; the only criterion is the degree to which these models correctly evaluate the *similarity* between pairs of sentences or phrases. We work on the verb-phrase part of the dataset, consisting of 72 short verb phrases (verb-object structures). These 72 phrases have been paired in three different ways to form groups exhibiting various degrees of similarity: the first group contains 36 pairs of highly similar phrases (e.g. *produce effect-achieve result*), the pairs of the second group are of medium similarity (e.g. *write book-hear word*), while a last group contains low-similarity pairs (*use knowledge-provide system*). The task is again to compare the similarity scores given by the various models for each phrase pair with those of human annotators. Additionally to the verb phrases task, we also perform a richer version of the experiment using transitive sentences.

Verb phrases It can be shown that for simple verb phrases the relational model reduces itself to the copy-subject model; for both of these methods, the meaning of the verb phrase is calculated according to Equation 6. Furthermore, according to the copy-object model the meaning of a verb phrase computed by a verb matrix $\sum_{ij} v_{ij}(\overrightarrow{n_i} \otimes \overrightarrow{n_j})$ and an object vector $\sum_j o_j \overrightarrow{n_j}$ becomes:

$$\overrightarrow{verb\ object}^2 = \sum_{ij} v_{ij} o_j (\overrightarrow{n_i} \otimes \overrightarrow{n_j}) \qquad (14)$$

Finally, the Kronecker model has no meaning for verb phrases, since the vector of a verb phrase will become $(\overrightarrow{v_s} \otimes \overrightarrow{v_s}) \times \overrightarrow{obj}$, which is equal to $\langle \overrightarrow{v_s} | \overrightarrow{obj} \rangle \overrightarrow{v_s}$, where $\langle \overrightarrow{v_s} | \overrightarrow{obj} \rangle$ denotes the inner product between vectors of verb and object. Hence, the

	Model	Ambig.		Disamb.
BL	Verbs only	0.310	$\ll$	0.420
M1	**Multiplicative**	**0.315**	$\ll$	**0.448**
M2	Additive	0.291	$\ll$	0.436
T1	Rel./Copy-sbj	0.340	$\ll$	0.367
T2	Copy-object	0.290	$\ll$	0.393
	Human agreement	0.550		

Difference between M1 and M2 is not s.s.
Difference between M1 and BL is s.s. with $p < 0.001$

Table 3: Results for the original M&L task.

meaning of a verb phrase becomes a scalar multiplication of the meaning of its verb. As a result, the cosine distance (used for measuring similarity) between the meanings of two verb phrases is reduced to the distance between the vectors of their verbs, completely dismissing the role of their objects.

Hence our models are limited to those of Table 3. The effects of disambiguation for this task are quite impressive: the differences between the scores of all disambiguated models and those of the ambiguous versions are highly statistically significant (with $p < 0.001$), while 4 of the 5 models present an improvement greater than 10 units of correlation. The models that benefit the most from disambiguation are the vector mixtures; both of these approaches perform significantly better than the best tensor-based model (copy-object). In fact, the score of M1 (0.45) is quite high, given that the inter-annotator agreement is 0.55 (best score reported by Mitchell and Lapata was 0.41 for their LDA-dilation model).

Transitive sentences The second part of this experiment aims to examine the extent to which the above picture can change for the case of text structures longer than verb phrases. In order to achieve this, we extend each one of the 72 verb phrases to a full transitive sentence by adding an appropriate subject such that the similarity relationships of the original dataset are retained as much as possible, so the human judgements for the verb phrase pairs could as well be used for the transitive cases. We worked pair-wise: for each pair of verb phrases, we first selected one of the 5 most frequent subjects for the first phrase; then, the subject of the other phrase was selected by a list of synonyms of the first subject in a way that the new pair of transitive sentences constitutes the least more specific version of the given verb-phrase pair. So, for example, the pair *produce effect/achieve result* became *drug produce effect/medication achieve result*, while the pair *pose problem/address question* became *study pose problem/paper address question*.[6]

The restrictions of the verb-phrase version do not hold here, so we evaluate on the full set of models (Table 4). Once more disambiguation produces better results in all cases, with highly statistically significant differences for all but one model. Furthermore, now the best score is delivered by one of the tensor-based models (Kronecker), with a difference *not* statistically significant from disambiguated additive. In any case, the result suggests that as the length of the text segments increases, the performance of vector mixtures and tensor-based models converges. Indeed, note how the performance of the vector mixture models are significantly decreased compared to the verb phrase task.

9 Discussion

The purpose of this work was twofold: our main objective was to investigate how disambiguation can affect the compositional models which are based on higher order vector spaces; a second, but not less important goal, was to compare this more linguistically motivated approach to the simpler vector mixture methods. Based on the experimental work presented here, we can say with enough confidence that disambiguation as an additional step prior to composition is indeed very beneficial for tensor-based models. Furthermore, our experiments confirm and strengthen previous work (Reddy et al., 2011; Kartsaklis et al., 2013) that showed better performance of disambiguated vector mixture models compared to their ambiguous versions. The positive effect of disambiguation is more evident for the vector mixture models (especially for the additive model) than for

[6]The dataset will be available at `http://www.cs.ox.ac.uk/activities/compdistmeaning/`.

	Model	Ambig.		Disamb.
BL	Verbs only	0.310	$\ll$	0.341
M1	Multiplicative	0.325	$\ll$	0.404
M2	Additive	0.368	$\ll$	0.410
T1	Relational	0.368	$\ll$	0.397
T2	**Kronecker**	**0.404**	$<$	**0.412**
T3	Copy-subject	0.310	$\ll$	0.337
T4	Copy-object	0.321	$\ll$	0.368
	Human agreement	0.550		

Difference between T2 and M2 is not s.s.

Table 4: Transitive version of M&L task.

the tensor-based ones. This is expected: composite representations created by element-wise operations are averages, and a prior step of disambiguation can make a great difference.

From a task perspective, the effect of disambiguation was much more definite in the phrase/sentence similarity task. This observation is really interesting, since the words of that dataset were *not* selected in order to be ambiguous in any way. The superior performance of the disambiguated models, therefore, implies that the proposed methodology can improve tasks based on phrase or sentence similarity *regardless* of the level of ambiguity in the vocabulary. For these cases, the proposed disambiguation algorithm acts as a fine-tuning process, the outcome of which seems to be always positive; it can only produce better composite representations, not worse. In general, the positive effect of disambiguation in the phrase/sentence similarity task is quite encouraging, especially given the fact that this task constitutes a more appropriate test for evaluating compositional models, avoiding the pitfalls of disambiguation-based experiments (as shortly discussed in Section 8.1).

For disambiguation-based tasks similar to those of Section 8.1, the form of dataset is very important; hence the inferior performance of disambiguated models in the G&S dataset, compared to the dataset of Kartsaklis et al.. In fact, the G&S dataset was the only one where disambiguation was not helpful for some cases (specifically, for vector mixtures and the Kronecker model). We believe the reason behind this lies in the fact that the automatic selection of landmark verbs using the JCN metric (as done with the G&S dataset) was not very efficient for certain cases. Note, for example, that the bare baseline of comparing just ambiguous versions of verbs (without any composition) in that dataset already achieves a very high correlation of 0.198 with human judgements (Table 1).[7]. This number is only 0.15 for the Kartsaklis et al. dataset, due to the more efficient verb selection procedure. In general, we consider the results gained by this latter experiment more reliable for the specific task, the successful evaluation of which requires genuinely ambiguous verbs.

The results are less conclusive for the second question we posed in the beginning of this section, regarding the comparison of the two classes of models. Despite the obvious benefits of the tensor-based approaches, this work suggests for one more time that vector mixture models might constitute a hard-to-beat baseline; similar observations have been made, for example, in the comparative study of Blacoe and Lapata (2012). However, when trying to interpret the mixing results regarding the effectiveness of the tensor-based models compared to vector mixtures, we need to take into account that the tensor-based models tested in this work were all "hybrid", in the sense that they all involved some element of point-wise operation; in other words, they constituted a trade-off between transformational power and complexity.

Even with this compromise, though, the study presented in Section 8.2 implies that the effectiveness of each method depends to some extent on the length of the text segment: when more words are involved, vector mixture models tend to be less effective; on the contrary, the performance of tensor-based models seems to be proportional to the length of the phrase or sentence—the more, the better. These observations comply with the nature of the approaches: "averaging" larger numbers of points results in more general (hence less accurate) representations; on the other hand, a larger number of arguments makes a function (such as a verb) more accurate.

10 Conclusion and future work

In the present paper we showed how to improve a number of tensor-based compositional distributional models of meaning by introducing a step of disambiguation prior to composition. Our simple algorithm (based on the procedure of Schütze (1998)) creates unambiguous versions of tensors before these are composed with vectors of nouns in order to construct vectors for sentences and phrases. This algorithm is quite generic, and can be applied to any model that follows the tensor contraction process described in Section 4. As for future work, we aim to investigate the application of this procedure to the regression model of Grefenstette et al. (2013).

Acknowledgements

We would like to thank Edward Grefenstette for his comments on the first draft of this paper, as well as the three anonymous reviewers for their fruitful suggestions. Support by EPSRC grant EP/F042728/1 is gratefully acknowledged by the authors.

[7] The reported number for this baseline by Grefenstette and Sadrzadeh (2011a) was 0.16 using vectors trained from BNC.

References

Baroni, M. and Zamparelli, R. (2010). Nouns are Vectors, Adjectives are Matrices. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Blacoe, W. and Lapata, M. (2012). A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556, Jeju Island, Korea. Association for Computational Linguistics.

Bourbaki, N. (1989). *Commutative Algebra: Chapters 1-7*. Srpinger Verlag, Berlin/New York.

Caliński, T. and Harabasz, J. (1974). A Dendrite Method for Cluster Analysis. *Communications in Statistics-Theory and Methods*, 3(1):1–27.

Coecke, B., Sadrzadeh, M., and Clark, S. (2010). Mathematical Foundations for Distributed Compositional Model of Meaning. Lambek Festschrift. *Linguistic Analysis*, 36:345–384.

Curran, J. (2004). *From Distributional to Semantic Similarity*. PhD thesis, School of Informatics, University of Edinburgh.

Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. (2008). Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Grefenstette, E., Dinu, G., Zhang, Y.-Z., Sadrzadeh, M., and Baroni, M. (2013). Multi-step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*.

Grefenstette, E. and Sadrzadeh, M. (2011a). Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Grefenstette, E. and Sadrzadeh, M. (2011b). Experimenting with Transitive Verbs in a DisCoCat. In *Proceedings of Workshop on Geometrical Models of Natural Language Semantics (GEMS)*.

Harris, Z. (1968). *Mathematical Structures of Language*. Wiley.

Jiang, J. and Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*, pages 19–33, Taiwan.

Kartsaklis, D., Sadrzadeh, M., and Pulman, S. (2012). A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of 24th International Conference on Computational Linguistics (COLING 2012): Posters*, pages 549–558, Mumbai, India. The COLING 2012 Organizing Committee.

Kartsaklis, D., Sadrzadeh, M., and Pulman, S. (2013). Separating Disambiguation from Composition in Distributional Semantics. In *Proceedings of 17th Conference on Computational Natural Language Learning (CoNLL-2013)*, Sofia, Bulgaria.

Kintsch, W. (2001). Predication. *Cognitive Science*, 25(2):173–202.

Landauer, T. and Dumais, S. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquision, Induction, and Representation of Knowledge. *Psychological Review*.

Manandhar, S., Klapaftis, I., Dligach, D., and Pradhan, S. (2010). Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68. Association for Computational Linguistics.

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mitchell, J. and Lapata, M. (2008). Vector-based Models of Semantic Composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 236–244.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Müllner, D. (2013). fastcluster: Fast Hierarchical Clustering Routines for R and Python. *Journal of Statistical Software*, 9(53):1–18.

Pickering, M. and Frisson, S. (2001). Processing ambiguous verbs: Evidence from eye movements. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(2):556.

Pulman, S. (2013). Combining Compositional and Distributional Models of Semantics. In Heunen, C., Sadrzadeh, M., and Grefenstette, E., editors, *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*. Oxford University Press.

Reddy, S., Klapaftis, I., McCarthy, D., and Manandhar, S. (2011). Dynamic and static prototype vectors for semantic composition. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 705–713.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 410–420.

Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*, 24:97–123.

Multi-Relational Latent Semantic Analysis

Kai-Wei Chang*
University of Illinois
Urbana, IL 61801, USA
kchang10@illinois.edu

Wen-tau Yih **Christopher Meek**
Microsoft Research
Redmond, WA 98052, USA
{scottyih,meek}@microsoft.com

Abstract

We present Multi-Relational Latent Semantic Analysis (MRLSA) which generalizes Latent Semantic Analysis (LSA). MRLSA provides an elegant approach to combining multiple relations between words by constructing a 3-way tensor. Similar to LSA, a low-rank approximation of the tensor is derived using a tensor decomposition. Each word in the vocabulary is thus represented by a vector in the latent semantic space and each relation is captured by a latent square matrix. The degree of two words having a specific relation can then be measured through simple linear algebraic operations. We demonstrate that by integrating multiple relations from both homogeneous and heterogeneous information sources, MRLSA achieves state-of-the-art performance on existing benchmark datasets for two relations, *antonymy* and *is-a*.

1 Introduction

Continuous semantic space representations have proven successful in a wide variety of NLP and IR applications, such as document clustering (Xu et al., 2003) and cross-lingual document retrieval (Dumais et al., 1997; Platt et al., 2010) at the document level and sentential semantics (Guo and Diab, 2012; Guo and Diab, 2013) and syntactic parsing (Socher et al., 2013) at the sentence level. Such representations also play an important role in applications for lexical semantics, such as word sense disambiguation (Boyd-Graber et al., 2007), measuring word similarity (Deerwester et al., 1990) and relational similarity (Turney, 2006; Zhila et al., 2013; Mikolov et al., 2013). In many of these applications, Latent Semantic Analysis (LSA) (Deerwester et al., 1990) has been widely used, serving as a fundamental component or as a strong baseline.

LSA operates by mapping text objects, typically documents and words, to a latent semantic space. The proximity of the vectors in this space implies that the original text objects are semantically related. However, one well-known limitation of LSA is that it is unable to differentiate fine-grained relations. For instance, when applied to lexical semantics, synonyms and antonyms may both be assigned high similarity scores (Landauer and Laham, 1998; Landauer, 2002). Asymmetric relations like hyponyms and hypernyms also cannot be differentiated. Although there exists some recent work, such as PILSA which tries to overcome this weakness of LSA by introducing the notion of polarity (Yih et al., 2012). This extension, however, can only handle two opposing relations (e.g., synonyms and antonyms), leaving open the challenge of encoding multiple relations.

In this paper, we propose Multi-Relational Latent Semantic Analysis (MRLSA), which strictly generalizes LSA to incorporate information of multiple relations concurrently. Similar to LSA or PILSA when applied to lexical semantics, each word is still mapped to a vector in the latent space. However, when measuring whether two words have a specific relation (e.g., antonymy or *is-a*), the word vectors will be mapped to a new space according to the relation where the degree of having this relation will be

*Work conducted while interning at Microsoft Research.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1602–1612,
Seattle, Washington, USA, 18-21 October 2013.

judged by cosine similarity. The raw data construction in MRLSA is straightforward and similar to the document-term matrix in LSA. However, instead of using one matrix to capture all relations, we extend the representation to a 3-way tensor. Each slice corresponds to the document-term matrix in the original LSA design but for a specific relation. Analogous to LSA, the whole linear transformation mapping is derived through tensor decomposition, which provides a low-rank approximation of the original tensor. As a result, previously unseen relations between two words can be discovered, and the information encoded in other relations can influence the construction of the latent representations, and thus potentially improves the overall quality. In addition, the information in different slices can come from heterogeneous sources (conceptually similar to (Riedel et al., 2013)), which not only improves the model, but also extends the word coverage in a reliable way.

We provide empirical evidence that MRLSA is effective using two different word relations: *antonymy* and *is-a*. We use the benchmark GRE test of closest-opposites (Mohammad et al., 2008) to show that MRLSA performs comparably to PILSA, which was the pervious state-of-the-art approach on this problem, when given the same amount of information. In addition, when other words and relations are available, potentially from additional resources, MRLSA is able to outperform previous methods significantly. We use the *is-a* relation to demonstrate that MRLSA is capable of handling asymmetric relations. We take the list of word pairs from the *Class-Inclusion* (i.e., *is-a*) relations in SemEval-2012 Task 2 (Jurgens et al., 2012), and use our model to measure the degree of two words have this relation. The measures derived from our model correlate with human judgement better than the best system that participated in the task.

The rest of this paper is organized as follows. We first survey some related work in Section 2, followed by a more detailed description of LSA and PILSA in Section 3. Our proposed model, MRLSA, is presented in Section 4. Section 5 presents our experimental results. Finally, Section 6 concludes the paper.

2 Related Work

MRLSA can be viewed as a model that derives general continuous space *representations* for capturing *lexical semantics*, with the help of *tensor decomposition* techniques. We highlight some recent work related to our approach.

The most commonly used continuous space representation of text is arguably the vector space model (VSM) (Turney and Pantel, 2010). In this representation, each text object can be represented by a high-dimensional sparse vector, such as a term-vector or a document-vector that denotes the statistics of term occurrences (Salton et al., 1975) in a large corpus. The text can also be represented by a low-dimensional dense vector derived by linear projection models like latent semantic analysis (LSA) (Deerwester et al., 1990), by discriminative learning methods like Siamese neural networks (Yih et al., 2011), recurrent neural networks (Mikolov et al., 2013) and recursive neural networks (Socher et al., 2011), or by graphical models such as probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) and latent Dirichlet allocation (LDA) (Blei et al., 2003). As a generalization of LSA, MRLSA is also a linear projection model. However, while the words are represented by vectors as well, multiple relations between words are captured separately by matrices.

In the context of lexical semantics, VSMs provide a natural way of measuring semantic word relatedness by computing the distance between the corresponding vectors, which has been a standard approach (Agirre et al., 2009; Reisinger and Mooney, 2010; Yih and Qazvinian, 2012). These approaches do not apply directly to the problem of modeling other types of relations. Existing methods that do handle multiple relations often use a model combination scheme to integrate signals from various types of information sources. For instance, morphological variations discovered from the Google n-gram corpus have been combined with information from thesauri and vector-based word relatedness models for detecting antonyms (Mohammad et al., 2008). An alternative approach proposed by Turney (2008) that handles synonyms, antonyms and associations is to use a uniform approach by first reducing the problem to determining whether two

pairs of words can be analogous, and then predicting it using a supervised model with features based on the frequencies of patterns in the corpus. Similarly, to measure whether two word pairs have the same relation, Zhila et al. (2013) proposed to combine heterogeneous models, which achieved state-of-the-art performance. In comparison, MRLSA models multiple lexical relations holistically. The degree that two words having a particular relation is estimated using the same linear function of the corresponding vectors and matrix.

Tensor decomposition generalizes matrix factorization and has been applied to several NLP applications recently. For example, Cohen et al. (2013) proposed an approximation algorithm for PCFG parsing that relies on Kruskal decomposition. Van de Cruys et al. (2013) modeled the composition of subject-verb-object triples using Tucker decomposition, which results in a better similarity measure for transitive phrases. Similar to this construction but used in the community-based question answering (CQA) scenario, Qiu et al. (2013) represented triples of question title, question content and answer as a tensor and applied 3-mode SVD to derive latent semantic representations for question matching. The construction of MRLSA bears some resemblance to the work that use tensors to capture triples. However, our goal of modeling different relations for lexical semantics is very different from the intended usage of tensor decomposition in the existing work.

3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is a widely used continuous vector space model that maps words and documents into a low dimensional space. LSA consists of two main steps. First, taking a collection of d documents that contains words from a vocabulary list of size n, it first constructs a $d \times n$ document-term matrix $\mathbf{W}$ to encode the occurrence information of a word in a document. For instance, in its simplest form, the element $\mathbf{W}_{i,j}$ can be the term frequency of the j-th word in the i-th document. In practice, a weighting scheme that better captures the importance of a word in the document, such as TF×IDF (Salton et al., 1975), is often used instead. Notice that "document" here simply means a group of words and has been applied to various texts including news articles, sentences and bags of words. Once the matrix is constructed, the second step is to apply singular value decomposition (SVD) to $\mathbf{W}$ in order to derive a low-rank approximation. To have a rank-k approximation, $\mathbf{X}$ is the reconstruction matrix of $\mathbf{W}$, defined as

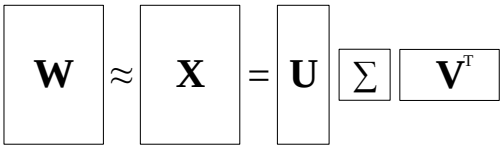


Figure 1: SVD applied to a $d \times n$ document-term matrix $\mathbf{W}$. The rank-k approximation, $\mathbf{X}$, is the multiplication of $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are $d \times k$ and $n \times k$ orthonormal matrices and $\mathbf{\Sigma}$ is a $k \times k$ diagonal matrix. The column vectors of $\mathbf{V}^T$ multiplied by the singular values $\mathbf{\Sigma}$ represent words in the latent semantic space.

$$\mathbf{W} \approx \mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (1)$$

where the dimensions of $\mathbf{U}$ and $\mathbf{V}$ are $d \times k$ and $n \times k$, respectively, and $\mathbf{\Sigma}$ is a $k \times k$ diagonal matrix. In addition, the columns in $\mathbf{U}$ and $\mathbf{V}$ are orthonormal and the elements in $\mathbf{\Sigma}$ are the singular values and are conventionally reverse-ordered. Figure 1 illustrates this decomposition.

LSA can be used to compute the similarity between two documents or two words in the latent space. For instance, to compare the u-th and v-th words in the vocabulary, one can compute the cosine similarity of the u-th and v-th column vectors of $\mathbf{X}$, the reconstruction matrix of $\mathbf{W}$. In contrast to a direct lexical matching via the columns of $\mathbf{W}$, the similarity measure computed as a result of the SVD may have a nonzero similarity score even if these two words do not co-occur in any documents. This is due to the fact that those words can share some latent components.

An alternative view of using LSA is to treat the column vectors of $\mathbf{\Sigma}\mathbf{V}^T$ as a representation of the words in a new k-dimensional latent space. This comes from the observation that the inner product of every two column vectors in $\mathbf{X}$ is the inner product of the corresponding column vectors of $\mathbf{\Sigma}\mathbf{V}^T$,

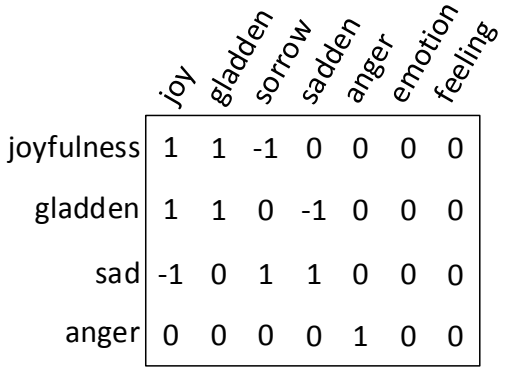


Figure 2: The matrix construction of PILSA. The vocabulary is {joy, gladden, sorrow, sadden, anger, emotion, feeling} and target words are {joyfulness, gladden, sad, anger}. For ease of presentation, we show the numbers with 0-1 values instead of TF×IDF scores. The polarity (i.e., sign) indicates whether the term in the vocabulary is a synonym or antonym of the target word.

which can be derived from the equations below.

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) \\ &= \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (\mathbf{\Sigma} \text{ is diagonal}) \\ &= \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \quad (\text{Columns of } \mathbf{U} \text{ are orthonormal}) \\ &= (\mathbf{\Sigma}\mathbf{V}^T)^T(\mathbf{\Sigma}\mathbf{V}^T) \end{aligned} \tag{2}$$

Thus, the semantic relatedness between the i-th and j-th words can be computed by cosine similarity[1]:

$$\cos(\mathbf{X}_{:,i}, \mathbf{X}_{:,j}) \tag{3}$$

When used to compare words, one well-known limitation of LSA is that the score captures the general notion of semantic similarity, and is unable to distinguish fine-grained word relations, such as antonyms (Landauer and Laham, 1998; Landauer, 2002). This is due to the fact that the raw matrix representation only records the occurrences of words in documents without knowing the specific relation between the word and document. To address this issue, Yih et al. (2012) proposed a polarity inducing latent semantic analysis model recently, which we introduce next.

[1]Cosine similarity is equivalent to the inner product of the normalized vectors.

3.1 Polarity Inducing Latent Semantic Analysis

In order to distinguish antonyms from synonyms, the polarity inducing LSA (PILSA) model (Yih et al., 2012) takes a thesaurus as input. Synonyms and antonyms of the same target word are grouped together as a "document" and a document-term matrix is constructed accordingly as done in LSA. Because each word in a group belongs to either one of the two opposite relations, *synonymy* and *antonymy*, the *polarity* information is induced by flipping the signs of antonyms. While the absolute value of each element in the matrix is still the same TF×IDF score, the elements that correspond to the antonyms become negative.

This design has an intriguing effect. When comparing two words using the cosine similarity (or simply inner product) of their corresponding column vectors in the matrix, the score of a synonym pair remains positive, but the score of an antonym pair becomes negative. Figure 2 illustrates this design using a simplified matrix as example.

Once the matrix is constructed, PILSA applies SVD as done in LSA, which generalizes the model to go beyond lexical matching. The sign of the cosine score of the column vectors of any two words indicates whether they are close to synonyms or to antonyms and the absolute value reflects the degree of the relation. When all the column vectors are normalized to unit vectors, it can also be viewed as synonyms are clustered together and antonyms lie on the opposite sides of a unit sphere. Although PILSA successfully extends LSA to handle not just one single *occurrence* relation, the extension is limited to encoding two opposing relations

4 Multi-Relational Latent Semantic Analysis

The fundamental reason why it is difficult to handle multiple relations is due to the 2-dimensional matrix representation. In order to overcome this, we encode the raw data in a 3-way tensor. Each slice captures a particular relation and is in the format of the document-term matrix in LSA. Just as in LSA, where the low-rank approximation by SVD helps generalize the representation and discover unseen relations, we apply a tensor decomposition method,

(a) Synonym layer (b) Antonym layer (c) Hypernym layer

Figure 3: The three slices of MRLSA raw tensor $\mathcal{W}$ for an example with vocabulary {joy, gladden, sorrow, sadden, anger, emotion, feeling} and target words {joyfulness, gladden, sad, anger}. Figures 3(a), 3(b), 3(c) show the matrices $\mathcal{W}_{:,:,\text{syn}}$, $\mathcal{W}_{:,:,\text{ant}}$, $\mathcal{W}_{:,:,\text{hyper}}$, respectively. Rows represent documents (see definition in text), and columns represent words. For ease of presentation, we show numbers with 0-1 values instead of TF×IDF scores.

the Tucker decomposition, to the tensor.

4.1 Representing Multi-Relational Data in Tensors

A tensor is simply a multi-dimensional array. In this work, we use a 3-way tensor $\mathcal{W}$ to encode multiple word relations. An element of $\mathcal{W}$ is denoted by $\mathcal{W}_{i,j,k}$ using its indices, and $\mathcal{W}_{:,:,k}$ represents the k-th *slice* of $\mathcal{W}$ (a slice of a 3-way tensor is a matrix, obtained by fixing the third index). Following (Kolda and Bader, 2009), a *fiber* of a tensor $\mathcal{W}_{:,j,k}$ is a vector, which is a high order analog of a matrix row or column.

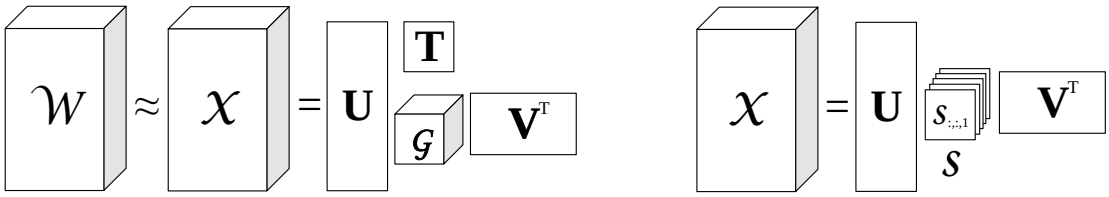
When constructing the raw tensor $\mathcal{W}$ in MRLSA, each slice is analogous to the document-term matrix in LSA, but created based on the data of a particular relation, such as synonyms. With a slight abuse of notation, we sometimes use the value rather than index when there is no confusion. For instance, $\mathcal{W}_{:,\text{“word”},k}$ represents the fiber corresponding to the “word” in slice k, and $\mathcal{W}_{:,:,\text{syn}}$ refers to the slice that encodes the synonymy relation. Below we use an example to compare this construction to the raw matrix in PILSA, and discuss how it extends LSA.

Suppose we are interested in representing two relations, synonymy and antonymy. The raw tensor in MRLSA would then consist of two slices, $\mathcal{W}_{:,:,\text{syn}}$ and $\mathcal{W}_{:,:,\text{ant}}$, to encode synonyms and antonyms of target words from a knowledge source (e.g., a thesaurus). Each row in $\mathcal{W}_{:,:,\text{syn}}$ represents the synonyms of a target word, and the corresponding row in $\mathcal{W}_{:,:,\text{ant}}$ encodes its antonyms. Figures 3(a) and 3(b) illustrate an example, where “joy”, ”gladden” are synonyms of the target word “joyfulness” and “sorrow” is its antonym. Therefore, the values of the corresponding entries are 1. Notice that the matrix $\mathbf{W}' = \mathcal{W}_{:,:,\text{syn}} - \mathcal{W}_{:,:,\text{ant}}$ is identical to the PILSA raw matrix. We can extend the construction above to enable MRLSA to utilize other semantic relations (e.g., *hypernymy*) by adding a slice corresponding to each relation of interest. Fig. 3(c) demonstrates how to add another slice $\mathcal{W}_{:,:,\text{hyper}}$ to the tensor for encoding hypernyms.

4.2 Tensor Decomposition

The MRLSA raw tensor encodes relations in one or more data resources, such as thesauri. However, the knowledge from a thesaurus is usually noisy and incomplete. In this section, we derive a low-rank approximation of the tensor to generalize the knowledge. This step is analogous to the rank-k approximation in LSA.

Various tensor decomposition methods have been proposed in literature. Among them, Tucker decomposition (Tucker, 1966) is recognized as a multi-dimensional extension of SVD and has been widely used in many applications. An illustration of this method is in Fig. 4(a). In Tucker decomposition, a $d \times n \times m$ tensor $\mathcal{W}$ is decomposed into four components $\mathcal{G}, \mathbf{U}, \mathbf{V}, \mathbf{T}$. A low-rank approximation



(a) Tucker Tensor Decomposition (b) Our Reformulation

Figure 4: Fig. 4(a) illustrates the Tucker tensor decomposition method which factors a 3-way tensor $\mathcal{W}$ to three orthogonal matrices, $\mathbf{U}, \mathbf{V}, \mathbf{T}$, and a core tensor $\mathcal{G}$. We further apply a n-mode matrix product on the core tensor $\mathcal{G}$ with $\mathbf{T}$. Consequently, each slice of the resulted core tensor $\mathcal{S}$ (a square matrix) captures a semantic relation type, and each column of $\mathbf{V}^T$ is a vector representing a word.

$\mathcal{X}$ of $\mathcal{W}$ is defined by

$$\mathcal{W}_{i,j,k} \approx \mathcal{X}_{i,j,k} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathcal{G}_{r_1,r_2,r_3} \mathbf{U}_{i,r_1} \mathbf{V}_{j,r_2} \mathbf{T}_{k,r_3},$$

where $\mathcal{G}$ is a core tensor with dimensions $R_1 \times R_2 \times R_3$ and $\mathbf{U}, \mathbf{V}, \mathbf{T}$ are orthogonal matrices with dimensions $d \times R_1, n \times R_2, m \times R_3$, respectively. The rank parameters $R_1 \leq d, R_2 \leq n, R_3 \leq m$ are given as input to the algorithm. In MRLSA, m (the number of relations) is usually small, while d and n are typically large (often in the scale of hundreds of thousands). Therefore, we choose $R_1 = R_2 = \tau$, $\tau \ll d, n$ and $R_3 = m$, where τ is typically less than 1000.

To make the analogy to SVD clear, we rewrite the results of Tucker decomposition by performing a n-mode matrix product over the core tensor $\mathcal{G}$ with the matrix $\mathbf{T}$. This produces a tensor $\mathcal{S}$ where each slice is a linear combination of the slices of $\mathcal{G}$ with coefficients given by $\mathbf{T}$ (see (Kolda and Bader, 2009) for detail). That is, we have

$$\mathcal{S}_{:,:,k} = \sum_{t=1}^{m} \mathbf{T}_{t,k} \mathcal{G}_{:,:,t}, \quad \forall k.$$

An illustration is shown in Fig. 4(b), Then, a straightforward calculation shows that k-th slice of tensor $\mathcal{W}$ is approximated by

$$\mathcal{W}_{:,:,k} \approx \mathcal{X}_{:,:,k} = \mathbf{U} \mathcal{S}_{:,:,k} \mathbf{V}^T. \tag{4}$$

Comparing Eq. (4) to Eq. (1), one can observe that matrices $\mathbf{U}$ and $\mathbf{V}$ play similar roles here, and each slice of the core tensor $\mathcal{S}$ is analogous to Σ. However, the square matrix $\mathcal{G}_{:,:,k}$ is not necessary to be diagonal. As in SVD, the column vectors of $\mathcal{G}_{:,:,k}\mathbf{V}^T$ (capture both word and relation information) behave similarly to the column vectors of the original tensor slice $\mathcal{W}_{:,:,k}$.

4.3 Measuring the Degrees of Word Relations

In principle, the raw information in the input tensor $\mathcal{W}$ can be used for computing lexical similarity using the cosine score between the column vectors for two words from the same slice of the tensor. To measure the degree of other relations, however, our approach requires one to specify a *pivot* slice. The key role of the pivot slice is to expand the lexical coverage of the relation of interest to additional lexical entries and, for this reason, the pivot slice should be chosen to capture the equivalence of the lexical entries. In this paper, we use the synonymy relation as our pivot slice. First we consider measuring the degree of a relation *rel* holding between the i-th and j-th words using the raw tensor $\mathcal{W}$, which can be computed as

$$\cos\left(\mathcal{W}_{:,i,\text{syn}}, \mathcal{W}_{:,j,rel}\right). \tag{5}$$

This measurement can be motivated from the logical rule: $\text{syn}(\text{word}_i, \text{target}) \wedge rel(\text{target}, \text{word}_j) \rightarrow rel(\text{word}_i, \text{word}_j)$, where the pivot relation *syn* expands the coverage of the relation of interest *rel*.

Turning to the use of the tensor decomposition, we use a similar derivation to Eq. (3), and measure the degree of relation *rel* between two words by

$$\cos\left(\mathcal{S}_{:,:,\text{syn}} \mathbf{V}_{i,:}^T, \mathcal{S}_{:,:,rel} \mathbf{V}_{j,:}^T\right). \tag{6}$$

For instance, the degree of antonymy between "joy" and "sorrow" is measured by the cosine similarity between the respective fibers $\cos(\mathcal{X}_{:,\text{"joy"},\text{syn}}, \mathcal{X}_{:,\text{"sorrow"},\text{ant}})$. We can encode both symmetric relations (e.g., *antonymy* and *synonymy*) and asymmetric relations (e.g., *hypernymy* and *hyponymy*) in the same tensor representation. For a symmetric relation, we use both $\cos(\mathcal{X}_{:,i,\text{syn}}, \mathcal{X}_{:,j,rel})$ and $\cos(\mathcal{X}_{:,j,\text{syn}}, \mathcal{X}_{:,i,rel})$ and measure the degree of a symmetric relation by the average of these two cosine similarity scores. However, for asymmetric relations, we use only $\cos(\mathcal{X}_{:,i,\text{syn}}, \mathcal{X}_{:,j,rel})$.

5 Experiments

We evaluate MRLSA on two tasks: answering the closest-opposite GRE questions and measuring degrees of various class-inclusion (i.e., *is-a*) relations. In both tasks, we design the experiments to empirically validate the following claims. When encoding two opposite relations from the same source, MRLSA performs comparably to PILSA. However, MRLSA generalizes LSA to model multiple relations, which could be obtained from both homogeneous and heterogeneous data sources. As a result, the performance of a target task can be further improved.

5.1 Experimental Setup

We construct the raw tensors to encode a particular relation in each slice based on two data sources.

Encarta The Encarta thesaurus is developed by Bloomsbury Publishing Plc[2]. For each target word, it provides a list of synonyms and antonyms. We use the same version of the thesaurus as in (Yih et al., 2012), which contains about 47k words and a vocabulary list of approximately 50k words.

WordNet We use four types of relations from WordNet: synonymy, antonymy, hypernymy and hyponymy. The number of target words and the size of the vocabulary in our version are 117,791 and 149,400, respectively. WordNet has better vocabulary coverage, but fewer antonym pairs. For instance, the WordNet antonym slice contains only 46,945 nonzero entries, while the Encarta antonym slice has 129,733.

[2]http://www.bloomsbury.com

We apply a memory-efficient Tucker decomposition algorithm (Kolda and Sun, 2008) implemented in *tensor_toolbox v2.5* (Bader et al., 2012)[3] to factor the tensor. The largest tensor considered in this paper can be decomposed in about 3 hours using less than 4GB of memory with a commodity PC.

5.2 Answering GRE Antonym Questions

The first task is to answer the closest-opposite questions from the GRE test provided by Mohammad et al. (2008)[4]. Each question in this test consists of a target word and five candidate words, where the goal is to pick the candidate word that has the most opposite meaning to the target word. In order to have a fair comparison, we use the same data split as in (Mohammad et al., 2008), with 162 questions used for the development set and 950 for test. Following (Mohammad et al., 2008; Yih et al., 2012), we report the results in precision (accuracy of the questions that the system attempts to answer), recall (percentage of the questions answered correctly over all questions) and F_1 (the harmonic mean of precision and recall).

We tune two sets of parameters using the development set: (1) the rank parameter τ in the tensor decomposition and (2) the scaling factors of different slices of the tensor. The rank parameter specifies the number of dimensions of the latent space. In the experiments, We pick the best value of τ from $\{100, 200, 300, 500, 750, 1000\}$. The scaling factors adjust the values of each slice of the tensor. The elements of each slice are multiplied by the scaling factor before factorization. This is important because Tucker decomposition minimizes the reconstruction error (the Frobenius norm of the residual tensor). As a result, the slice with a larger range of values becomes more influential to $\mathbf{U}$ and $\mathbf{V}$. In this work, we fix $\mathcal{W}_{:,:,\text{ant}}$, and search for the scaling factor of $\mathcal{W}_{:,:,\text{syn}}$ in $\{0.25, 0.5, 1, 2, 4\}$ and the factors of $\mathcal{W}_{:,:,\text{hyper}}$ and $\mathcal{W}_{:,:,\text{hypo}}$ in $\{0.0625, 0.125, 0.25\}$.

Table 1 summarizes the results of training

[3]`http://www.sandia.gov/~tgkolda/TensorToolbox`. The Tucker decomposition involves performing SVD on a large matrix. We modify the MATLAB code of *tensor_toolbox* to use the built-in `svd` function instead of `svds`. This modification reduces both the running time and memory usage.

[4]`http://www.saifmohammad.com`

	Dev. Set			Test Set		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1
WordNet Lookup	0.40	0.40	0.40	0.42	0.41	0.42
WordNet RawTensor	0.42	0.41	0.42	0.42	0.41	0.42
WordNet PILSA	0.63	0.62	0.62	0.60	0.60	0.60
WordNet MRLSA:Syn+Ant	0.63	0.62	0.62	0.59	0.58	0.59
WordNet MRLSA:4-layers	0.66	0.65	0.65	0.61	0.59	0.60
Encarta Lookup	0.65	0.61	0.63	0.61	0.56	0.59
Encarta RawTensor	0.67	0.64	0.65	0.62	0.57	0.59
Encarta PILSA	0.86	0.81	0.84	0.81	0.74	0.77
Encarta MRLSA:Syn+Ant	0.87	0.82	0.84	**0.82**	0.74	0.78
MRLSA:WordNet+Encarta	**0.88**	**0.85**	**0.87**	0.81	**0.77**	**0.79**

Table 1: GRE antonym test results of models based on Encarta and WordNet data in precision, recall and F_1. RawTensor evaluates the performance of the tensor with 2 slices encoding synonyms and antonyms before decomposition (see Eq. (5)), which is comparable to checking the original data directly (Lookup). MRLSA:Syn+Ant applies Tucker decomposition to the raw tensor and measures the degree of antonymy using Eq. (6). The result is similar to that of PILSA (see Sec. 3.1). MRLSA:4-layers adds hypernyms and hyponyms from WordNet; MRLSA:WordNet+Encarta consists of synonyms/antonyms from Encarta and hypernyms/hyponyms from WordNet, where the target words are aligned using the synonymy relations. Both models demonstrate the advantage of encoding more relations, from either the same or different resources.

MRLSA using two different corpora, Encarta and WordNet. The performance of the MRLSA raw tensor is close to that of looking up the thesaurus. This indicates the tensor representation is able to capture the word relations explicitly described in the thesaurus. After conducting tensor decomposition, MRLSA:Syn+Ant achieves similar results to PILSA. This confirms our claim that when giving the same among of information, MRLSA performs at least comparably to PILSA. However, the true power of MRLSA is its ability to incorporate other semantic relations to boost the performance of the target task. For example, when we add the hypernymy and hyponymy relations to the tensor, these class-inclusion relations provide a weak signal to help resolve antonymy. We suspect that this is due to the fact that antonyms typically share the same properties but only have the opposite meaning on one particular semantic dimension. For instance, the antonyms "sadness" and "happiness" are different forms of emotion. When two words are hyponyms of a target word, the likelihood that they are antonyms should thus be increased. We show that the target relations and these auxiliary semantic relations can be collected from the same data source (e.g., WordNet MRLSA:4-layers) or from multiple, heterogeneous sources (e.g., MRLSA:WordNET+Encarta). In both cases, the performance of the model improves as more relations are incorporated. Moreover, our experiments show that adding the hypernym and hyponym layers from WordNet improves modeling antonym relations based on the Encarta thesaurus. This suggests that the weak signal from a resource with a large vocabulary (e.g., WordNet) can help predict relations between out-of-vocabulary words and thus improve the recall.

To better understand the model, we examine the top antonyms for three question words from the GRE test. The lists below show antonyms and their MRLSA scores for each of the GRE question words as determined by the MRLSA:WordNET+Encarta model. Antonyms that can be found directly in the Encarta thesaurus are in italics.

inanimate *alive* (0.91), living (0.90), *bodily* (0.90), in-the-flesh (0.89), *incarnate* (0.89)

alleviate exacerbate (0.68), make-worse (0.67), inflame (0.66), amplify (0.65), stir-up (0.64)

relish detest (0.33), abhor (0.33), abominate (0.33), despise (0.33), loathe (0.31)

We can see that from these examples, MRLSA not

	Dev.	Test			
	1a (Taxonomic)	1b (Functional)	1c (Singular)	1d (Plural)	Avg.
WordNet Lookup	52.9	34.5	41.4	34.3	36.7
WordNet RawTensor	51.0	38.3	50.0	42.1	43.5
WordNet MRLSA:Syn+Hypony	55.8	41.7 (43.2)	51.0 (51.4)	37.5 (44.4)	43.4 (46.3)
WordNet MRLSA:4-layers	52.9	51.5 (53.9)	51.9 (60.0)	43.5 (50.5)	49.0 (54.8)
MRLSA:WordNet+Encarta	**62.1**	**55.3** (**58.7**)	**57.1** (**65.7**)	**48.6** (**53.7**)	**55.8** (**60.1**)
UTD_{NB} (Rink and Harabagiu, 2012)	-	38.3	36.7	28.2	34.4

Table 2: Results of measuring the class-inclusion (*is-a*) relations in MaxDiff accuracy (see text for detail). RawTensor has synonym and hyponym slices and measures the degree of *is-a* relation using Eq. (5). MRLSA:Syn+Hypo factors the raw tensor and judges the relation by Eq. (6). The constructions of MRLSA:4-layers and MRLSA:WordNet+Encarta are the same as in Sec. 5.2 (see the caption of Table 1 for detail). For MRLSA models, numbers shown in the parentheses are the results when parameters are tuned using the test sets. UTD_{NB} is the results of the best performing system in SemEval-2012 Task 2.

only preserves the antonyms in the thesaurus, but also discovers additional ones, such as *exacerbate* and *inflame* for "alleviate". Another interesting finding is that while the scores are useful in ranking the candidate words, they might not be comparable across different question words. This could be an issue for some applications, which need to make a binary decision on whether two words are antonyms.

5.3 Measuring degrees of *Is-A* relations

We evaluate MRLSA using the class-inclusion portion of SemEval-2012 Task 2 data (Jurgens et al., 2012). Here the goal is to measure the degree of two words having the *is-a* relation. Five annotated datasets are provided for different subcategories of this relation: 1a-taxonomic, 1b-functional, 1c-singular, 1d-plural, 1e-class individual. We omit 1e because it focuses on real world entities (e.g., queen:Elizabeth, river:Nile), which are not included in WordNet.

Each dataset contains about 100 questions based on approximately 40 word pairs. The question consists of 4 randomly chosen word pairs and asks the best and worst pairs that exemplify the specific *is-a* relation. The performance is measured by the average prediction accuracy, also called the MaxDiff accuracy (Louviere and Woodworth, 1991).

Because the questions are generated from the same set of word pairs, these questions are not mutually independent. Therefore, it is not proper to split the data of each subcategory into the development and test sets. Alternatively, we follow the setting of SemEval-2012 Task 2 and use the first subcategory (1a-taxonomy) to tune the model and evaluate its performance based on the results on other datasets. Since the models are tuned and tested on different types of subcategories, they might not be the optimal ones when evaluated on the test sets. Therefore, we show results using the best parameters tuned on the development set and those tuned on the test set, where the latter suggests a performance upper-bound. Besides the rank parameter, we tune the scaling factors of the synonym, hypernym and hyponym slices from $\{4, 16, 64\}$. The scaling factor of the antonym slice is fixed to 1.

Table 2 shows the performance in MaxDiff accuracy. Results show that even the raw tensor representation (RawTensor) performs better than WordNet lookup. We suspect that this is because the tensor representation can capture the fact that the hyponyms of a word are usually synonymous to each other. By performing Tucker decomposition on the raw Tensor, MRLSA achieves better performance. MRLSA:4-layers further leverages the information from antonyms and hypernyms and thus improves the model. As we notice in the GRE antonym test, models based on the Encarta thesaurus perform better in predicting antonyms. Therefore, it is interesting to check if combining synonyms and antonyms from Encarta helps. As a result, MRLSA:WordNet+Encarta improves over MRLSA:4-layers significantly. This demonstrates again that MRLSA can leverage knowledge stored in heterogeneous resources. Notably, MRLSA outper-

forms the best system participated in the SemEval-2012 task with a large margin, with a difference of 21.4 in MaxDiff accuracy.

Next we examine the top words that have the *is-a* relation relative to three question words from the task. The lists below show the hyponyms and their respective MRLSA scores for each of the question words as determined by MRLSA:4-layers.

bird ostrich (0.75), gamecock (0.75), nighthawk (0.75), amazon (0.74), parrot (0.74)

automobile minivan (0.48), wagon (0.48), taxi (0.46), minicab (0.45), gypsy cab (0.45)

vegetable buttercrunch (0.61), yellow turnip (0.61), romaine (0.61), chipotle (0.61), chilli (0.61)

Although the model in general does a good job finding hyponyms, we observe that some suggested words, such as *buttercrunch* (a mild lettuce) vs. "vegetable", do not seem intuitive (e.g., compared to *carrot*). Having one additional slice to capture the general term co-occurrence relation may help improve the model in this respect.

6 Conclusions

In this paper, we propose Multi-Relational Latent Semantic Analysis (MRLSA) which generalizes Latent Semantic Analysis (LSA) for lexical semantics. MRLSA models multiple word relations by leveraging a 3-way tensor, where each slice captures one particular relation. A low-rank approximation of the tensor is then derived using a tensor decomposition. Consequently, words in the vocabulary are represented by vectors in the latent semantic space, and each relation is captured by a latent square matrix. Given two words, MRLSA not only can measure their degree of having a specific relation, but also can discover unknown relations between them. These advantages have been demonstrated in our experiments. By encoding relations from both homogeneous or heterogeneous data sources, MRLSA achieves state-of-the-art performance on existing benchmark datasets for two relations, *antonymy* and *is-a*.

For future work, we plan to explore directions that aim for improving both the quality and word coverage of the model. For instance, the knowledge encoded by MRLSA can be enriched by adding more relations from a variety of linguistic resources, including the co-occurrence relations from large corpora. On model refinement, we notice that MRLSA can be viewed as a 3-layer neural network without applying the sigmoid function. Following the strategy of using Siamese neural networks to enhance PILSA (Yih et al., 2012), training MRLSA with a multi-task discriminative learning setting can be a promising approach as well.

Acknowledgments

We thank Geoff Zweig for valuable discussions and the anonymous reviewers for their comments.

References

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca and A. Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *NAACL '09*, pages 19–27.

Brett W. Bader, Tamara G. Kolda, et al. 2012. Matlab tensor toolbox version 2.5. Available online, January.

David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan L Boyd-Graber, David M Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *EMNLP-CoNLL*, pages 1024–1033.

Shay B. Cohen, Giorgio Satta, and Michael Collins. 2013. Approximate PCFG parsing using tensor decomposition. In *NAACL-HLT 2013*, pages 487–496.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(96).

S. Dumais, T. Letsche, M. Littman, and T. Landauer. 1997. Automatic cross-language retrieval using latent semantic indexing. In *AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval*.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *ACL 2012*, pages 864–872.

Weiwei Guo and Mona Diab. 2013. Improving lexical semantics for sentential semantics: Modeling selectional preference and similar words in a latent variable model. In *NAACL-HLT 2013*, pages 739–745.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 289–296.

D. Jurgens, S. Mohammad, P. Turney, and K. Holyoak. 2012. SemEval-2012 Task 2: Measuring degrees of relational similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 356–364.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September.

Tamara G. Kolda and Jimeng Sun. 2008. Scalable tensor decompositions for multi-aspect data mining. In *ICDM 2008*, pages 363–372.

T. Landauer and D. Laham. 1998. Learning human-like knowledge by singular value decomposition: A progress report. In *NIPS 1998*.

T. Landauer. 2002. On the computational basis of learning and cognition: Arguments from lsa. *Psychology of Learning and Motivation*, 41:43–84.

Jordan J. Louviere and G. G. Woodworth. 1991. Best-worst scaling: A model for the largest difference judgments. Technical report, University of Alberta.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *NAACL-HLT 2013*.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word pair antonymy. In *Empirical Methods in Natural Language Processing (EMNLP)*.

John Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual document representations from discriminative projections. In *Proceedings of EMNLP*, pages 251–261.

Xipeng Qiu, Le Tian, and Xuanjing Huang. 2013. Latent semantic tensor indexing for community-based question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 434–439, Sofia, Bulgaria, August. Association for Computational Linguistics.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of HLT-NAACL*, pages 109–117.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT 2013*, pages 74–84.

Bryan Rink and Sanda Harabagiu. 2012. UTD: Determining relational similarity using lexical patterns. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 413–418, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

G. Salton, A. Wong, and C. S. Yang. 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11).

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML '11*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

P. D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Peter Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *International Conference on Computational Linguistics (COLING)*.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1142–1151, Atlanta, Georgia, June. Association for Computational Linguistics.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, New York, NY, USA. ACM.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of NAACL-HLT*, pages 616–620, Montréal, Canada, June.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256, Portland, Oregon, USA, June. Association for Computational Linguistics.

Wen-tau Yih, Geoffrey Zweig, and John Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of NAACL-HLT*, pages 1212–1222, Jeju Island, Korea, July.

Alisa Zhila, Wen-tau Yih, Christopher Meek, Geoffrey Zweig, and Tomas Mikolov. 2013. Combining heterogeneous models for measuring relational similarity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1000–1009, Atlanta, Georgia, June. Association for Computational Linguistics.

A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else)

Ivan Vulić and Marie-Francine Moens
Department of Computer Science
KU Leuven
Celestijnenlaan 200A
Leuven, Belgium
{ivan.vulic,marie-francine.moens}@cs.kuleuven.be

Abstract

We present a new language pair agnostic approach to inducing bilingual vector spaces from non-parallel data without any other resource in a bootstrapping fashion. The paper systematically introduces and describes all key elements of the bootstrapping procedure: (1) starting point or seed lexicon, (2) the confidence estimation and selection of new dimensions of the space, and (3) convergence. We test the quality of the induced bilingual vector spaces, and analyze the influence of the different components of the bootstrapping approach in the task of bilingual lexicon extraction (BLE) for two language pairs. Results reveal that, contrary to conclusions from prior work, the seeding of the bootstrapping process has a heavy impact on the quality of the learned lexicons. We also show that our approach outperforms the best performing fully corpus-based BLE methods on these test sets.

1 Introduction

Bilingual lexicons serve as an indispensable source of knowledge for various cross-lingual tasks such as cross-lingual information retrieval (Lavrenko et al., 2002; Levow et al., 2005) or statistical machine translation (Och and Ney, 2003). Additionally, they are a crucial component in cross-lingual knowledge transfer, where the knowledge about utterances in one language may be transferred to another. The utility of the transfer or annotation projection by means of bilingual lexicons has already been proven in various tasks such as semantic role labeling (Padó and Lapata, 2009; van der Plas et al., 2011), parsing (Zhao et al., 2009; Durrett et al., 2012; Täckström et al., 2013b), POS tagging (Yarowsky and Ngai, 2001; Das and Petrov, 2011; Täckström et al., 2013a), etc.

Techniques for automatic bilingual lexicon extraction (BLE) from parallel corpora on the basis of word alignment models are well established (Och and Ney, 2003). However, due to a relative scarceness of parallel data for many language pairs and domains, alternative approaches that rely on comparable corpora have also gained much interest (e.g., Fung and Yee (1998); Rapp (1999)).

The models that rely on non-parallel data typically represent each word by a high-dimensional vector in a feature vector space, where the dimensions of the vector are its *context features*. The context features are typically words co-occurring with the word in a predefined context.[1] The similarity of two words, w_1^S given in the source language L_S with vocabulary V^S and w_2^T in the target language L_T with vocabulary V^T is then computed as $sim(w_1^S, w_2^T) = SF(cv(w_1^S), cv(w_2^T))$. $cv(w_1^S) = [sc_1^S(c_1), \ldots, sc_1^S(c_N)]$ is a context vector for w_1^S with N context features c_k, where $sc_1^S(c_k)$ denotes the score for w_1^S associated with context feature c_k (similar for w_2^T). SF is a similarity function (e.g., cosine, the Kullback-Leibler divergence, the Jaccard index) operating on the context vectors (Lee, 1999).

When operating with 2 languages, the context features cannot be compared directly. Therefore, in order to compare the feature vectors $cv(w_1^S)$ and $cv(w_2^T)$, the context features need to span a shared

[1] The context may be a document, a paragraph, a window of predefined size around each occurrence of w_i^S in $\mathcal{C}_S$, etc. For an overview, see, e.g., (Tamura et al., 2012).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1613–1624,
Seattle, Washington, USA, 18-21 October 2013.

bilingual vector space. The standard way of building a bilingual vector space is to use *bilingual lexicon entries* (Rapp, 1999; Fung and Cheung, 2004; Gaussier et al., 2004) as dimensions of the space. However, there seems to be an apparent flaw in logic, since the methods assume that there exist readily available bilingual lexicons that are then used to induce bilingual lexicons! Therefore, the focus of the researchers has turned to designing BLE methods that do not rely on any external translation resources such as machine-readable bilingual lexicons and parallel corpora (Haghighi et al., 2008; Vulić et al., 2011).

In order to circumvent this issue, one line of recent work aims to *bootstrap high-quality bilingual vector spaces from a small initial seed lexicon*. The seed lexicon is constructed by harvesting identical or similarly spelled words across languages (Koehn and Knight, 2002; Peirsman and Padó, 2010), and it spans the initial bilingual vector space. *The space is then gradually enriched with new dimensions/axes during the bootstrapping procedure*. The bootstrapping process has already proven its validity in inducing bilingual lexicons for closely similar languages such as Spanish-Portuguese or Croatian-Slovene (Fišer and Ljubešić, 2011), but it still lacks further generalization to more distant language pairs.

The main goal of this paper is to shed new light on the bootstrapping approaches to bilingual lexicon extraction, and to construct a language pair agnostic bootstrapping method that is able to build high-quality bilingual vector spaces that consequently lead to high-quality bilingual lexicons for more distant language pairs where orthographic similarity is not sufficient to seed bilingual vector spaces. We aim to answer the following key questions:

- How to seed bilingual vector spaces besides using only orthographically similar words?
- Is it better to seed bilingual spaces with translation pairs/dimensions that are frequent in the corpus, and does the frequency matter at all? Does the size of the initial seed lexicon matter?
- How to enrich bilingual vector spaces with only highly reliable dimensions in order to prevent semantic drift?

With respect to these questions, the main contributions of this article are:

- We present a complete overview of the framework of bootstrapping bilingual vector spaces from non-parallel data without any additional resources. We dissect the bootstrapping process and describe all its key components.
- We introduce a new way of seeding the bootstrapping procedure that does not rely on any orthographic clues and that yields bilingual vector spaces of higher quality. We analyze the impact of different seed lexicons on the quality of induced bilingual vector spaces.
- We show that in the setting without any external translation resources, our bootstrapping approach yields lexicons that outperform the best performing corpus-based BLE methods on standard test datasets for 2 language pairs.

2 Boostrapping Bilingual Vector Spaces: A General Overview

This section presents the complete bootstrapping procedure that starts with an initial seed lexicon which spans the initial bilingual vector space, while as the output in each iteration of the procedure it produces an updated bilingual vector space that can be used to extract a bilingual lexicon.

2.1 General Framework

We assume that we are solely in possession of a (non-parallel) bilingual corpus $\mathcal{C}$ that is composed of a sub-corpus $\mathcal{C}_S$ given in the source language L_S, and a sub-corpus $\mathcal{C}_T$ in the target language L_T. All word types that occur in $\mathcal{C}_S$ constitute a set V^S. All word types in $\mathcal{C}_T$ constitute a set V^T. The goal is to build a bilingual vector space using only corpus $\mathcal{C}$.

Assumption 1. *Dimensions of the bilingual vector space are one-to-one word translation pairs*. For instance, dimensions of a Spanish-English space are pairs like *(perro, dog)*, *(ciencia, science)*, etc. The *one-to-one constraint* (Melamed, 2000), although not valid in general, simplifies the construction of the bootstrapping procedure. $\mathcal{Z}$ denotes the set of translation pairs that are the dimensions of the space.

Computing cross-lingual word similarity in a bilingual vector space. Now, assume that our bilingual vector space consists of N one-to-one word translation pairs $c_k = (c_k^S, c_k^T), k = 1, \ldots, N$. For each word $w_i^S \in V^S$, we compute the similarity of

that word with each word $w_j^T \in V^T$ by computing the similarity between their context vectors $cv(w_i^S)$ and $cv(w_j^T)$, which are actually their representations in the N-dimensional bilingual vector space.

The cross-lingual similarity is computed following the standard procedure (Gaussier et al., 2004):
(1) For each source word $w_i^S \in V^S$, build its N-dimensional context vector $cv(w_i^S)$ that consists of association scores $sc_k^S(c_k^S)$, that is, we compute the strength of association with the "source" part of each dimension c_k that constitutes the N-dimensional bilingual space. The association is dependent on the co-occurrence of w_i^S and c_k^S in a predefined context. Various functions such as the log-likelihood ratio (LLR) (Rapp, 1999; Ismail and Manandhar, 2010), TF-IDF (Fung and Yee, 1998), or pointwise mutual information (PMI) (Bullinaria and Levy, 2007; Shezaf and Rappoport, 2010) are typically used as *weighting functions* to quantify the strength of the association.
(2) Repeat step (1) for each target word $w_j^T \in V^T$ and build context vectors $cv(w_j^T)$ that consist of scores $sc_k^T(c_k^T)$.
(3) Since c_k^S and c_k^T address the same dimension c_k in the bilingual vector space for each $k = 1, \ldots, N$, we are able to compute the similarity between $cv(w_i^S)$ and $cv(w_j^T)$ using any similarity measure such as the Jaccard index, the Kullback-Leibler or the Jensen-Shannon divergence, the cosine measure, or others (Lee, 1999; Cha, 2007).

The similarity score for two words w_i^S and w_j^T is $sim(w_i^S, w_j^T)$. For each source word w_i^S, we can build a *ranked list* $RL(w_i^S)$ that consists of all words $w_j^T \in V^T$ ranked according to their respective similarity scores $sim(w_i^S, w_j^T)$. In the similar fashion, we can build a ranked list $RL(w_j^T)$, for each target word w_j^T. We call the top scoring target word w_j^T for some source word w_i^S its *translation candidate*, and write $TC(w_i^S) = w_j^T$. Additionally, we label the ranked list $RL(w_i^S)$ that is pruned at position M as $RL_M(w_i^S)$.

Bootstrapping. The key idea of the bootstrapping approach relies on an insight that *highly reliable translation pairs* (w_1^S, w_2^T) that are encountered using the N-dimensional bilingual vector space might be added as new dimensions of the space. By adding these new dimensions, it might be possible to extract more highly reliable translation pairs that were previously not used as dimensions of the space, and the iterative procedure repeats until no new dimensions are found. The induced bilingual vector space may then be observed as a bilingual lexicon *per se*, but it may also be used to find translation equivalents for other words which are not used to span the space.

Algorithm 1: Bootstrapping a bilingual vector space

Input : Bilingual corpus $\mathcal{C} = \mathcal{C}_S \cup \mathcal{C}_T$
Initialize: (1) Obtain a one-to-one seed lexicon. The entries from the lexicon are initial dimensions of the space: $\mathcal{Z}_0$; (2) $s = 0$;
Bootstrap:
repeat
1: For each $w_i^S \in V^S$: compute $RL(w_i^S)$ using $\mathcal{Z}_s$;
2: For each $w_j^T \in V^T$: compute $RL(w_j^T)$ using $\mathcal{Z}_s$;
3: For each $w_i^S \in V^S$ and $w_j^T \in V^T$: score each translation pair $(w_i^S, TC(w_i^S))$ and $(TC(w_j^T), w_j^T)$ and add them to a *pool of candidate dimensions* ;
4: Choose *the best candidates* from the pool and add them as new dimensions: $\mathcal{Z}_{s+1} \leftarrow \mathcal{Z}_s \cup \{best\}$;
5: Resolve collisions in $\mathcal{Z}_{s+1}$;
6: $s \leftarrow s + 1$;
until *no new dimensions are found (convergence)* ;
Output: One-to-one translation pairs $\rightarrow$ Dimensions of a bilingual vector space $\mathcal{Z}_{final}$

The overview of the procedure as given by alg. 1 reveals these crucial points in the procedure: (Q1) how to provide initial dimensions of the space? (the initialization step), (Q2) how to score each translation pair, estimate their confidence, and how to choose the best candidates from the pool of candidates? (steps 3 and 4), and (Q3) how to resolve potential collisions that violate the one-to-one constraint? (step 5). We will discuss (Q1) and (Q2) in more detail later, while we resolve (Q3) following a simple heuristic as follows:

Assumption 2. *In case of collision, dimensions/pairs that are found at later stages of the bootstrapping process overwrite previous dimensions.*

The intuition here is that we expect for the quality of the space to increase at each stage of the bootstrapping process, and newer translation pairs should be more confident than the older ones. For instance, if 2 out of N dimensions of a Spanish-English bilingual space are pairs *(piedra,wall)* and *(tapia,stone)*, but then if during the bootstrapping process we extract a new candidate pair *(piedra,stone)*, we will delete the former two dimensions and add the latter.

2.2 Initializing Bilingual Vector Spaces

Seeding or initializing a bootstrapping procedure is often a critical step regardless of the actual task (McIntosh and Curran, 2009; Kozareva and Hovy, 2010), and it decides whether the complete process will end as a success or a failure. However, Peirsman and Padó (2011) argue that the initialization step is not crucial when dealing with bootstrapping bilingual vector spaces. Here, we present two different strategies of initializing the bilingual vector space.

Identical words and cognates. Previous work relies exclusively on identical and similarly spelled words to build the initial set of dimensions $\mathcal{Z}_0$ (Koehn and Knight, 2002; Peirsman and Padó, 2010; Fišer and Ljubešić, 2011). This strategy yields promising results for closely similar language pairs, but is of limited use for other language pairs.

High-frequency seeds. Another problem with using only identical words and cognates as seeds lies in the fact that many of them might be infrequent in the corpus, and as a consequence the expressiveness of a bilingual vector space might be limited. On the other hand, high-frequency words offer a lot of evidence in the corpus that could be exploited in the bootstrapping approach. In order to induce initial translation pairs, we rely on the framework of *multilingual probabilistic topic modeling* (MuPTM) (Boyd-Graber and Blei, 2009; De Smet and Moens, 2009; Mimno et al., 2009; Zhang et al., 2010), that does not require a bilingual lexicon, it operates with non-parallel data, and is able to produce highly confident translation pairs for high-frequency words (Mimno et al., 2009; Vulić and Moens, 2013).[2] Therefore, we can construct the initial seed lexicon as follows:

(1) Train a multilingual topic model on the corpus.

(2) Obtain one-to-one translation pairs using any of the MuPTM-based models of cross-lingual similarity, e.g., (Vulić et al., 2011; Vulić and Moens, 2013).

(3) Retain only *symmetric* translation pairs. This step ensures that only highly confident pairs are used as seed translation pairs.

(4) Rank translation pairs according to their frequency in the corpus and use a subset of the most frequent symmetric pairs as seeds.

2.3 Estimating Confidence of New Dimensions

Another crucial step in the bootstrapping procedure is the estimation of confidence in a translation pair/candidate dimension. Errors in the early stages of the procedure may negatively affect the learning process and even cause *semantic drift* (Riloff and Shepherd, 1999; McIntosh and Curran, 2009). We therefore impose the constraint which requires translation pairs to be *symmetric* in order to qualify as potential new dimensions of the space. In other words, given the current set of dimensions $\mathcal{Z}_s$, a translation pair (w_i^S, w_j^T) has a possibility to be chosen as a new dimension from the pool of candidate dimensions if and only if it holds: $TC(w_i^S) = w_j^T$ and $TC(w_j^T) = w_i^S$. This *symmetry constraint* should ensure a relative reliability of translation pairs.

In each iteration of the bootstrapping process, we may add all symmetric pairs from the pool of candidates as new dimensions, or we could impose additional selection criteria that quantify the degree of confidence in translation pairs. We are then able to rank the symmetric candidate translation pairs in the pool of candidates according to their confidence scores (step 3 of alg. 1), and choose only the best B candidates from the pool in each iteration (step 4) as done in (Thelen and Riloff, 2002; McIntosh and Curran, 2009; Huang and Riloff, 2012). By picking only a subset of the B most confident candidates in each iteration, we hope to further prevent a possibility of semantic drift, i.e., "poisoning" the bootstrapping process that might happen if we include incorrect translation pairs as dimensions of the space.

In this paper, we investigate 3 different confidence estimation functions:[3]

(1) **Absolute similarity score.** Confidence of a translation pair $CF(w_i^S, TC(w_i^S))$ is simply the absolute similarity value $sim(w_i^S, TC(w_i^S))$

(2) **M-Best confidence function.** It contrasts the score of the translation candidate with the average score over the first M most similar words in the ranked list. The larger the difference, the more confidence we have in the translation candidate. Given a word $w_i^S \in V^S$ and a ranked list $RL_M(w_i^S)$, the

[2] One can also use other models that are similar to MuPTM such as (Haghighi et al., 2008; Daumé III and Jagarlamudi, 2011) to produce the initial seed lexicon, but that analysis is beyond the scope of this work.

[3] A symmetrized version of the confidence functions is computed as the geometric mean of source-to-target and target-to-source confidence scores.

average score of the best M words is computed as:

$$sim_M(w_i^S) = \frac{1}{M} \sum_{w_j^T \in RL_M(w_i^S)} sim(w_i^S, w_j^T)$$

The final confidence score is then:

$$CF(w_i^S, TC(w_i^S)) = sim(w_i^S, TC(w_i^S)) - sim_M(w_i^S)$$

(3) **Entropy-based confidence function.** We adapt the well-known entropy-based confidence (Smith and Eisner, 2007; Tu and Honavar, 2012) to this particular task. First, we need to define a distribution:

$$p(w_j^T|w_i^S) = \frac{e^{sim(w_i^S, w_j^T)}}{\sum_{w_l^T \in V^T} e^{sim(w_i^S, w_l^T)}}$$

The confidence function is then minus the entropy of the probability distribution p:

$$CF(w_i^S, TC(w_i^S)) = \sum_{w_l^T \in V^T} p(w_l^T|w_i^S) \log p(w_l^T|w_i^S)$$

3 Experimental Setup

Data collections. We investigate our bootstrapping approach on the BLE task for 2 language pairs: Spanish-English (ES-EN) and Italian-English (IT-EN), and work with the following corpora previously used by Vulić and Moens (2013): (i) a collection of 13,696 Spanish-English Wikipedia article pairs (**Wiki-ES-EN**), (ii) 18,898 Italian-English Wikipedia article pairs (**Wiki-IT-EN**).[4]

Following (Koehn and Knight, 2002; Haghighi et al., 2008; Prochasson and Fung, 2011; Vulić and Moens, 2013), we use TreeTagger (Schmid, 1994) for POS-tagging and lemmatization of the corpora, and then retain only nouns that occur at least 5 times in the corpus. We record the lemmatized form when available, and the original form otherwise. Our final vocabularies consist of 9,439 Spanish nouns and 12,945 nouns for ES-EN, and 7,160 Italian nouns and 9,116 English nouns for IT-EN.

Ground truth. The goal of the BLE task is to extract a bilingual lexicon of one-to-one translations. In order to test the quality of bilingual vector spaces induced by our bootstrapping approach, we evaluate it on standard 1000 ground truth one-to-one translation pairs built for the Wiki-ES-EN and Wiki-IT-EN datasets (Vulić and Moens, 2013). Note that we do not explicitly test the bilingual vector space as a bilingual lexicon, but rather its ability to find semantically similar words and translations also for words that are not used as dimensions of the space (see sect. 2.1).

Evaluation metrics. We measure the performance on the BLE task using a standard *Top M* accuracy (Acc_M) metric. It denotes the number of source words w_i^S from ground truth translation pairs whose list $RL_M(w_i^S)$ contains the correct translation according to our ground truth over the total number of ground truth translation pairs (=*1000*) (Gaussier et al., 2004; Tamura et al., 2012).[5] Additionally, we report the *mean reciprocal rank (MRR)* scores (Voorhees, 1999) for some experimental runs.

Multilingual topic model. We utilize a straightforward multilingual extension of the standard Blei et al.'s LDA model (Blei et al., 2003) called *bilingual LDA* (Mimno et al., 2009; Ni et al., 2009; De Smet and Moens, 2009). BiLDA training follows the procedure from (Vulić and Moens, 2013), that is, the training method is Gibbs sampling with the number of topics set to $K = 2000$. Hyperparameters of the model are set to standard values (Steyvers and Griffiths, 2007): $\alpha = 50/K$ and $\beta = 0.01$.

Building initial seed lexicons. To produce the lists of one-to-one translation pairs that are used as seeds for the bootstrapping approach (see sect. 2.2), we experiment with the *TopicBC* and the *ResponseBC* methods from (Vulić and Moens, 2013), which are the MuPTM-based models of cross-lingual semantic similarity that obtain the best results in the BLE task on these datasets. In short, the *TopicBC* method computes the similarity of two words according to the similarity of their conditional topic distributions (Griffiths et al., 2007; Vulić et al., 2011) using

[4] Vulić and Moens (2013) also worked with Dutch-English (NL-EN), but we have decided to leave out the results obtained for that language pair due to space constraints, high similarity between the two languages, and the fact that the results obtained for that language pair are qualitatively similar to the results we report for ES-EN and IT-EN. Hence including the results for NL-EN would not contribute to the paper with any new important insight and conclusion.

[5] We can build a one-to-one bilingual lexicon by harvesting one-to-one translation pairs $(w_i^S, TC(w_i^S))$, and the quality of that lexicon is best reflected in the Acc_1 score.

the Bhattacharyya coefficient (BC) (Kazama et al., 2010) as the similarity function. *ResponseBC* is a second-order similarity method. It first computes initial similarity scores between all words cross-lingually and monolingually using the cross-lingual topical space and, in the second step, it computes the similarity between 2 words as the similarity between their word vectors that now contain the initial word-to-word similarity scores with all source and target words. The similarity function is again BC.

We use these models of similarity as a black box to acquire seeds for the bootstrapping approach, but we encourage the interested reader to find more details about the methods in the relevant literature. These two models also serve as our *baseline models*, and our goal is to test whether we are able to obtain bilingual lexicons of higher quality using bootstrapping that starts from the output of these models.

Weighting and similarity functions. We have experimented with different families of weighting (e.g., PMI, LLR, TF-IDF, chi-square) and similarity functions (e.g., cosine, Dice, Kullback-Leibler, Jensen-Shannon) (Lee, 1999; Turney and Pantel, 2010). In this paper, we present results obtained by *positive pointwise mutual information (PPMI)* (Niwa and Nitta, 1994) as a weighting function, which is a standard choice in vector space semantics (Turney and Pantel, 2010), and (combined with cosine) yields the best results over a group of semantic tasks according to (Bullinaria and Levy, 2007). We use a smoothed version of PPMI as presented in (Pantel and Lin, 2002; Turney and Pantel, 2010). Again, based on the results reported in the relevant literature (Bullinaria and Levy, 2007; Laroche and Langlais, 2010; Turney and Pantel, 2010), we opt for the cosine similarity as a standard choice for SF. We have also experimented with different window sizes ranging from 3 to 15 in both directions around the pivot word, but we have not detected any major qualitative difference in the results and their interpretation. Therefore, all results reported in the paper are obtained by setting the window size to 6.

4 Results and Discussion

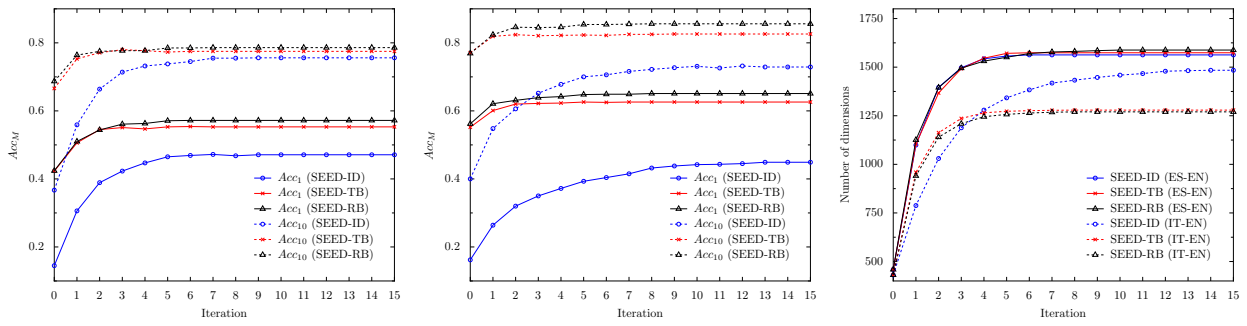
4.1 Are Seeds Important?

In recent work, Peirsman and Padó (2010; 2011) report that "the size and quality of the (seed) lexicon are not of primary importance given that the bootstrapping procedure effectively helped filter out incorrect translation pairs and added more newly identified mutual nearest neighbors." According to their findings, (1) noisy translation pairs are corrected in later stages of the bootstrapping process, since the quality of bilingual vector spaces gradually increases, (2) the size of the seed lexicon does not matter since the bootstrapping approach is able to learn translation pairs that were previously not present in the seed lexicon. Additionally, they do not provide any insight whether the frequency of seeds in the corpus influences the quality of induced bilingual vector spaces. In this paper, we question these claims with a series of BLE experiments.

All experiments conducted in this section do not rely on any extra confidence estimation except for the symmetry constraint, that is, in each step we enrich the bilingual vector space with all new symmetric translation pairs (see alg. 1 and sect. 2.3).

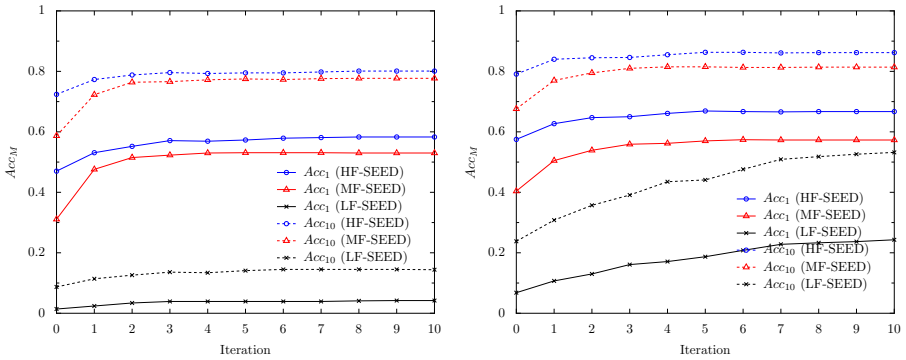
Exp. I: Same size, different seedings? The goal of this experiment is to test whether the quality of seeds plays an important role in the bootstrapping approach. We experiment with 3 different seed lexicons: (1) Following (Peirsman and Padó, 2010; Fišer and Ljubešić, 2011), we harvest identically spelled words across 2 languages and treat them as one-to-one translations. This procedure results in 459 seed translation pairs for ES-EN, and 431 pairs for IT-EN (SEED-ID), (2) We obtain symmetric translation pairs using the *TopicBC* method (see sect. 3) and use 459 pairs that have the highest frequency in the Wiki-ES-EN corpus as seeds for ES-EN (similarly 431 pairs for IT-EN) (SEED-TB), (3) As in (2), but we now use the *ResponseBC* method to acquire seeds (SEED-RB). The frequency of a one-to-one translation pair is simply computed as the geometric mean of the frequencies of words that constitute the translation pair.

Fig. 1(a) and 1(b) display the progress of the same bootstrapping procedure using the 3 different seed lexicons. We derive several interesting conclusions: (i) *Regardless of the actual choice of the seeding method, the bootstrapping process proves its validity and utility* since we observe that the quality of induced bilingual vector spaces increases over time for all 3 seeding methods. The bootstrapping procedure converges quickly. The increase is especially



(a) Spanish to English (b) Italian to English (c) # Dimensions over iterations

Figure 1: Results with 3 different seeding methods as starting points of the bootstrapping process: (i) identical words only (SEED-ID), (ii) the *TopicBC* method (SEED-TB), (iii) the *ResponseBC* method (SEED-RB). (a) Acc_M scores for ES-EN; (b) Acc_M scores for IT-EN; (c) the number of dimensions in the space with the 3 different seeding methods in each iteration for ES-EN and IT-EN. The bootstrapping procedure typically converges after a few iterations.



(a) Spanish to English (b) Italian to English

Figure 2: Results on the BLE task with SEED-RB when using seed translation pairs of different frequency: (i) high-frequency (HF-SEED), (ii) medium-frequency (MF-SEED), (iii) low-frequency (LF-SEED).

prominent in the first few iterations, when the approach learns more new dimensions (see fig. 1(c)). (ii) *The seeding method is important.* A bootstrapping approach that starts with a better seed lexicon is able to extract bilingual lexicons of higher quality as reflected in Acc_1 scores. Although the bootstrapping approach seems more beneficial when dealing with noisier seed lexicons (226% increase in terms of Acc_1 for ES-EN and 177% increase for IT-EN when starting with SEED-ID, compared to 35% increase for ES-EN, and 15% for IT-EN with SEED-RB), when starting from a noisy seed lexicon such as SEED-ID the method is unable to reach the same level of performance. Starting with SEED-ID, the approach is able to recover noisy dimensions from an initial bilingual vector space, but it is still unable to match the results that are obtained when starting from a better initial space (e.g., SEED-RB).

(iii) SEED-RB produces slightly better results than SEED-TB (e.g., the final Acc_1 of 0.649 for SEED-RB compared to 0.626 for SEED-TB for IT-EN, and 0.572 compared to 0.553 for ES-EN). This finding is in line with the results reported in (Vulić and Moens, 2013) where *ResponseBC* proved to be a more robust and a more effective method when applied to the BLE task directly. In all further experiments we use *ResponseBC* to acquire seed pairs, i.e., the seeding method is SEED-RB.

Exp. II: Does the frequency of seeds matter? In the next experiment, we test whether the frequency of seeds in the corpus plays an important role in the bootstrapping process. The intuition is that by using highly frequent and highly confident translation pairs the bootstrapping method has more reliable clues that help extract new dimensions in subsequent iterations. On the other hand, low-frequency

pairs (although potentially correct one-to-one translations) do not occur in the corpus and in the contexts of other words frequently enough, and are therefore not sufficient to extract reliable new dimensions of the space.

To test the hypothesis, we again obtain all symmetric translation pairs using *ResponseBC* and then sort them in descending order based on their frequency in the corpus. In total, we retrieve a sorted list of 2031 symmetric translation pairs for ES-EN, and 1689 pairs for IT-EN. Following that, we split the list in 3 parts of equal size: (i) the top third comprises translation pairs with the highest frequency in the corpus (HF-SEED), (ii) the middle third comprises pairs of "medium" frequency (MF-SEED), (iii) the bottom third are low-frequency pairs (LF-SEED). We then use these 3 different seed lexicons of equal size to seed the bootstrapping approach. Fig. 2(a) and 2(b) show the progress of the bootstrapping process using these 3 seed lexicons. We again observe several interesting phenomena:
(i) *High-frequency seed translation pairs are better seeds*, and that finding is in line with our hypothesis. Although the bootstrapping approach again displays a positive trend regardless of the actual choice of seeds (we observe an increase even when using LF-SEED), high-frequency seeds lead to better overall results in the BLE task. Besides its high presence in contexts of other words, another advantage of high-frequency seed pairs is the fact that an initial similarity method will typically acquire more reliable translation candidates for such words (Pekar et al., 2006). For instance, 89.5% of ES-EN pairs in HF-SEED are correct one-to-one translations, compared to 65.1% in MF-SEED, and 44.3% in LF-SEED.
(ii) The difference in results between HF-SEED and MF-SEED is more visible in Acc_1 scores. Although both seed lexicons for all test words provide ranked lists which contain words that exhibit some semantic relation to the given word, the reliability and the frequency of translation pairs are especially important for detecting the relation of cross-lingual word synonymy, that is, the translational equivalence that is exploited in building one-to-one bilingual lexicons.

Exp. III: Does size matter? The following experiment investigates whether bilingual vector spaces may be effectively bootstrapped from small high-quality seed lexicons, and if larger seed lexicons necessarily lead to bilingual vector spaces of higher quality as reflected in BLE results. We again retrieve a sorted list of symmetric translation pairs as in Exp. II. Following that, we build seed lexicons of various sizes by retaining only the first N pairs from the list, where we vary N from 200 to 1400 in steps of 200. We also use the entire sorted list as a seed lexicon (*All*), and compare the results on the BLE task with the results obtained by applying the *ResponseBC* and *TopicBC* methods directly (Vulić and Moens, 2013). The results are summarized in tables 1 and 2. We observe the following:
(i) If we provide a seed lexicon with sufficient entries, the bootstrapping procedure provides comparable results regardless of the seed lexicon size, although results tend to be higher for larger seed lexicons (e.g., compare results when starting with 600 and 1200 lexicon entries). When starting with the size of 600, the bootstrapping approach is able to find dimensions that were already in the seed lexicon of size 1200. The consequence is that, although bootstrapping with a smaller seed lexicon displays a slower start (see the difference in results at iteration 0), the performances level after convergence.
(ii) Regardless of the seed lexicon size, the bootstrapping approach is valuable. It consistently improves the quality of the induced bilingual vector space, and consequently, the quality of bilingual lexicons extracted using that vector space. The positive impact is more prominent for smaller seed lexicons, i.e., we observe an increase of 78% for ES-EN when starting with only 200 seed pairs, compared to an increase of 15% when starting with 800 seed pairs, and 10% when starting with 1400 seed pairs.
(iii) The bootstrapping approach outperforms *ResponseBC* and *TopicBC* in terms of Acc_1 and MRR scores for both language pairs when the seed lexicon provides a sufficient number of entries. However, in terms of Acc_{10}, *TopicBC* and *ResponseBC* still exhibit comparable (for IT-EN) or even better (ES-EN) results. Both *TopicBC* and *ResponseBC* are MuPTM-based methods that, due to MuPTM properties, model the similarity of two words at the level of documents as contexts, while the bootstrapping approach is a window-based approach that narrows down the context to a local neighborhood of a word. The MuPTM-based models are better suited to detect a general *topical* similarity of words, and

Iteration:	0			2			5			10		
Seed lexicon	Acc_1	MRR	Acc_{10}	Acc_1	MRR	Acc_{10}	Acc_1	MRR	Acc_{10}	Acc_1	MRR	Acc_{10}
200(→1617)	0.274	0.352	0.525	0.446	0.534	0.713	0.481	0.569	0.753	0.488	0.576	0.752
400(→1563)	0.416	0.499	0.663	0.518	0.602	0.774	0.542	0.620	0.787	0.545	0.625	0.788
600(→1554)	0.459	0.539	0.707	0.550	0.630	0.787	0.573	0.650	0.803	0.578	0.654	0.802
800(→1582)	0.494	0.572	0.728	0.548	0.631	0.799	0.563	0.644	0.802	0.567	0.646	0.806
1000(→1636)	0.516	0.591	0.744	0.563	0.644	0.805	0.578	0.656	0.813	0.581	0.658	0.817
1200(→1740)	0.536	0.613	0.764	0.586	0.661	0.804	0.588	0.664	0.812	0.591	0.667	0.814
1400(→1888)	0.536	0.620	0.776	0.583	0.659	0.808	0.589	0.666	0.815	0.588	0.666	0.818
All-2031(→2437)	0.543	0.625	0.785	0.589	0.667	0.813	0.597	0.675	0.818	**0.599**	**0.677**	0.820
TopicBC	0.433	0.576	0.843	–	–	–	–	–	–	–	–	–
ResponseBC	0.517	0.635	**0.891**	–	–	–	–	–	–	–	–	–

Table 1: ES-EN: Results with different sizes of the seed lexicon. The number in the parentheses denotes the number of dimensions in the bilingual space after the bootstrapping procedure converges. The seeding method is SEED-RB.

Iteration:	0			2			5			10		
Seed lexicon	Acc_1	MRR	Acc_{10}	Acc_1	MRR	Acc_{10}	Acc_1	MRR	Acc_{10}	Acc_1	MRR	Acc_{10}
200(→1255)	0.394	0.469	0.703	0.515	0.595	0.757	0.548	0.621	0.782	0.555	0.628	0.787
400(→1265)	0.546	0.618	0.757	0.623	0.690	0.831	0.639	0.704	0.840	0.644	0.709	0.844
600(→1309)	0.585	0.657	0.798	0.653	0.718	0.856	0.664	0.726	0.859	**0.672**	0.734	0.862
800(→1365)	0.602	0.672	0.813	0.657	0.723	0.857	0.663	0.726	0.865	0.665	0.730	0.867
1000(→1416)	0.616	0.688	0.828	0.629	0.706	0.853	0.636	0.709	0.857	0.642	0.714	0.861
1200(→1581)	0.628	0.700	0.840	0.655	0.724	0.869	0.664	0.733	0.877	0.668	**0.736**	**0.883**
1400(→1749)	0.626	0.701	0.851	0.654	0.727	0.867	0.656	0.728	0.867	0.661	0.733	0.874
All-1689(→2008)	0.616	0.695	0.850	0.643	0.716	0.860	0.653	0.724	0.862	0.654	0.726	0.866
TopicBC	0.578	0.667	0.834	–	–	–	–	–	–	–	–	–
ResponseBC	0.622	0.729	0.882	–	–	–	–	–	–	–	–	–

Table 2: IT-EN: Results with different sizes of the seed lexicon. The number in the parentheses denotes the number of dimensions in the bilingual space after the bootstrapping procedure converges. The seeding method is SEED-RB.

are therefore not always able to push the real cross-lingual synonyms higher in the ranked list of semantically similar words, while the window-based bootstrapping approach is better tailored to model the relation of cross-lingual synonymy, i.e., to extract one-to-one translation pairs (as reflected in Acc_1 scores). A similar conclusion for monolingual settings is drawn by Baroni and Lenci (2010).

(iv) Since our bootstrapping approach utilizes *ResponseBC* or *TopicBC* as a preprocessing step, it is obvious that the approach leads to an increased complexity. On top of the initial complexity of *ResponseBC* and *TopicBC*, the bootstrapping method requires $|V^S||V^T|$ comparisons at each iteration, but given the fact that each $w_i^S \in V^S$ may be processed independently of any other $w_j^S \in V^S$ in each iteration, the bootstrapping method is trivially parallelizable. That makes the method computationally feasible even for vocabularies larger than the ones reported in the paper.

4.2 Is Confidence Estimation Important?

According to the results from tables 1 and 2, regardless of the seed lexicon size, the bootstrapping approach does not suffer from semantic drift, i.e., if we seed the process with high-quality symmetric translation pairs, it is able to recover more pairs and add them as new dimensions of the bilingual vector space. However, we also study the influence of applying different confidence estimation functions on top of the symmetry constraint (see sect 2.3), but we do not observe any improvement in the BLE results, regardless of the actual choice of a confidence estimation function. The only observed phenomenon, as illustrated by fig. 3, is the *slower convergence rate* when setting the parameter B to lower values.

The symmetry constraint alone seems to be sufficient to prevent semantic drift, but it might also be a too strong and a too conservative assumption, since only a small portion of all possible translation pairs is used to span the bilingual vector space (for instance,

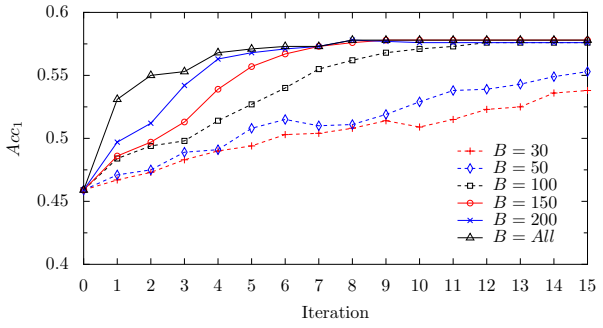


Figure 3: The effect of learning rate B on bootstrapping. Language pair: ES-EN, seed lexicon: SEED-RB with 600 pairs, confidence function: symmetrized M-Best.

when starting with 600 entries for ES-EN, the final bilingual vector space consists of only 1554 pairs, while the total number of ES nouns is 9439). One line of future work will address the construction of bootstrapping algorithms that also enable the usage of highly reliable asymmetric pairs as dimensions, and the confidence estimation functions might have a more important role in that setting.

5 Conclusion

We have presented a new bootstrapping approach to inducing bilingual vector spaces from non-parallel data, and have shown the utility of the induced space in the BLE task. The approach is fully corpus-based and, unlike previous work, it does not rely on the availability of machine-readable translation dictionaries or predefined concept categories. We have systematically described, analyzed and evaluated all key components of the bootstrapping approach. Results reveal that, contrary to conclusions from prior work, the initialization of the bilingual vector space is especially important. We have presented a novel approach to initializing the bootstrapping procedure, and have shown that better results in the BLE task are obtained by starting from seed lexicons that comprise highly reliable high-frequent translation pairs. The bootstrapping framework presented in the paper is completely language pair independent, which makes it effectively applicable to any language pair.

In future work, we will investigate other models of similarity besides *TopicBC* and *ResponseBC* (e.g, the method from (Haghighi et al., 2008)) that could be used as preliminary models for constructing an initial bilingual vector space. Furthermore, we plan to study other confidence functions and explore if asymmetric translation candidates could also contribute to the bootstrapping method. Finally, we also plan to test the robustness of our fully corpus-based bootstrapping approach by porting it to more language pairs.

Acknowledgments

We would like to thank the anonymous reviewers for their useful suggestions. This research has been carried out in the framework of the TermWise Knowledge Platform (IOF-KP/09/001) funded by the Industrial Research Fund, KU Leuven, Belgium.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of UAI*, pages 75–82.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526.

Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL-HLT*, pages 600–609.

Hal Daumé III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of ACL-HLT*, pages 407–412.

Wim De Smet and Marie-Francine Moens. 2009. Cross-language linking of news stories on the Web using interlingual topic modeling. In *Proceedings of the CIKM 2009 Workshop on Social Web Search and Mining*, pages 57–64.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of EMNLP-CoNLL*, pages 1–11.

Darja Fišer and Nikola Ljubešić. 2011. Bilingual lexicon extraction from comparable corpora for closely related languages. In *Proceedings of RANLP*, pages 125–131.

Pascale Fung and Percy Cheung. 2004. Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and EM. In *Proceedings of EMNLP*, pages 57–63.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of COLING*, pages 414–420.

Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of ACL*, pages 526–533.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, pages 771–779.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of EACL*, pages 286–295.

Azniah Ismail and Suresh Manandhar. 2010. Bilingual lexicon extraction from comparable corpora using in-domain terms. In *Proceedings of COLING*, pages 481–489.

Jun'ichi Kazama, Stijn De Saeger, Kow Kuroda, Masaki Murata, and Kentaro Torisawa. 2010. A Bayesian method for robust estimation of distributional similarities. In *Proceedings of ACL*, pages 247–256.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition*, pages 9–16.

Zornitsa Kozareva and Eduard H. Hovy. 2010. Not all seeds are equal: Measuring the quality of text mining seeds. In *Proceedings of NAACL-HLT*, pages 618–626.

Audrey Laroche and Philippe Langlais. 2010. Revisiting context-based projection methods for term-translation spotting in comparable corpora. In *Proceedings of COLING*, pages 617–625.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of SIGIR*, pages 175–182.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of ACL*, pages 25–32.

Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information Processing and Management*, 41(3):523–547.

Tara McIntosh and James R. Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of ACL*, pages 396–404.

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

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of EMNLP*, pages 880–889.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from Wikipedia. In *Proceedings of WWW*, pages 1155–1156.

Yoshiki Niwa and Yoshihiko Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of COLING*, pages 304–309.

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

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of KDD*, pages 613–619.

Yves Peirsman and Sebastian Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *Proceedings of NAACL-HLT*, pages 921–929.

Yves Peirsman and Sebastian Padó. 2011. Semantic relations in bilingual lexicons. *ACM Transactions on Speech and Language Processing*, 8(2):article 3.

Viktor Pekar, Ruslan Mitkov, Dimitar Blagoev, and Andrea Mulloni. 2006. Finding translations for low-frequency words in comparable corpora. *Machine Translation*, 20(4):247–266.

Emmanuel Prochasson and Pascale Fung. 2011. Rare word translation extraction from aligned comparable documents. In *Proceedings of ACL*, pages 1327–1335.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of ACL*, pages 519–526.

Ellen Riloff and Jessica Shepherd. 1999. A corpus-based bootstrapping algorithm for semi-automated semantic lexicon construction. *Natural Language Engineering*, 5(2):147–156.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*.

Daphna Shezaf and Ari Rappoport. 2010. Bilingual lexicon generation using non-aligned signatures. In *Proceedings of ACL*, pages 98–107.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of EMNLP-CoNLL*, pages 667–677.

Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 427(7):424–440.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013a. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of ACL*, 1:1–12.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013b. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL-HLT*, pages 1061–1071.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *Proceedings of EMNLP-CoNLL*, pages 24–36.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of EMNLP*, pages 214–221.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of EMNLP-CoNLL*, pages 1324–1334.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artifical Intelligence Research*, 37(1):141–188.

Lonneke van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of ACL-HLT*, pages 299–304.

Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In *Proceedings of TREC*, pages 77–82.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of NAACL-HLT*, pages 106–116.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying word translations from comparable corpora using latent topic models. In *Proceedings of ACL-HLT*, pages 479–484.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*, pages 200–207.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction. In *Proceedings of ACL*, pages 1128–1137.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL*, pages 55–63.

Deriving adjectival scales from continuous space word representations

Joo-Kyung Kim
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210, USA
kimjook@cse.ohio-state.edu

Marie-Catherine de Marneffe
Department of Linguistics
The Ohio State University
Columbus, OH 43210, USA
mcdm@ling.ohio-state.edu

Abstract

Continuous space word representations extracted from neural network language models have been used effectively for natural language processing, but until recently it was not clear whether the spatial relationships of such representations were interpretable. Mikolov et al. (2013) show that these representations do capture syntactic and semantic regularities. Here, we push the interpretation of continuous space word representations further by demonstrating that vector offsets can be used to derive adjectival scales (e.g., *okay* $<$ *good* $<$ *excellent*). We evaluate the scales on the indirect answers to *yes/no* questions corpus (de Marneffe et al., 2010). We obtain 72.8% accuracy, which outperforms previous results ($\sim$60%) on this corpus and highlights the quality of the scales extracted, providing further support that the continuous space word representations are meaningful.

1 Introduction

There has recently been a surge of interest for deep learning in natural language processing. In particular, neural network language models (NNLMs) have been used to learn distributional word vectors (Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010): the models jointly learn an embedding of words into an n-dimensional feature space. One of the advantages put forth for such distributed representations compared to traditional n-gram models is that similar words are likely to have similar vector representations in a continuous space model, whereas the discrete units of an n-gram model do not exhibit any inherent relation with one another. It has been shown that the continuous space representations improve performance in a variety of NLP tasks, such as POS tagging, semantic role labeling, named entity resolution, parsing (Collobert and Weston, 2008; Turian et al., 2010; Huang et al., 2012).

Mikolov et al. (2013) show that there are some syntactic and semantic regularities in the word representations learned, such as the singular/plural relation (the difference of singular and plural word vectors are equivalent: *apple* $-$ *apples* $\approx$ *car* $-$ *cars* $\approx$ *family* $-$ *families*) or the gender relation (a masculine noun can be transformed into the feminine form: *king* $-$ *man* $+$ *woman* $\approx$ *queen*).

We extend Mikolov et al. (2013)'s approach and explore further the interpretation of the vector space. We show that the word vectors learned by NNLMs are meaningful: we can extract scalar relationships between adjectives (e.g., *bad* $<$ *okay* $<$ *good* $<$ *excellent*), which can not only serve to build a sentiment lexicon but also be used for inference. To evaluate the quality of the scalar relationships learned by NNLMs, we use the indirect *yes/no* question answer pairs (IQAP) from (de Marneffe et al., 2010), where scales between adjectives are needed to infer a *yes/no* answer from a reply without explicit *yes* or *no* such as *Was the movie good? It was excellent.* Our method reaches 72.8% accuracy, which is the best result reported so far when scales are used.

2 Previous work

We use the continuous word representations from (Mikolov et al., 2011), extracted from a recurrent neural network language model (RNNLM), whose

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1625–1630,
Seattle, Washington, USA, 18-21 October 2013.

three-layer architecture is represented in Figure 1.

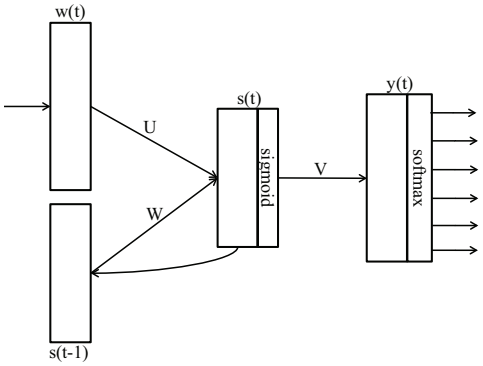


Figure 1: The architecture of the RNNLM.

In the input layer, $w(t)$ is the input word represented by 1-of-N coding at time t when the vocabulary size is N. When there are M nodes in the hidden layer, the number of connections between the input layer and the hidden layer is NM and the connections can be represented by a matrix U.

The hidden layer is also connected recurrently to the context $s(t-1)$ at time $t-1$ ($s(0)$ is initialized with small values like 0.1). The connections between the previous context and the hidden layer are represented by a matrix W. The dimensionality of the word representations is controlled by the size of W. The output of the hidden layer is $s(t) = f(Uw(t) + Ws(t-1))$, where f is a sigmoid function.

Because the inputs of the hidden layer consist of the word $w(t)$ and the previous hidden layer output $s(t-1)$, the current context of the RNN is influenced by the current word and the previous context. Therefore, we can regard that the continuous representations from the RNNLM exploit the context implicitly considering the word sequence information (Mikolov et al., 2010).

V is a N by M matrix representing the connections between the hidden layer and the output layer. The final output is $y(t) = g(Vs(t))$, where g is a softmax function to represent the probability distribution over all the words in the vocabulary.

When the RNN is trained by the back propagation algorithm, we can regard the ith column vector of U as the continuous representation of the ith word in the vocabulary since the column was adjusted correspondingly to the ith element of $w(t)$. Because the $s(t)$ outputs of two input words will be similar when they have similar $s(t-1)$ values, the corresponding column vectors of the words will also be similar.

Mikolov et al. (2013) showed that constant vector offsets of word pairs can represent linguistic regularities. Let w_a and w_b denote the vectors for the words a and b, respectively. Then the vector offset of the word pair is $w_a - w_b$. If a and b are syntactically or semantically related, the vector offset can be interpreted as a transformation of the syntactic form or the meaning. The offset can also be added to another word vector c. The word vector nearest to $w_a - w_b + w_c$ would be related to word c with the syntactic or semantic difference as the difference between a and b, as it is the case for the *king*, *man*, and *woman* example, where *king* $-$ *man* $+$ *woman* would approximately represent *king* with feminine gender (i.e., *queen*). They also tried to use the continuous representations generated by Latent Semantic Analysis (LSA) (Landauer et al., 1998). However, the results using LSA were worse because LSA is a bag-of-words model, in which it is difficult to exploit word sequence information as the context.

For all the experiments in this paper, we use the precomputed word representations generated by the RNNLM from (Mikolov et al., 2013). Their RNN is trained with 320M words from the Broadcast News data (the vocabulary size is 82,390 words), and we used word vectors with a dimensionality of 1,600 (the highest dimensionality provided).[1] We standardized the dataset so that the mean and the variance of the representations are 0 and 1, respectively.[2]

3 Deriving adjectival scales

Here we explore further the interpretation of word vectors. Assuming that the transformation of form or meaning represented by the vector offset is linear, an intermediate vector between two word vectors would represent some "middle" form or meaning. For example, given the positive and superlative forms of an adjective (e.g., *good* and *best*), we expect that the word representation in the middle of

[1] We also experimented with smaller dimensions, but consistent with the analyses in (Mikolov et al., 2013), the highest dimensionality gave better results.

[2] http://www.fit.vutbr.cz/~imikolov/rnnlm/word_projections-1600.txt.gz

Input words	Words with highest cosine similarities to the mean vector			
good:best	**better: 0.738**	strong: 0.644	normal: 0.619	less: 0.609
bad:worst	terrible: 0.726	great: 0.678	horrible: 0.674	**worse: 0.665**
slow:slowest	**slower: 0.637**	sluggish: 0.614	steady: 0.558	brisk: 0.543
fast:fastest	**faster: 0.645**	slower: 0.602	quicker: 0.542	harder: 0.518

Table 1: Words with corresponding vectors closest to the mean of positive:superlative word vectors.

First word (-)		1st quarter		Half		3rd quarter		Second word (+)	
furious	1	angry	0.632	unhappy	0.640	pleased	0.516	**happy**	1
furious	1	angry	0.615	tense	0.465	quiet	0.560	**calm**	1
terrible	1	horrible	0.783	incredible	0.714	wonderful	0.772	**terrific**	1
cold	1	mild	0.348	warm	0.517	sticky	0.424	**hot**	1
ugly	1	nasty	0.672	wacky	0.645	lovely	0.715	**gorgeous**	1

Table 2: Adjectival scales extracted from the RNN: each row represent a scale, and for each intermediate point the closest word in term of cosine similarity is given.

them will correspond to the comparative form (i.e., *better*). To extract the "middle" word between two word vectors w_a and w_b, we take the vector offset $w_a - w_b$ divided by 2, and add w_b: $w_b + (w_a - w_b)/2$. The result corresponds to the midpoint between the two words. Then, we find the word whose cosine similarity to the midpoint is the highest.

Table 1 gives some positive:superlative pairs and the top four closest words to the mean vectors, where the distance metric is the cosine similarity. The correct comparative forms (in bold) are quite close to the mean vector of the positive and superlative form vectors, highlighting the fact that there is some meaningful interpretation of the vector space: the word vectors are constituting a scale.

We can extend this idea of extracting an ordering between two words. For any two semantically related adjectives, intermediate vectors extracted along the line connecting the first and second word vectors should exhibit scalar properties, as seen above for the positive-comparative-superlative triplets. If we take two antonyms (*furious* and *happy*), words extracted at the intermediate points x_1, x_2 and x_3 should correspond to words lying on a scale of happiness (from "less furious" to "more happy"), as illustrated in Figure 2. Table 2 gives some adjectival scales that we extracted from the continuous word space, using antonym pairs. We picked three points with equal intervals on the line from the first to the second word (1st quarter, half and 3rd quarter). The extracted scales look quite reasonable: the words form a continuum from more negative to more positive meanings.

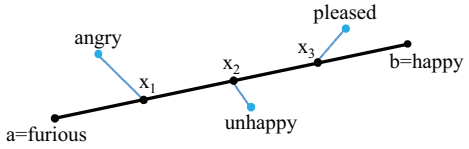


Figure 2: An example of vectors with the highest cosine similarity to intermediate points on the line between *furious* and *happy*.

Tables 1 and 2 demonstrate that the word vector space is interpretable: intermediate vectors between two word vectors represent a semantic continuum.

4 Evaluation: Indirect answers to *yes/no* questions

To evaluate the quality of the adjective scales learned by the neural network approach, we use the corpus of indirect answers to *yes/no* questions created by (de Marneffe et al., 2010), which consists of question-answer pairs involving gradable modifiers to test scalar implicatures. We focus on the 125 pairs in the corpus where both the question and answer contain an adjective: e.g., *Is Obama qualified? I think he's young.*[3] Each question-answer pair has

[3]These 125 pairs correspond to the 'Other adjective' category in (de Marneffe et al., 2010).

been annotated via Mechanical Turk for whether the answer conveys *yes*, *no* or *uncertain*.

4.1 Method

The previous section showed that we can draw a line passing through an adjective and its antonym and that the words extracted along the line are roughly semantically ordered. To infer a *yes* or *no* answer in the case of the IQAP corpus, we use the following approach illustrated with the *Obama* example above (Figure 3). Using WordNet 3.1 (Fellbaum, 1998), we look for an antonym of the adjective in the question *qualified*: *unqualified* is retrieved. Since the scales extracted are only roughly ordered, to infer *yes* when the question and answer words are very close, we set the decision boundary perpendicular to the line connecting the two words and passing through the midpoint of the line.

Since the answer word is *young*, we check whether *young* is in the area including *qualified* or in the other area. We infer a *yes* answer in the former case, and a *no* answer in the latter case. If *young* is on the boundary, we infer *uncertain*. If a sentence contains a negation (e.g., *Are you stressed? I am not peaceful.*), we compute the scale for *stressed-peaceful* and then reverse the answer obtained, similarly to what is done in (de Marneffe et al., 2010).

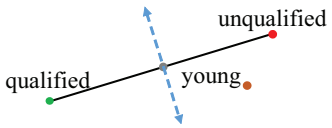


Figure 3: An example of the decision boundary given *qualified* as the question and *young* as the answer.

Since a word can have multiple senses and different antonyms for the senses, it is important to select the most appropriate antonym to build a more accurate decision boundary. We consider all antonyms across senses[4] and select the antonym that is most collinear with the question and the answer. For the word vectors of the question w_q, the ith antonym w_{ant_i}, and the answer w_a, we select ant_i where $argmax_{ant_i}|cos(w_q - w_a, w_q - w_{ant_i})|$. Figure 4 schematically shows antonym selection when the question is *good* and the answer is *excellent*: *bad* and *evil* are the antonym candidates of *good*. Because the absolute cosine similarity of *good-excellent* to *good-bad* is higher than to *good-evil*, we choose *bad* as the antonym in this case.

[4]Antonyms in WordNet can be directly opposed to a given word or indirectly opposed via other words. When there are direct antonyms for the question word, we only consider those.

	Acc	Macro P	Macro R	Macro F1
de Marneffe (2010)	60.00	59.72	59.40	59.56
Mohtarami (2011)	–	62.23	60.88	61.55
RNN model	**72.80**	**69.78**	**71.39**	**70.58**

Table 3: Score (%) comparison on the 125 scalar adjective pairs in the IQAP corpus.

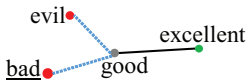


Figure 4: An example of antonym selection.

4.2 Results and discussion

Table 3 compares our results with previous ones where adjectival scales are considered: de Marneffe et al. (2010) propose an unsupervised approach where scales are learned from distributional information in a Web corpus; Mohtarami et al. (2011)'s model is similar to ours but uses word representations obtained by LSA and a word sense disambiguation system (Zhong and Ng, 2010) to choose antonyms. To compare with Mohtarami et al. (2011), we use macro-averaged precision and recall for *yes* and *no*. For the given metrics, our model significantly outperforms the previous ones ($p < 0.05$, McNemar's test).

Mohtarami et al. (2011) present higher numbers obtained by replacing the answer words with their synonyms in WordNet. However, that approach fails to capture orderings. Two words of different degree are often regarded as synonyms: even though *furious* means extremely angry, *furious* and *angry* are synonyms in WordNet. Therefore using synonyms, the system will output the same answer irrespective of the order in the pair. Mohtarami et al. (2012) also presented results on the interpretation of indirect questions on the IQAP corpus, but their method

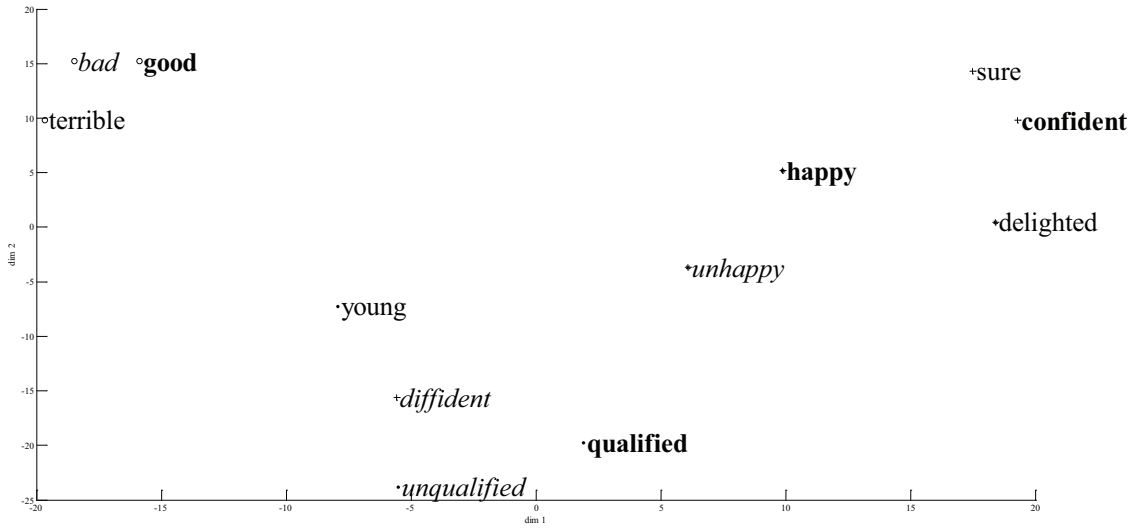


Figure 5: Question words (bold), their antonyms (italic), and answer words (normal) of four pairs from the IQAP dataset. The words are visualized by MDS.

did not involve learning or using scalar implicatures.

Figure 5 gives a qualitative picture: the question words, antonyms and answer words for four of the IQAP pairs are visualized in 2D space by multi-dimensional scaling (MDS). Note that MDS introduces some distortion in the lower dimensions. Bullet markers correspond to words in the same pair. Question words, antonyms, and answer words are displayed by bold, italic, and normal fonts, respectively. In the *Obama* example previously mentioned (*Is Obama qualified? I think he's young.*), the question word is *qualified* and the answer word is *young*. In Figure 5, *qualified* is around (2,-20) while its antonym *unqualified* is around (-6,-24). Since *young* is around (-7,-8), we infer that *young* is semantically closer to *unqualified* which corroborates with the Turkers' intuitions in this case. (1), (2) and (3) give the other examples displayed in Figure 5.

(1) A: Do you think she'd be *happy* with this book?
B: I think she'd be *delighted* by it.

(2) A: Do you think that's a *good* idea?
B: It's a *terrible* idea.

(3) A: The president is promising support for Americans who have suffered from this hurricane. Are you *confident* you are going to be getting that?
B: I'm not so *sure* about my insurance company.

In (1), *delighted* is stronger than *happy*, leading to a *yes* answer, whereas in (2), *terrible* is weaker than *good* leading to a *no* answer. In (3), the presence of a negation will reverse the answer inferred, leading to *no*.

5 Conclusion

In this paper we give further evidence that the relationships in the continuous vector space learned by recurrent neural network models are interpretable. We show that using vector offsets, we can successfully learn adjectival scales, which are useful for scalar implicatures, as demonstrated by the high results we obtain on the IQAP corpus.

Acknowledgements

We thank Eric Fosler-Lussier and the anonymous reviewers for their helpful comments on previous versions of this paper.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2010. Was it good? It was provocative. Learning the meaning of scalar adjectives. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*, pages 167–176.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT Press.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 873–882. Association for Computational Linguistics.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*, pages 1045–1048.

Tomas Mikolov, Daniel Povey, Lukáš Burget, and Jan Cernocky. 2011. Strategies for training large scale neural network language models. In *Proceedings of ASRU*, pages 196–201.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751.

Mitra Mohtarami, Hadi Amiri, Man Lan, and Chew Lim Tan. 2011. Predicting the uncertainty of sentiment adjectives in indirect answers. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2485–2488.

Mitra Mohtarami, Hadi Amiri, Man Lan, Thanh Phu Tran, and Chew Lim Tan. 2012. Sense sentiment similarity: an analysis. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1706–1712.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: a wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83.

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang,
Christopher D. Manning, Andrew Y. Ng and Christopher Potts
Stanford University, Stanford, CA 94305, USA
richard@socher.org,{aperelyg,jcchuang,ang}@cs.stanford.edu
{jeaneis,manning,cgpotts}@stanford.edu

Abstract

Semantic word spaces have been very useful but cannot express the meaning of longer phrases in a principled way. Further progress towards understanding compositionality in tasks such as sentiment detection requires richer supervised training and evaluation resources and more powerful models of composition. To remedy this, we introduce a Sentiment Treebank. It includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences and presents new challenges for sentiment compositionality. To address them, we introduce the Recursive Neural Tensor Network. When trained on the new treebank, this model outperforms all previous methods on several metrics. It pushes the state of the art in single sentence positive/negative classification from 80% up to 85.4%. The accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features baselines. Lastly, it is the only model that can accurately capture the effects of negation and its scope at various tree levels for both positive and negative phrases.

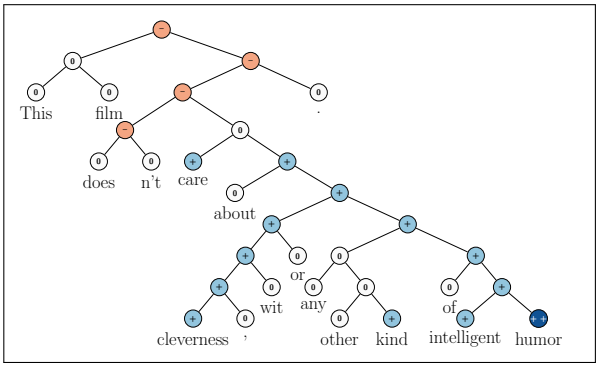


Figure 1: Example of the Recursive Neural Tensor Network accurately predicting 5 sentiment classes, very negative to very positive (– –, –, 0, +, + +), at every node of a parse tree and capturing the negation and its scope in this sentence.

1 Introduction

Semantic vector spaces for single words have been widely used as features (Turney and Pantel, 2010). Because they cannot capture the meaning of longer phrases properly, compositionality in semantic vector spaces has recently received a lot of attention (Mitchell and Lapata, 2010; Socher et al., 2010; Zanzotto et al., 2010; Yessenalina and Cardie, 2011; Socher et al., 2012; Grefenstette et al., 2013). However, progress is held back by the current lack of large and labeled compositionality resources and models to accurately capture the underlying phenomena presented in such data. To address this need, we introduce the Stanford Sentiment Treebank and a powerful Recursive Neural Tensor Network that can accurately predict the compositional semantic effects present in this new corpus.

The *Stanford Sentiment Treebank* is the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus is based on the dataset introduced by Pang and Lee (2005) and consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser (Klein and Manning, 2003) and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. This new dataset allows us to analyze the intricacies of sentiment and to capture complex linguistic phenomena. Fig. 1 shows one of the many examples with clear compositional structure. The granularity and size of

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642,
Seattle, Washington, USA, 18-21 October 2013.

this dataset will enable the community to train compositional models that are based on supervised and structured machine learning techniques. While there are several datasets with document and chunk labels available, there is a need to better capture sentiment from short comments, such as Twitter data, which provide less overall signal per document.

In order to capture the compositional effects with higher accuracy, we propose a new model called the Recursive Neural Tensor Network (RNTN). Recursive Neural Tensor Networks take as input phrases of any length. They represent a phrase through word vectors and a parse tree and then compute vectors for higher nodes in the tree using the same tensor-based composition function. We compare to several supervised, compositional models such as standard recursive neural networks (RNN) (Socher et al., 2011b), matrix-vector RNNs (Socher et al., 2012), and baselines such as neural networks that ignore word order, Naive Bayes (NB), bi-gram NB and SVM. All models get a significant boost when trained with the new dataset but the RNTN obtains the highest performance with 80.7% accuracy when predicting fine-grained sentiment for all nodes. Lastly, we use a test set of positive and negative sentences and their respective negations to show that, unlike bag of words models, the RNTN accurately captures the sentiment change and scope of negation. RNTNs also learn that sentiment of phrases following the contrastive conjunction 'but' dominates.

The complete training and testing code, a live demo and the Stanford Sentiment Treebank dataset are available at `http://nlp.stanford.edu/sentiment`.

2 Related Work

This work is connected to five different areas of NLP research, each with their own large amount of related work to which we cannot do full justice given space constraints.

Semantic Vector Spaces. The dominant approach in semantic vector spaces uses distributional similarities of single words. Often, co-occurrence statistics of a word and its context are used to describe each word (Turney and Pantel, 2010; Baroni and Lenci, 2010), such as tf-idf. Variants of this idea use more complex frequencies such as how often a word appears in a certain syntactic context (Pado and Lapata, 2007; Erk and Padó, 2008). However, distributional vectors often do not properly capture the differences in antonyms since those often have similar contexts. One possibility to remedy this is to use neural word vectors (Bengio et al., 2003). These vectors can be trained in an unsupervised fashion to capture distributional similarities (Collobert and Weston, 2008; Huang et al., 2012) but then also be fine-tuned and trained to specific tasks such as sentiment detection (Socher et al., 2011b). The models in this paper can use purely supervised word representations learned entirely on the new corpus.

Compositionality in Vector Spaces. Most of the compositionality algorithms and related datasets capture two word compositions. Mitchell and Lapata (2010) use e.g. two-word phrases and analyze similarities computed by vector addition, multiplication and others. Some related models such as holographic reduced representations (Plate, 1995), quantum logic (Widdows, 2008), discrete-continuous models (Clark and Pulman, 2007) and the recent compositional matrix space model (Rudolph and Giesbrecht, 2010) have not been experimentally validated on larger corpora. Yessenalina and Cardie (2011) compute matrix representations for longer phrases and define composition as matrix multiplication, and also evaluate on sentiment. Grefenstette and Sadrzadeh (2011) analyze subject-verb-object triplets and find a matrix-based categorical model to correlate well with human judgments. We compare to the recent line of work on supervised compositional models. In particular we will describe and experimentally compare our new RNTN model to recursive neural networks (RNN) (Socher et al., 2011b) and matrix-vector RNNs (Socher et al., 2012) both of which have been applied to bag of words sentiment corpora.

Logical Form. A related field that tackles compositionality from a very different angle is that of trying to map sentences to logical form (Zettlemoyer and Collins, 2005). While these models are highly interesting and work well in closed domains and on discrete sets, they could only capture sentiment distributions using separate mechanisms beyond the currently used logical forms.

Deep Learning. Apart from the above mentioned

work on RNNs, several compositionality ideas related to neural networks have been discussed by Bottou (2011) and Hinton (1990) and first models such as Recursive Auto-associative memories been experimented with by Pollack (1990). The idea to relate inputs through three way interactions, parameterized by a tensor have been proposed for relation classification (Sutskever et al., 2009; Jenatton et al., 2012), extending Restricted Boltzmann machines (Ranzato and Hinton, 2010) and as a special layer for speech recognition (Yu et al., 2012).

Sentiment Analysis. Apart from the above-mentioned work, most approaches in sentiment analysis use bag of words representations (Pang and Lee, 2008). Snyder and Barzilay (2007) analyzed larger reviews in more detail by analyzing the sentiment of multiple aspects of restaurants, such as food or atmosphere. Several works have explored sentiment compositionality through careful engineering of features or polarity shifting rules on syntactic structures (Polanyi and Zaenen, 2006; Moilanen and Pulman, 2007; Rentoumi et al., 2010; Nakagawa et al., 2010).

3 Stanford Sentiment Treebank

Bag of words classifiers can work well in longer documents by relying on a few words with strong sentiment like 'awesome' or 'exhilarating.' However, sentiment accuracies even for binary positive/negative classification for single sentences has not exceeded 80% for several years. For the more difficult multiclass case including a neutral class, accuracy is often below 60% for short messages on Twitter (Wang et al., 2012). From a linguistic or cognitive standpoint, ignoring word order in the treatment of a semantic task is not plausible, and, as we will show, it cannot accurately classify hard examples of negation. Correctly predicting these hard cases is necessary to further improve performance.

In this section we will introduce and provide some analyses for the new *Sentiment Treebank* which includes labels for every syntactically plausible phrase in thousands of sentences, allowing us to train and evaluate compositional models.

We consider the corpus of movie review excerpts from the `rottentomatoes.com` website originally collected and published by Pang and Lee (2005). The original dataset includes 10,662 sentences, half of which were considered positive and the other half negative. Each label is extracted from a longer movie review and reflects the writer's overall intention for this review. The normalized, lowercased text is first used to recover, from the original website, the text with capitalization. Remaining HTML tags and sentences that are not in English are deleted. The Stanford Parser (Klein and Manning, 2003) is used to parses all 10,662 sentences. In approximately 1,100 cases it splits the snippet into multiple sentences. We then used Amazon Mechanical Turk to label the resulting 215,154 phrases. Fig. 3 shows the interface annotators saw. The slider has 25 different values and is initially set to neutral. The phrases in each hit are randomly sampled from the set of all phrases in order to prevent labels being influenced by what follows. For more details on the dataset collection, see supplementary material.

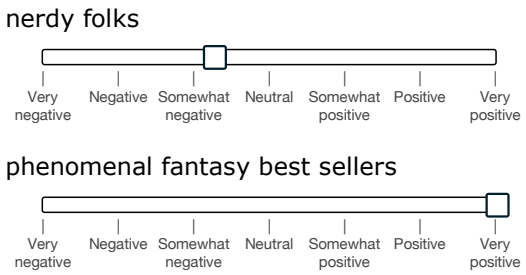


Figure 3: The labeling interface. Random phrases were shown and annotators had a slider for selecting the sentiment and its degree.

Fig. 2 shows the normalized label distributions at each n-gram length. Starting at length 20, the majority are full sentences. One of the findings from labeling sentences based on *reader's perception* is that many of them could be considered neutral. We also notice that stronger sentiment often builds up in longer phrases and the majority of the shorter phrases are neutral. Another observation is that most annotators moved the slider to one of the five positions: negative, somewhat negative, neutral, positive or somewhat positive. The extreme values were rarely used and the slider was not often left in between the ticks. Hence, *even a 5-class classification into these categories captures the main variability of the labels*. We will name this *fine-grained sentiment* classification and our main experiment will be to recover these five labels for phrases of all lengths.

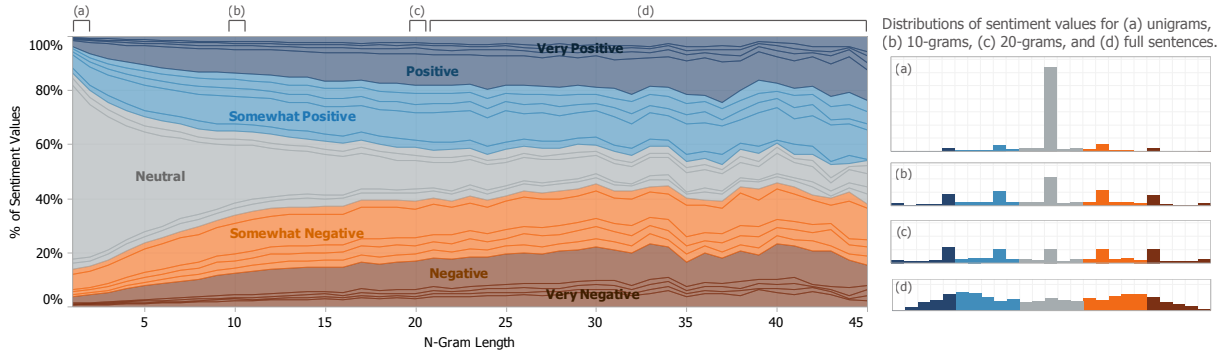


Figure 2: Normalized histogram of sentiment annotations at each n-gram length. Many shorter n-grams are neutral; longer phrases are well distributed. Few annotators used slider positions between ticks or the extreme values. Hence the two strongest labels and intermediate tick positions are merged into 5 classes.

4 Recursive Neural Models

The models in this section compute compositional vector representations for phrases of variable length and syntactic type. These representations will then be used as features to classify each phrase. Fig. 4 displays this approach. When an n-gram is given to the compositional models, it is parsed into a binary tree and each leaf node, corresponding to a word, is represented as a vector. Recursive neural models will then compute parent vectors in a bottom up fashion using different types of compositionality functions g. The parent vectors are again given as features to a classifier. For ease of exposition, we will use the tri-gram in this figure to explain all models.

We first describe the operations that the below recursive neural models have in common: word vector representations and classification. This is followed by descriptions of two previous RNN models and our RNTN.

Each word is represented as a d-dimensional vector. We initialize all word vectors by randomly sampling each value from a uniform distribution: $\mathcal{U}(-r, r)$, where $r = 0.0001$. All the word vectors are stacked in the word embedding matrix $L \in \mathbb{R}^{d \times |V|}$, where $|V|$ is the size of the vocabulary. Initially the word vectors will be random but the L matrix is seen as a parameter that is trained jointly with the compositionality models.

We can use the word vectors immediately as parameters to optimize and as feature inputs to a softmax classifier. For classification into five classes, we compute the posterior probability over

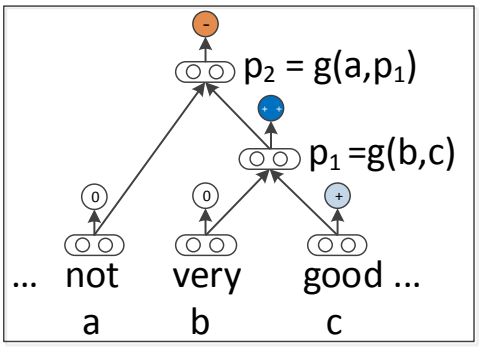


Figure 4: Approach of Recursive Neural Network models for sentiment: Compute parent vectors in a bottom up fashion using a compositionality function g and use node vectors as features for a classifier at that node. This function varies for the different models.

labels given the word vector via:

$$y^a = \text{softmax}(W_s a), \tag{1}$$

where $W_s \in \mathbb{R}^{5 \times d}$ is the sentiment classification matrix. For the given tri-gram, this is repeated for vectors b and c. The main task of and difference between the models will be to compute the hidden vectors $p_i \in \mathbb{R}^d$ in a bottom up fashion.

4.1 RNN: Recursive Neural Network

The simplest member of this family of neural network models is the standard recursive neural network (Goller and Küchler, 1996; Socher et al., 2011a). First, it is determined which parent already has all its children computed. In the above tree example, p_1 has its two children's vectors since both are words. RNNs use the following equations to compute the parent vectors:

$$p_1 = f\left(W \begin{bmatrix} b \\ c \end{bmatrix}\right), p_2 = f\left(W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right),$$

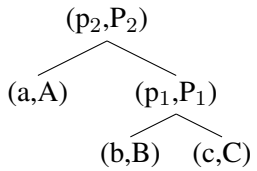
where $f = \tanh$ is a standard element-wise nonlinearity, $W \in \mathbb{R}^{d \times 2d}$ is the main parameter to learn and we omit the bias for simplicity. The bias can be added as an extra column to W if an additional 1 is added to the concatenation of the input vectors. The parent vectors must be of the same dimensionality to be recursively compatible and be used as input to the next composition. Each parent vector p_i, is given to the same $\mathrm{softmax}$ classifier of Eq. 1 to compute its label probabilities.

This model uses the same compositionality function as the recursive autoencoder (Socher et al., 2011b) and recursive auto-associate memories (Pollack, 1990). The only difference to the former model is that we fix the tree structures and ignore the reconstruction loss. In initial experiments, we found that with the additional amount of training data, the reconstruction loss at each node is not necessary to obtain high performance.

4.2 MV-RNN: Matrix-Vector RNN

The MV-RNN is linguistically motivated in that most of the parameters are associated with words and each composition function that computes vectors for longer phrases depends on the actual words being combined. The main idea of the MV-RNN (Socher et al., 2012) is to represent every word and longer phrase in a parse tree as both a vector and a matrix. When two constituents are combined the matrix of one is multiplied with the vector of the other and vice versa. Hence, the compositional function is parameterized by the words that participate in it.

Each word's matrix is initialized as a $d \times d$ identity matrix, plus a small amount of Gaussian noise. Similar to the random word vectors, the parameters of these matrices will be trained to minimize the classification error at each node. For this model, each n-gram is represented as a list of (vector,matrix) pairs, together with the parse tree. For the tree with (vector,matrix) nodes:



the MV-RNN computes the first parent vector and its matrix via two equations:

$$p_1 = f\left(W \begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M \begin{bmatrix} B \\ C \end{bmatrix}\right),$$

where $W_M \in \mathbb{R}^{d \times 2d}$ and the result is again a $d \times d$ matrix. Similarly, the second parent node is computed using the previously computed (vector,matrix) pair (p_1, P_1) as well as (a, A). The vectors are used for classifying each phrase using the same $\mathrm{softmax}$ classifier as in Eq. 1.

4.3 RNTN:Recursive Neural Tensor Network

One problem with the MV-RNN is that the number of parameters becomes very large and depends on the size of the vocabulary. It would be cognitively more plausible if there was a single powerful composition function with a fixed number of parameters. The standard RNN is a good candidate for such a function. However, in the standard RNN, the input vectors only implicitly interact through the nonlinearity (squashing) function. A more direct, possibly multiplicative, interaction would allow the model to have greater interactions between the input vectors.

Motivated by these ideas we ask the question: Can a single, more powerful composition function perform better and compose aggregate meaning from smaller constituents more accurately than many input specific ones? In order to answer this question, we propose a new model called the Recursive Neural Tensor Network (RNTN). The main idea is to use the same, tensor-based composition function for all nodes.

Fig. 5 shows a single tensor layer. We define the output of a tensor product $h \in \mathbb{R}^d$ via the following vectorized notation and the equivalent but more detailed notation for each slice $V^{[i]} \in \mathbb{R}^{d \times d}$:

$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]} \begin{bmatrix} b \\ c \end{bmatrix}.$$

where $V^{[1:d]} \in \mathbb{R}^{2d \times 2d \times d}$ is the tensor that defines multiple bilinear forms.

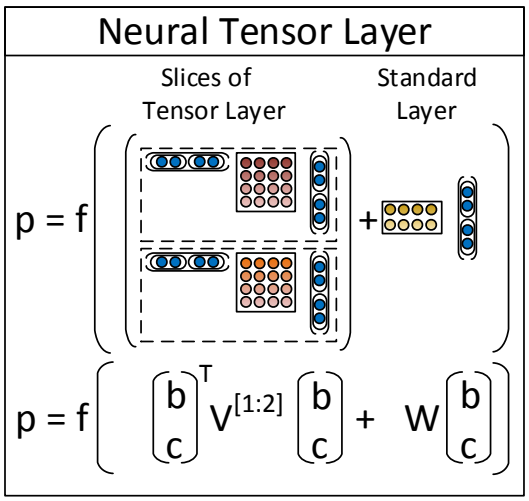


Figure 5: A single layer of the Recursive Neural Tensor Network. Each dashed box represents one of d-many slices and can capture a type of influence a child can have on its parent.

The RNTN uses this definition for computing p_1:

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right),$$

where W is as defined in the previous models. The next parent vector p_2 in the tri-gram will be computed with the same weights:

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right).$$

The main advantage over the previous RNN model, which is a special case of the RNTN when V is set to 0, is that the tensor can directly relate input vectors. Intuitively, we can interpret each slice of the tensor as capturing a specific type of composition.

An alternative to RNTNs would be to make the compositional function more powerful by adding a second neural network layer. However, initial experiments showed that it is hard to optimize this model and vector interactions are still more implicit than in the RNTN.

4.4 Tensor Backprop through Structure

We describe in this section how to train the RNTN model. As mentioned above, each node has a $\mathrm{softmax}$ classifier trained on its vector representation to predict a given ground truth or target vector t. We assume the target distribution vector at each node has a 0-1 encoding. If there are C classes, then it has length C and a 1 at the correct label. All other entries are 0.

We want to maximize the probability of the correct prediction, or minimize the cross-entropy error between the predicted distribution $y^i \in \mathbb{R}^{C\times 1}$ at node i and the target distribution $t^i \in \mathbb{R}^{C\times 1}$ at that node. This is equivalent (up to a constant) to minimizing the KL-divergence between the two distributions. The error as a function of the RNTN parameters $\theta = (V, W, W_s, L)$ for a sentence is:

$$E(\theta) = \sum_i \sum_j t_j^i \log y_j^i + \lambda \|\theta\|^2 \qquad (2)$$

The derivative for the weights of the $\mathrm{softmax}$ classifier are standard and simply sum up from each node's error. We define x^i to be the vector at node i (in the example trigram, the $x^i \in \mathbb{R}^{d\times 1}$'s are (a, b, c, p_1, p_2)). We skip the standard derivative for W_s. Each node backpropagates its error through to the recursively used weights V, W. Let $\delta^{i,s} \in \mathbb{R}^{d\times 1}$ be the $\mathrm{softmax}$ error vector at node i:

$$\delta^{i,s} = \left(W_s^T (y^i - t^i)\right) \otimes f'(x^i),$$

where $\otimes$ is the Hadamard product between the two vectors and f' is the element-wise derivative of f which in the standard case of using $f = \tanh$ can be computed using only $f(x^i)$.

The remaining derivatives can only be computed in a top-down fashion from the top node through the tree and into the leaf nodes. The full derivative for V and W is the sum of the derivatives at each of the nodes. We define the complete incoming error messages for a node i as $\delta^{i,com}$. The top node, in our case p_2, only received errors from the top node's $\mathrm{softmax}$. Hence, $\delta^{p_2,com} = \delta^{p_2,s}$ which we can use to obtain the standard backprop derivative for W (Goller and Küchler, 1996; Socher et al., 2010). For the derivative of each slice $k = 1, \ldots, d$, we get:

$$\frac{\partial E^{p_2}}{\partial V^{[k]}} = \delta_k^{p_2,com} \begin{bmatrix} a \\ p_1 \end{bmatrix} \begin{bmatrix} a \\ p_1 \end{bmatrix}^T,$$

where $\delta_k^{p_2,com}$ is just the k'th element of this vector. Now, we can compute the error message for the two

children of p_2:

$$\delta^{p_2,down} = \left(W^T\delta^{p_2,com} + S\right) \otimes f'\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}\right),$$

where we define

$$S = \sum_{k=1}^{d} \delta_k^{p_2,com}\left(V^{[k]} + \left(V^{[k]}\right)^T\right)\begin{bmatrix} a \\ p_1 \end{bmatrix}$$

The children of p_2, will then each take half of this vector and add their own $\mathrm{softmax}$ error message for the complete δ. In particular, we have

$$\delta^{p_1,com} = \delta^{p_1,s} + \delta^{p_2,down}[d+1:2d],$$

where $\delta^{p_2,down}[d+1:2d]$ indicates that p_1 is the right child of p_2 and hence takes the 2nd half of the error, for the final word vector derivative for a, it will be $\delta^{p_2,down}[1:d]$.

The full derivative for slice $V^{[k]}$ for this trigram tree then is the sum at each node:

$$\frac{\partial E}{\partial V^{[k]}} = \frac{E^{p_2}}{\partial V^{[k]}} + \delta_k^{p_1,com}\begin{bmatrix} b \\ c \end{bmatrix}\begin{bmatrix} b \\ c \end{bmatrix}^T,$$

and similarly for W. For this nonconvex optimization we use AdaGrad (Duchi et al., 2011) which converges in less than 3 hours to a local optimum.

5 Experiments

We include two types of analyses. The first type includes several large quantitative evaluations on the test set. The second type focuses on two linguistic phenomena that are important in sentiment.

For all models, we use the dev set and cross-validate over regularization of the weights, word vector size as well as learning rate and minibatch size for AdaGrad. Optimal performance for all models was achieved at word vector sizes between 25 and 35 dimensions and batch sizes between 20 and 30. Performance decreased at larger or smaller vector and batch sizes. This indicates that the RNTN does not outperform the standard RNN due to simply having more parameters. The MV-RNN has orders of magnitudes more parameters than any other model due to the word matrices. The RNTN would usually achieve its best performance on the dev set after training for 3 - 5 hours. Initial experiments showed that the recursive models worked significantly worse (over 5% drop in accuracy) when no nonlinearity was used. We use $f = \tanh$ in all experiments.

Model	Fine-grained		Positive/Negative	
	All	Root	All	Root
NB	67.2	41.0	82.6	81.8
SVM	64.3	40.7	84.6	79.4
BiNB	71.0	41.9	82.7	83.1
VecAvg	73.3	32.7	85.1	80.1
RNN	79.0	43.2	86.1	82.4
MV-RNN	78.7	44.4	86.8	82.9
RNTN	**80.7**	**45.7**	**87.6**	**85.4**

Table 1: Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

We compare to commonly used methods that use bag of words features with Naive Bayes and SVMs, as well as Naive Bayes with bag of bigram features. We abbreviate these with NB, SVM and biNB. We also compare to a model that averages neural word vectors and ignores word order (VecAvg).

The sentences in the treebank were split into a train (8544), dev (1101) and test splits (2210) and these splits are made available with the data release. We also analyze performance on only positive and negative sentences, ignoring the neutral class. This filters about 20% of the data with the three sets having 6920/872/1821 sentences.

5.1 Fine-grained Sentiment For All Phrases

The main novel experiment and evaluation metric analyze the accuracy of fine-grained sentiment classification for all phrases. Fig. 2 showed that a fine grained classification into 5 classes is a reasonable approximation to capture most of the data variation.

Fig. 6 shows the result on this new corpus. The RNTN gets the highest performance, followed by the MV-RNN and RNN. The recursive models work very well on shorter phrases, where negation and composition are important, while bag of features baselines perform well only with longer sentences. The RNTN accuracy upper bounds other models at most n-gram lengths.

Table 1 (left) shows the overall accuracy numbers for fine grained prediction at all phrase lengths and full sentences.

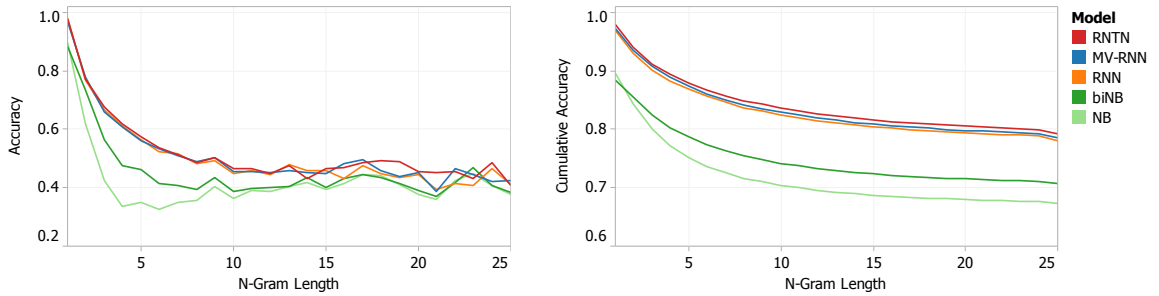


Figure 6: Accuracy curves for fine grained sentiment classification at each n-gram lengths. Left: Accuracy separately for each set of n-grams. Right: Cumulative accuracy of all $\leq n$-grams.

5.2 Full Sentence Binary Sentiment

This setup is comparable to previous work on the original rotten tomatoes dataset which only used full sentence labels and binary classification of positive/negative. Hence, these experiments show the improvement even baseline methods can achieve with the sentiment treebank. Table 1 shows results of this binary classification for both all phrases and for only full sentences. The previous state of the art was below 80% (Socher et al., 2012). With the coarse bag of words annotation for training, many of the more complex phenomena could not be captured, even by more powerful models. The combination of the new sentiment treebank and the RNTN pushes the state of the art on short phrases up to 85.4%.

5.3 Model Analysis: Contrastive Conjunction

In this section, we use a subset of the test set which includes only sentences with an 'X *but* Y' structure: A phrase X being followed by *but* which is followed by a phrase Y. The conjunction is interpreted as an argument for the second conjunct, with the first functioning concessively (Lakoff, 1971; Blakemore, 1989; Merin, 1999). Fig. 7 contains an example. We analyze a strict setting, where X and Y are phrases of different sentiment (including neutral). The example is counted as correct, if the classifications for both phrases X and Y are correct. Furthermore, the lowest node that dominates both of the word *but* and the node that spans Y also have to have the same correct sentiment. For the resulting 131 cases, the RNTN obtains an accuracy of 41% compared to MV-RNN (37), RNN (36) and biNB (27).

5.4 Model Analysis: High Level Negation

We investigate two types of negation. For each type, we use a separate dataset for evaluation.

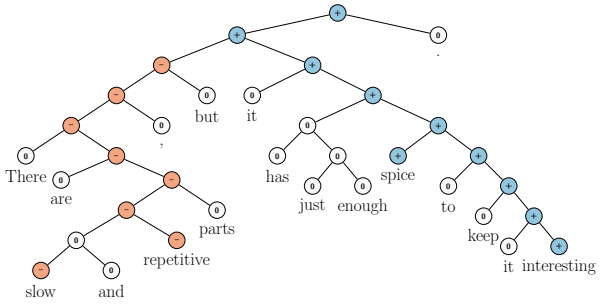


Figure 7: Example of correct prediction for contrastive conjunction X *but* Y.

Set 1: Negating Positive Sentences. The first set contains positive sentences and their negation. In this set, the negation changes the overall sentiment of a sentence from positive to negative. Hence, we compute accuracy in terms of correct sentiment reversal from positive to negative. Fig. 9 shows two examples of positive negation the RNTN correctly classified, even if negation is less obvious in the case of 'least'. Table 2 (left) gives the accuracies over 21 positive sentences and their negation for all models. The RNTN has the highest reversal accuracy, showing its ability to structurally learn negation of positive sentences. But what if the model simply makes phrases very negative when negation is in the sentence? The next experiments show that the model captures more than such a simplistic negation rule.

Set 2: Negating Negative Sentences. The second set contains negative sentences and their negation. When negative sentences are negated, the sentiment treebank shows that overall sentiment should become *less negative*, but not necessarily positive. For instance, 'The movie was terrible' is negative but the 'The movie was not terrible' says only that it was less bad than a terrible one, not that it was good (Horn, 1989; Israel, 2001). Hence, we evaluate ac-

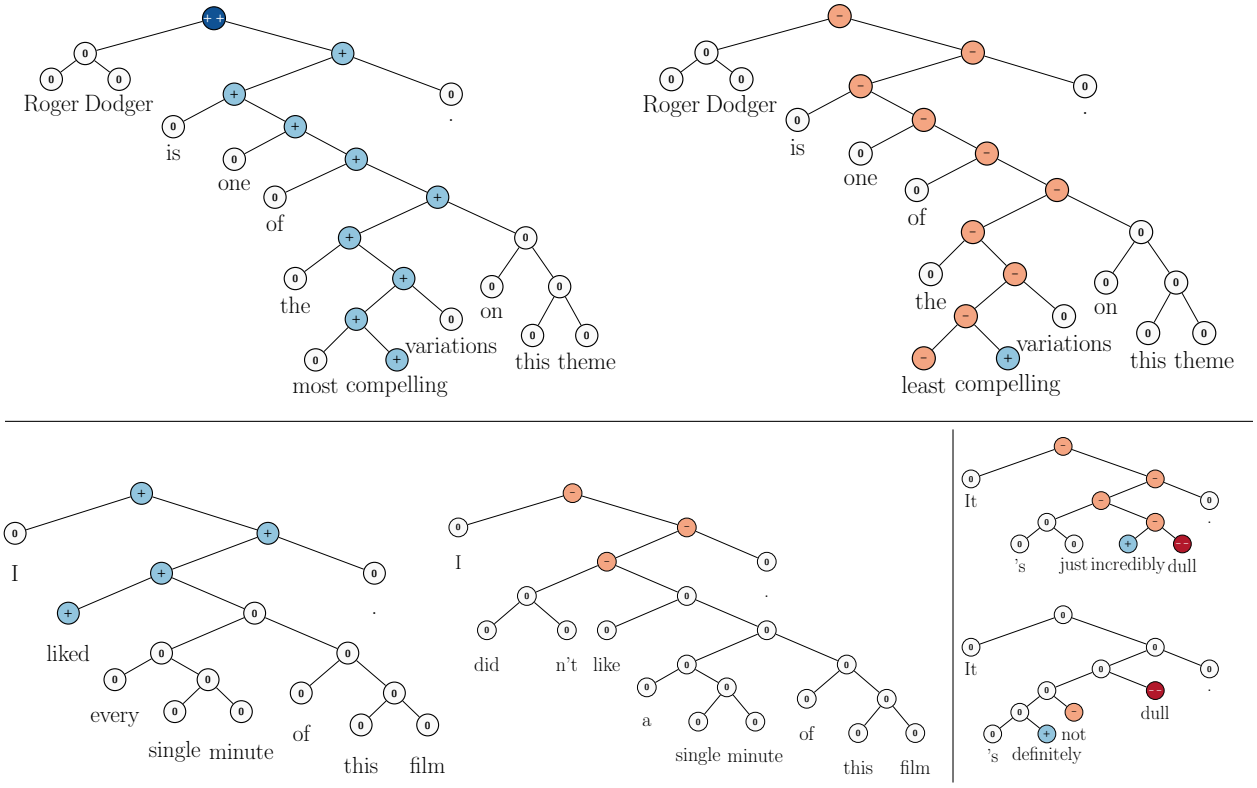


Figure 9: RNTN prediction of positive and negative (bottom right) sentences and their negation.

Model	Accuracy	
	Negated Positive	Negated Negative
biNB	19.0	27.3
RNN	33.3	45.5
MV-RNN	52.4	54.6
RNTN	**71.4**	**81.8**

Table 2: Accuracy of negation detection. Negated positive is measured as correct sentiment inversions. Negated negative is measured as increases in positive activations.

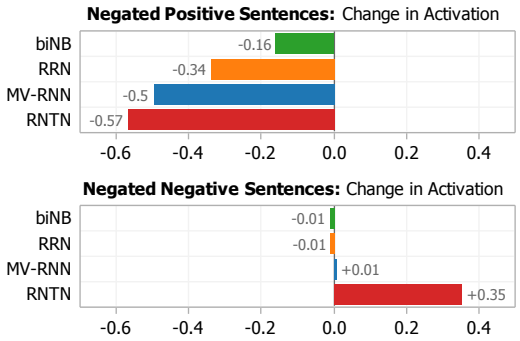


Figure 8: Change in activations for negations. Only the RNTN correctly captures both types. It decreases positive sentiment more when it is negated and learns that negating negative phrases (such as *not terrible*) should increase neutral and positive activations.

curacy in terms of how often each model was able to increase non-negative activation in the sentiment of the sentence. Table 2 (right) shows the accuracy. In over 81% of cases, the RNTN correctly increases the positive activations. Fig. 9 (bottom right) shows a typical case in which sentiment was made more positive by switching the main class from negative to neutral even though both *not* and *dull* were negative. Fig. 8 shows the changes in activation for both sets. Negative values indicate a decrease in average positive activation (for set 1) and positive values mean an increase in average positive activation (set 2). The RNTN has the largest shifts in the correct directions. Therefore we can conclude that the RNTN is best able to identify the effect of negations upon both positive and negative sentiment sentences.

n	Most positive n-grams	Most negative n-grams
1	engaging; best; powerful; love; beautiful	bad; dull; boring; fails; worst; stupid; painfully
2	excellent performances; A masterpiece; masterful film; wonderful movie; marvelous performances	worst movie; very bad; shapeless mess; worst thing; instantly forgettable; complete failure
3	an amazing performance; wonderful all-ages triumph; a wonderful movie; most visually stunning	for worst movie; A lousy movie; a complete failure; most painfully marginal; very bad sign
5	nicely acted and beautifully shot; gorgeous imagery, effective performances; the best of the year; a terrific American sports movie; refreshingly honest and ultimately touching	silliest and most incoherent movie; completely crass and forgettable movie; just another bad movie. A cumbersome and cliche-ridden movie; a humorless, disjointed mess
8	one of the best films of the year; A love for films shines through each frame; created a masterful piece of artistry right here; A masterful film from a master filmmaker,	A trashy, exploitative, thoroughly unpleasant experience ; this sloppy drama is an empty vessel.; quickly drags on becoming boring and predictable.; be the worst special-effects creation of the year

Table 3: Examples of n-grams for which the RNTN predicted the most positive and most negative responses.

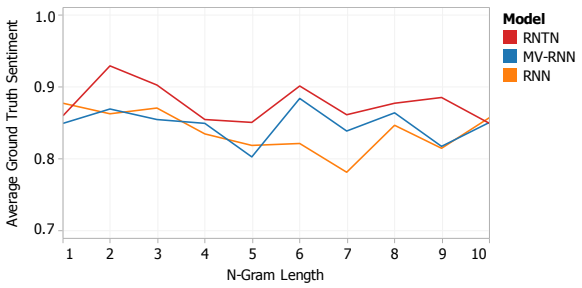


Figure 10: Average ground truth sentiment of top 10 most positive n-grams at various n. The RNTN correctly picks the more negative and positive examples.

5.5 Model Analysis: Most Positive and Negative Phrases

We queried the model for its predictions on what the most positive or negative n-grams are, measured as the highest activation of the most negative and most positive classes. Table 3 shows some phrases from the dev set which the RNTN selected for their strongest sentiment.

Due to lack of space we cannot compare top phrases of the other models but Fig. 10 shows that the RNTN selects more strongly positive phrases at most n-gram lengths compared to other models.

For this and the previous experiment, please find additional examples and descriptions in the supplementary material.

6 Conclusion

We introduced Recursive Neural Tensor Networks and the Stanford Sentiment Treebank. The combination of new model and data results in a system for single sentence sentiment detection that pushes state of the art by 5.4% for positive/negative sentence classification. Apart from this standard setting, the dataset also poses important new challenges and allows for new evaluation metrics. For instance, the RNTN obtains 80.7% accuracy on fine-grained sentiment prediction across all phrases and captures negation of different sentiments and scope more accurately than previous models.

Acknowledgments

We thank Rukmani Ravisundaram and Tayyab Tariq for the first version of the online demo. Richard is partly supported by a Microsoft Research PhD fellowship. The authors gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0040, the DARPA Deep Learning program under contract number FA8650-10-C-7020 and NSF IIS-1159679. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3, March.

D. Blakemore. 1989. Denial and contrast: A relevance theoretic analysis of 'but'. *Linguistics and Philosophy*, 12:15–37.

L. Bottou. 2011. From machine learning to machine reasoning. *CoRR*, abs/1102.1808.

S. Clark and S. Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pages 52–55.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12, July.

K. Erk and S. Padó. 2008. A structured vector space model for word meaning in context. In *EMNLP*.

C. Goller and A. Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks (ICNN-96)*.

E. Grefenstette and M. Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *EMNLP*.

E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *IWCS*.

G. E. Hinton. 1990. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2).

L. R. Horn. 1989. *A natural history of negation*, volume 960. University of Chicago Press Chicago.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*.

M. Israel. 2001. Minimizers, maximizers, and the rhetoric of scalar reasoning. *Journal of Semantics*, 18(4):297–331.

R. Jenatton, N. Le Roux, A. Bordes, and G. Obozinski. 2012. A latent factor model for highly multi-relational data. In *NIPS*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

R. Lakoff. 1971. If's, and's, and but's about conjunction. In Charles J. Fillmore and D. Terence Langendoen, editors, *Studies in Linguistic Semantics*, pages 114–149. Holt, Rinehart, and Winston, New York.

A. Merin. 1999. Information, relevance, and social decisionmaking: Some principles and results of decision-theoretic semantics. In Lawrence S. Moss, Jonathan Ginzburg, and Maarten de Rijke, editors, *Logic, Language, and Information*, volume 2. CSLI, Stanford, CA.

J. Mitchell and M. Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

K. Moilanen and S. Pulman. 2007. Sentiment composition. In *In Proceedings of Recent Advances in Natural Language Processing*.

T. Nakagawa, K. Inui, and S. Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *NAACL*, HLT.

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

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

B. Pang and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

T. A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

L. Polanyi and A. Zaenen. 2006. Contextual valence shifters. In W. Bruce Croft, James Shanahan, Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text: Theory and Applications*, volume 20 of *The Information Retrieval Series*, chapter 1.

J. B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46, November.

M. Ranzato and A. Krizhevsky G. E. Hinton. 2010. Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images. *AISTATS*.

V. Rentoumi, S. Petrakis, M. Klenner, G. A. Vouros, and V. Karkaletsis. 2010. United we stand: Improving sentiment analysis by joining machine learning and rule based methods. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

S. Rudolph and E. Giesbrecht. 2010. Compositional matrix-space models of language. In *ACL*.

B. Snyder and R. Barzilay. 2007. Multiple aspect ranking using the Good Grief algorithm. In *HLT-NAACL*.

R. Socher, C. D. Manning, and A. Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

R. Socher, C. Lin, A. Y. Ng, and C.D. Manning. 2011a. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*.

I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. 2009. Modelling relational data using Bayesian clustered tensor factorization. In *NIPS*.

P. D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*.

D. Widdows. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.

A. Yessenalina and C. Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *EMNLP*.

D. Yu, L. Deng, and F. Seide. 2012. Large vocabulary speech recognition using deep tensor neural networks. In *INTERSPEECH*.

F.M. Zanzotto, I. Korkontzelos, F. Fallucchi, and S. Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *COLING*.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Open Domain Targeted Sentiment

Margaret Mitchell **Jacqueline Aguilar** **Theresa Wilson** **Benjamin Van Durme**
Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD 21218, USA
{m.mitchell,jacqui.aguilar}@jhu.edu, Theresa.Wilson@oberlin.edu, vandurme@cs.jhu.edu

Abstract

We propose a novel approach to sentiment analysis for a low resource setting. The intuition behind this work is that sentiment expressed towards an entity, *targeted sentiment*, may be viewed as a span of sentiment expressed across the entity. This representation allows us to model sentiment detection as a sequence tagging problem, jointly discovering people and organizations along with whether there is sentiment directed towards them. We compare performance in both Spanish and English on microblog data, using only a sentiment lexicon as an external resource. By leveraging linguistically-informed features within conditional random fields (CRFs) trained to minimize empirical risk, our best models in Spanish significantly outperform a strong baseline, and reach around 90% accuracy on the combined task of named entity recognition and sentiment prediction. Our models in English, trained on a much smaller dataset, are not yet statistically significant against their baselines.

1 Introduction

Sentiment analysis is a multi-faceted problem. Determining when a positive or negative sentiment is being expressed is a large part of the challenge, but identifying other attributes, such as the target of the sentiment, is also crucial if the ultimate goal is to pinpoint and extract opinions. Consider the examples below, all of which contain a positive sentiment:

(1) So happy that Kentucky lost to Tennessee!

(2) Kentucky versus Kansas I can hardly wait...

(3) Kentucky is the best alley-oop throwing team since Sherman Douglas' Syracuse squads!!

The entities in these examples are college basketball teams, and the events referred to are games. In (1), although there is a positive sentiment, the target of the sentiment is an event (Kentucky losing to Tennessee). However, from the positive sentiment toward this event, we can infer that the speaker has a negative sentiment toward Kentucky and a positive sentiment toward Tennessee. In (2), the positive sentiment is toward a future event, but we are not given enough information to infer a sentiment toward the mentioned entities. In (3), Kentucky is the direct target of the positive sentiment. We can also infer a positive sentiment toward Douglas's Syracuse teams, and even toward Douglas himself.

These examples illustrate the importance of the target when interpreting sentiment in context. If we are looking for sentiments toward Kentucky, for example, we would want to identify (1) as negative, (2) as neutral (no sentiment) and (3) as positive. However, if we are looking for sentiment toward Tennessee, we would want to identify (1) as positive, and (2) and (3) as neutral.

The expression of these and other kinds of sentiment can be understood as involving three items:

(1) An experiencer

(2) An attitude

(3) A target (optionally)

Research in sentiment analysis often focuses on (2), predicting overall sentiment polarity (Agarwal et al., 2011; Bora, 2012). Recent work has begun to combine (2) with (3), examining how to automatically predict the sentiment polarity expressed towards a target entity (Jiang et al., 2011; Chen et al., 2012) for a fixed set of targets. This topic-dependent sentiment classification requires that the target entity be

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1643–1654,
Seattle, Washington, USA, 18-21 October 2013.

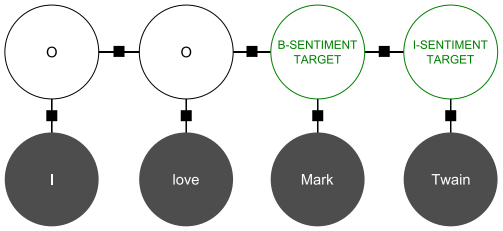


Figure 1: Sentiment expressed across an entity.

given, and returns statements expressing sentiment towards the given entity.

In this paper, we take a step towards open-domain, *targeted sentiment analysis* by investigating how to detect both the named entity and the sentiment expressed toward it. We observe that sentiment expressed towards a target entity may be possible to learn in a graphical model along the span of the entity itself: Similar to how named entity recognition (NER) learns labels along the span of each word in an entity name, sentiment may be expressed along the entity as well. A small example is shown in Figure 1. We focus on people and organizations (*volitional named entities*), which are the primary targets of sentiment in our microblog data (see Table 1).

Both NER and opinion expression extraction have achieved impressive results using conditional random fields (CRFs) (Lafferty et al., 2001) to define the conditional probability of entity categories (McCallum and Li, 2003; Choi et al., 2006; Yang and Cardie, 2013). We develop such models to jointly predict the NE and the sentiment expressed towards it using minimum risk training (Stoyanov and Eisner, 2012). We learn our models on informal Spanish and English language taken from the social network Twitter,[1] where the language variety makes NLP particularly challenging (see Figure 2).

Our ultimate goal is to develop models that will be useful for low resource languages, where a sentiment lexicon may be known or bootstrapped, but more sophisticated linguistic tools may not be readily available. We therefore do not rely on an external part-of-speech tagger or parser, which are often used for features in fine-grained sentiment analysis; such tools are not available in many languages, and if they are, are not usually adapted for noisy social media.

Instead, we use information from sentiment lexicons and some simple hand-written features, and otherwise use only features of the word that can be

@[user] le dijo erralo muy por lo bajo jaja un grande juancito grandes amigos mios
@[*user*] *he told him it was very on the dl haha a great juancito great friends of mine*

@[user] buenos días Profe!! Nos quedamos accidentados otra vez en la carretera vieja guarenas echando gasoil, estamos a la interperie
@[*user*] *good morning, Prof!! We were wrecked again on the old guarenas highway while getting diesel, we're out in the open*

Sin ánimo de ofender a los Militares, que realmente se merecen ese aumento y más. Pero, dónde queda la misma recompensa para Médicos.
I do not intend to offend the military in the slightest, they truly deserve the raise and more. However, I'm wondering whether doctors will ever receive a similar compensation.

Figure 2: Messages on Twitter use a wide range of formality, style, and errors, which makes extracting information particularly difficult. Examples from Spanish (screen names anonymized), with approximate translations in English.

extracted without supervision. These include features based on unsupervised word tags (Brown clusters) and a method that automatically syllabifies a word based on the orthography of the language. All tools and code used for this research are released with this paper.[2]

2 Related Work

As the scale of social media has grown, using sources such as Twitter to mine public sentiment has become increasingly promising. Commercial systems include Sentiment140[3] (products and brands) and tweetfeel[4] (suggests searching for popular movies, celebrities and companies).

The majority of academic research has focused on supervised classification of message sentiment irrespective of target (Barbosa and Feng, 2010; Pak and Paroubek, 2010; Bifet and Frank, 2010; Davidov et al., 2010; Kouloumpis et al., 2011; Agarwal et al., 2011). Large datasets are collected for this work by leveraging the sentiment inherent in emoticons (e.g., smilies and frownies) and/or select Twitter hashtags (e.g., #bestdayever, #fail), resulting in noisy collec-

[1] www.twitter.com

[2] www.m-mitchell.com/code

[3] www.sentiment140.com

[4] www.tweetfeel.com

tions appropriate for initial exploration. Prior work includes: the use of a social network (Speriosu et al., 2011; Tan et al., 2011; Calais Guerra et al., 2011; Jiang et al., 2011; Li et al., 2012; Hu et al., 2013); user-adapted models based on collaborative online-learning (Li et al., 2010b); unsupervised, joint sentiment-topic modeling (Saif et al., 2012); tracking changing sentiment during debates (Diakopoulos and Shamma, 2010); and how orthographic conventions such as word-lengthening can be used to adapt a Twitter-specific sentiment lexicon (Brody and Diakopoulos, 2011).

Efforts in targeted sentiment (Bermingham and Smeaton, 2010; Jin and Ho, 2009; Li et al., 2010a; Jiang et al., 2011; Tan et al., 2011; Wang et al., 2011; Li et al., 2012; Chen et al., 2012), have mostly focused on topic-dependent analysis. In these approaches, messages are collected on a fixed set of topics/targets, such as products or sports teams, and sentiment is learned for the given set. In contrast, we aim to predict sentiment in tweets for any named person or organization. We refer to this task as *open domain targeted sentiment analysis*.

Within topic-dependent sentiment analysis, several approaches have explored applying CRFs or HMMs to extract sentiment and target words from text (Jin and Ho, 2009; Li et al., 2010a). In these approaches, opinion expressions are extracted, and polarity is annotated across the opinion expression. However, as noted by many researchers in sentiment, opinion orientation towards a specific target is often not equal to the orientation of a neighboring opinion expression; and opinion expressions in one context may not be opinion expressions in another (Kim and Hovy, 2006), making open domain approaches particularly challenging.

The above work by Jiang et al. (2011) is most similar to our own. They do not use joint learning, but they do incorporate a number of parse-based features designed to capture relationships between sentiment terms and topic references. In our work these relationships are captured by the CRF model, and we compare against their approach in Section 6.

Recent work by Yang and Cardie (2013) is similar in spirit to our own, where the identification of opinion holders, opinion targets, and opinion expressions is modeled as a sequence tagging problem using a CRF. However, similar to previous work applying CRFs to extract sentiment, Yang and Cardie use syntactic relations to connect an opinion target to an opinion expression. In contrast, we model the expression of sentiment polarity across the sentiment target itself, extracting both the sentiment target and the sentiment expressed towards it within the same span of words. This allows us to use surrounding context to determine sentiment polarity without identifying explicit opinion expressions or relying on a parser to help link expression to target.

Most work in targeted sentiment outside the microblogging domain has been in relation to product review mining (e.g., Yi et al. (2003), Hu and Liu (2004), Popescu and Etzioni (2005), Qiu et al. (2011)). Rather than identify named entities (NEs), this work seeks to identify products and their features mentioned in reviews, and classify these for sentiment. Recent work by Qui et al. jointly learns targets and opinion words, and Jakob and Gurevych (2010) use CRFs to extract the targets of opinions, but do not attempt to classify the sentiment toward these targets. To the best of our knowledge, this is the first work to approach targeted sentiment in a low resource setting and to jointly predict NEs and targeted sentiment.

3 Data

Twitter Collection We use the Spanish/English Twitter dataset of Etter et al. (2013) to train and test our models. Approximately 30,000 Spanish tweets and 10,000 English were labeled for named entities in BIO encoding: The start of an NE is labeled B-{NE} and the rest of the NE is labeled I-{NE}. The

NE	COUNT	NEUTRAL	POS	NEG
PERSON	5462	80%	20%	0%
ORGANIZATION	4408	80%	20%	0%
LOCATION	1405	100%	0%	0%
URL	1030	100%	0%	0%
TIME	535	70%	10%	20%
DATE	222	100%	0%	0%
MONEY	95	90%	0%	10%
PERCENT	81	80%	20%	0%
TELEPHONE	23	100%	0%	0%
EMAIL	8	100%	0%	0%

Table 1: Distribution of named entities in our Spanish Twitter corpus. Targeted sentiment percentages are based on expert annotations from a random sample of 10 (or all) of of each entity. Most entities are not sentiment targets (NEUTRAL). PERSON and ORGANIZATION are most frequent, and among the top recipients of sentiment.

full set of NE categories are shown in Table 1. For example, the sequence "Mark Twain" would be labeled B-PERSON, I-PERSON. We are interested in both PERSON and ORGANIZATION entities, which make up the majority of named entities in this data, and we evaluate these using the more general entity category VOLITIONAL. Removing retweets, 7,105 Spanish tweets contained a total of 9,870 volitional entities and 2,350 English tweets contained a total of 3,577 volitional entities.

Sentiment Lexicons We use two sentiment lexicon sources in each language. For English, we use the MPQA lexicon (Wilson et al., 2005), which identifies 12,296 manually and semi-automatically produced subjective terms along with their polarity. For the second lexicon, we use SentiWordNet 3.0 (Baccianella et al., 2010), which assigns positive and negative polarity scores to WordNet synsets. We use the majority polarity of all words with a subjectivity score above 0.5.

For Spanish, the first lexicon is obtained from Volkova et al. (2013), who automatically translated strongly subjective terms from the MPQA lexicon (Wilson et al., 2005) into Spanish. The resulting Spanish lexicon contains about 65K words. The second lexicon is available from Perez-Rosas et al. (2012). This contains approximately 1000 sentiment-bearing words collected leveraging manual resources and 2000 collected leveraging automatic resources.

Annotation To collect sentiment labels, we use crowdsourcing through Amazon's Mechanical Turk.[5] Annotators ("Turkers") were shown six tweets at a time, each with a single highlighted named entity. Turkers were instructed to (1) select the sentiment being expressed towards the entity (*positive*, *negative*, or *no sentiment*); and (2) rate their level of confidence in their selection. Following best practices on collecting language data with Mechanical Turk (Callison-Burch and Dredze, 2010), two controls were placed among each set of six tweets to screen out unreliable judgments. An example prompt is shown in Figure 3.

Each ⟨tweet, NE⟩ pair was shown to three Turkers, and those with majority consensus on sentiment polarity were extracted. Tweets without sentiment consensus on all NEs were removed. In Spanish, this yielded 6,658 unique ⟨tweet, NE⟩ pairs. In English, which is a smaller data set, this yielded 3,288 unique pairs. We split the data into folds for 10-fold cross-validation, developing on the data from one fold and reporting results for the remaining nine.

[5] www.mturk.com/mturk

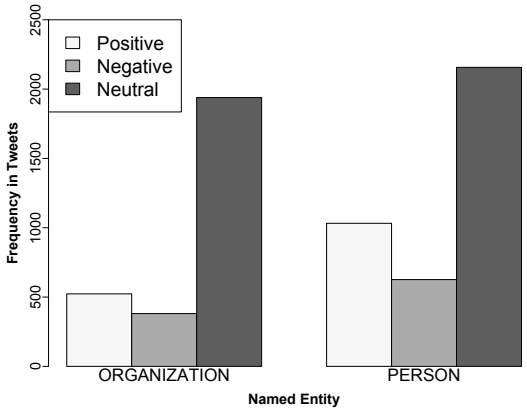


Figure 4: Targeted sentiment annotated for Spanish.

		Majority		
		POS	NEUTRAL	NEG
Minority	POS	757	1249	130
	NEUTRAL	707	2151	473
	NEG	129	726	452

Table 2: Number of targeted sentiment instances where at least two of the three annotators (Majority) agreed. Common disagreements with a third annotator (Minority) were over whether no sentiment or positive sentiment was expressed, and whether no sentiment or negative sentiment was expressed.

The distribution of sentiment for the named entities annotated by Turkers is shown in Figure 4. Neutral (no targeted sentiment) dominates, followed by positive sentiment for both organizations and people. As shown in Table 2, common disagreements were over whether or not there was targeted positive sentiment, and whether or not there was targeted negative sentiment. This is in line with previous research showing that distinguishing positive sentiment from no sentiment (and distinguishing negative sentiment from no sentiment) is often more challenging than distinguishing between positive and negative sentiment (Wilson et al., 2009). Indeed, we see that it was more common for annotators to disagree than to agree on targeted sentiment, particularly for negative targeted sentiment, where more instances had NEUTRAL/NEGATIVE disagreement than NEGATIVE three-way agreement.

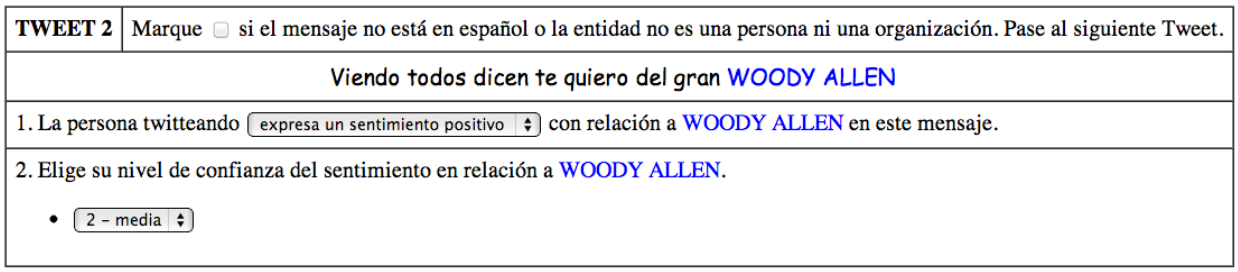


Figure 3: Example Tweet shown to Turkers.

Variable	Possible values
Sentiment (s) (PIPE & JOINT models)	NOT-TARG, SENT-TARG
Named Entity (l) (PIPE & JOINT models)	O, B-VOLITIONAL, I-VOLITIONAL
Combined Sent/NE (y) (COLL models)	O, B+NOT-TARG, I+NOT-TARG B+SENT-TARG, I+SENT-TARG

Table 3: Possible values for random variables, targeted subjectivity (is/is not sentiment target). COLL models collapse targeted subjectivity and NE label into one node.

Variable	Possible values
Sentiment (s) (PIPE & JOINT models)	NOT-TARG, POS, NEG
Named Entity (l) (PIPE & JOINT models)	O, B-VOLITIONAL, I-VOLITIONAL
Combined Sent/NE (y) (COLL models)	O, B+NOT-TARG, I+NOT-TARG B+POS, I+POS B+NEG, I+NEG

Table 4: Possible values for random variables, targeted sentiment. The COLL models collapse both targeted sentiment and NE label into one node.

4 Targeted Subjectivity and Sentiment

Formally, we define the problem as follows: Given an observed message $\mathbf{w} = (w_1 \dots w_n)$, where n is the number of words in the message and $w_j (1 \leq j \leq n)$ is a word, we learn the probability of a label sequence $\mathbf{l} = (l_1 \dots l_n)$, where $l_i \in$ the set of named entity values; and a sentiment sequence $\mathbf{s} = (s_1 \dots s_n)$, where $s_i \in$ the set of sentiment values. We additionally explore simpler linear-chain models that learn the probability of a single label sequence $\mathbf{y} = (y_1 \dots y_n)$, where $y_i \in$ the set of conjoined entity+sentiment values (Tables 3 and 4).

Our basic model is a linear conditional random field, an undirected graph that represents the conditional distribution $p(\mathbf{l}, \mathbf{s}|\mathbf{w})$.[6] Sentiment towards a named entity may be modeled in a CRF as a sequence of random variables for sentiment $\mathbf{s}$ connected to named entities $\mathbf{l}$. In all models, entity variables are connected by a factor to their neighbors in sequence, and we include skip-chains (Finkel and Manning, 2010) connecting identical words where at least one is capitalized. Our model strategies include: a pipeline that first learns volitional entities then sentiment directed towards them (PIPE); one that jointly learns volitional entities along with sentiment directed towards them (JOINT); and one that learns volitional entities and targeted sentiment with combined labels (COLL) (Figure 5).

Using these models, we explore two primary tasks: (1) the task of detecting whether sentiment is targeted at an entity, which we refer to as *targeted subjectivity*; and (2) the task of detecting whether positive, negative, or neutral sentiment (no sentiment) is targeted at an entity, which we refer to as *targeted sentiment*. Moving from targeted subjectivity prediction to targeted sentiment prediction is possible by changing the sentiment target (SENT-TARG) variable into two variables, one for positive targeted sentiment (POS) and one for negative (NEG). Possible values for targeted subjectivity are shown in Table 3, and possible values for targeted sentiment are shown in Table 4.

In the pipeline models (PIPE), we first build a CRF where each word is connected by a factor to an entity label $l_i \in \mathbf{l}$. In a second model, every observed volitional entity node is connected by a factor to a sentiment label $s_i \in \mathbf{s}$. An example is shown in Figure 5 (1).

In the joint models (JOINT), each $s_i \in \mathbf{s}$ is connected by a factor to the corresponding entity label in the sequence, $l_i \in \mathbf{l}$. Sentiment in this model is partially observed: All sentiment variables are treated as latent except for the sentiment connected to the volitional entity. An example is shown in Figure 5 (2).

[6] For the COLL models, this is instead the conditional distribution $p(\mathbf{y}|\mathbf{w})$, where entity and sentiment labels are conjoined in one sequence assignment $\mathbf{y}$.

In the collapsed models (COLL), we combine sentiment and named entity into one label sequence (e.g., O, B+SENT-TARG, I+SENT-TARG). An example is shown in Figure 5 (3). The JOINT and PIPE models therefore predict named entity sequences, their category labels, and the sentiment expressed towards volitional named entities.[7] The collapsed models predict volitional labels and targeted sentiment as combined categories. The COLL and PIPE models are considerably faster than JOINT models, where exact inference is intractable.

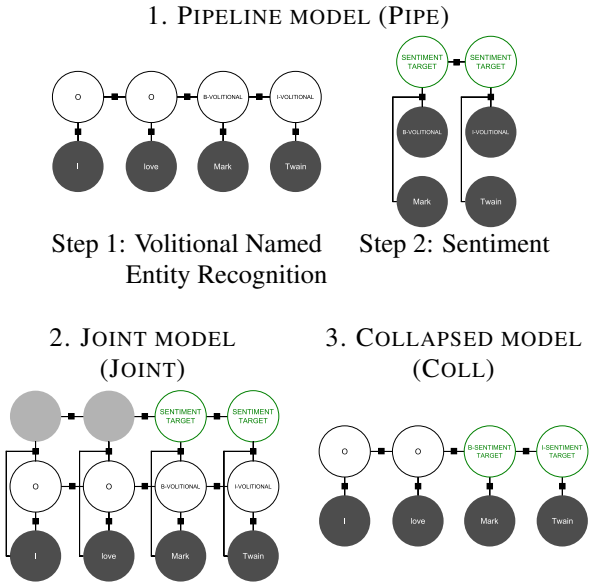


Figure 5: Example CRFs for targeted subjectivity with observed variables (dark nodes), predicted variables (white nodes) and hidden variables (light grey nodes).

5 Training

Minimum-Risk CRF Training We use the ERMA system (Stoyanov et al., 2011) to learn our models.[8] ERMA (Empirical Risk Minimization under Approximations) learns parameters to minimize loss on the training data. Predicting NE labels using a linear-chain CRF trained with empirical risk minimization has been shown to result in a statistically significant improvement over the common approach of maximum likelihood estimation (Stoyanov and Eisner, 2012). All models are trained to optimize log likelihood using 20 iterations of stochastic gradient descent, and a maximum of 100 iterations of belief propagation to compute the marginals for each example.

Features Features of the models are shown in Table 5. For an observed word, features are extracted for the word itself as well as within a context window of three words in either direction. Words seen only once are treated as out-of-vocabulary. Surface features and linguistic features are concatenated in groups of two and three to create further features. All algorithms and code that we have developed for feature extraction are available online.[9]

Because we aim to develop models that do not heavily rely on language-specific resources, we are interested in exploring unsupervised and lightly supervised methods for learning relevant features. Rather than use part-of-speech tags, we therefore use Brown cluster labels as unsupervised word tags (Brown et al., 1992; Koo et al., 2008). Brown clustering is a distributional similarity method that merges pairs of word clusters in the training data[10] to create the smallest decrease in corpus likelihood, using a bigram language model on the clusters. For our task, we cut clusters at length 3 and length 5, and these serve as rough part-of-speech tags without the need to train additional models. For example, the word *hello* is tagged as belonging to cluster 011 (length 3) and 01111 (length 5).

During development, we found that being able to syllabify the word (break the word into syllables) was a positive indicator of people names, but a negative indicator of organization names. This observation can be approximated automatically using constraints from the sonority sequencing principle (Hooper, 1976; Clements, 1990; Blevins, 1996; Morelli, 2003) on a language's orthography. This is a phonotactic principle that states that syllables will tend to have a sonority peak, usually a vowel, in the center of the syllable, followed on either side by consonants with decreasing sonority. Although languages may violate this principle, the core idea that a vowel forms the nucleus of a syllable with op-

[7] We found that learning the VOLITIONAL categories during training rather than maintaining beliefs about separate named entities during inference (ORGANIZATION, PERSON) and then post-processing to VOLITIONAL leads to slightly better accuracy.

[8] sites.google.com/site/ermasoftware

[9] www.m-mitchell.com/code

[10] For Spanish, we train on a sample of ~7 million Spanish tweets. For English, we train on the essays (Pennebaker et al., 2007) and Facebook data (Kosinskia et al., 2013) available from ICWSM 2013.

tional consonants before (the onset) and after (the coda) can be used to begin to automatically learn syllable structure.[11] We learn this in an unsupervised way, using the most frequent (seen more than 1,000 times) word-initial non-vowel sequences from the Brown cluster data as allowable syllable onset consonants. Similarly, the most frequent word-final non-vowel sequences are learned as possible syllable codas. For each word, we then attempt to segment syllables using the learned onsets and codas around each vowel. If a word cannot be syllabified, it is often an initialism (e.g., *CND*, *lsat*).

We follow the approach from the out-of-vocabulary assignment in the Berkeley parser (Petrov et al., 2006) to encode common surface patterns such as capitalization and lexical patterns such as verb endings as a single feature for words we have seen once or less. We also use the Jerboa toolkit (Van Durme, 2012) to extract further language-independent features from the data, such as features for emoticons and binning for repeated characters (like !!!). In addition, we include features for whether the word is three or four letters, which is often used for acronyms and initialisms in several languages (including Spanish and English); whether the word is neighbored by a punctuation mark; word identity; word length; message length; and position in the sentence.

We utilize a speaker of each language to simply list word forms for sentiment features that may be indicative of sentiment, totaling less than two hours of annotation time. This set includes intensifiers (e.g., *hella*, *freakin'* in English; e.g., *muy*, *sumamente* in Spanish), positive/negative abbreviations (*WTF*, *pso*), positive/negative slang words, and positive/negative prefix and suffixes (e.g., *anti-* in English and Spanish, *-ito* in Spanish).

6 Experiments

We are interested in both PERSON and ORGANIZATION entities, and evaluate these in the collapsed category VOLITIONAL. This suggests that the data may be pre-processed to label all volitional entities as VOLITIONAL NEs, or the models may be learned with the traditional named entities in place, and post-processed to identify those that are VOLITIONAL. We explored results using both methods, and found that training models on VOLITIONAL tags yielded the best performance overall; we report numbers for this approach below.

[11] Further development is necessary to extend a similar idea to languages that do not ordinarily mark all vowels in their orthography, such as Hebrew and Arabic.

SURFACE FEATURES
binned word length, message length, and sentence position; Jerboa features; word identity; word lengthening; punctuation characters, has digit; has dash; is lower case; is 3 or 4 letters; first letter capitalized; more than one letter capitalized, etc.
LINGUISTIC FEATURES
function words; can syllabify; curse words; laugh words; words for *good*, *bad*, *no*, *my*; slang words; abbreviations; intensifiers; subjective suffixes and prefixes (such as diminutive forms); common verb endings; common noun endings
BROWN CLUSTERING FEATURES
cluster at length 3; cluster at length 5
SENTIMENT FEATURES
is sentiment-bearing word; prior sentiment polarity

Table 5: Features used in model.

We compare against a baseline (BASE-NS) where we use our volitional entity labels and assign no sentiment directed towards the entity (the majority case). This is a strong baseline to isolate how our methods perform specifically for the task of identifying sentiment targeted at an entity.

We report on precision, recall, and sensitivity for the tasks of NER and targeted subjectivity/sentiment prediction in isolation; and we report on accuracy for the targeted subjectivity and targeted sentiment models. For sentiment, a true positive is an instance where the label has sentiment, and a true negative is an instance where the label has no sentiment (neutral). For NER, a true positive is an instance where the label is a B- or I- label; a true negative is an instance where the label is O. The three systems are evaluated against one another for NER, subjectivity (entity has/does not have sentiment expressed towards it), and sentiment (positive/negative/no sentiment) using paired t-tests across folds, with a Bonferroni correction to set α to 0.02.

NER We include results for the isolated task of volitional named entity recognition in Table 6. In both Spanish and English, all three models are roughly comparable for precision, recall, and specificity. The task of finding O tags – spans that are *not* named entities – works especially well (**NE spec**). Common

Model	Spanish			English		
	Joint	Pipe	Coll	Joint	Pipe	Coll
NE prec	**65.2**	64.3	65.1	59.8	**62.3**	60.5
NE rec	**65.8**	64.7	61.2	**60.2**	57.2	56.5
NE spec	95.4	95.2	**95.6**	94.3	**95.1**	94.7

Table 6: Average precision, recall, and specificity for volitional entity NER (in %).

mistakes include confusing B- labels with I- labels.

Subjectivity and Sentiment Table 7 shows results for the isolated task of predicting the presence of sentiment about a volitional entity. In Spanish, the pipeline models (PIPE) perform optimally for subjectivity recall (**Subj rec**), and significantly above the COLL models ($p<.001$). Precision and specificity are comparable across models. In English as in Spanish, the collapsed model is particularly poor at subjectivity recall.

As discussed in Section 2, the subtask of predicting whether subjectivity is expressed towards an entity is comparable to the main task of Jiang et al. (2011), and so we compare our approach here. The Jiang et al. study is similar to the current study in that they aim to detect targeted sentiment, but it differs from the current study in that they focus exclusively on subjectivity towards five manually selected entities: {*Obama*, *Google*, *iPad*, *Lakers*, *Lady Gaga*}. They also evaluate on artificially balanced evaluation data, and evaluate sentiment polarity (positive/negative) separately from subjectivity (has/does not have sentiment).

Our dataset includes any entity labeled as PERSON or ORGANIZATION, and is not balanced (most targets have no sentiment expressed towards them; see Table 1), thus we can only roughly compare against their approach. *Lakers* and *Lady Gaga* are rare in our collection (appearing less than 3 times), and so we updated the comparison set prior to evaluation to: {*Obama*, *Google*, *iPad*, *BBC*, *Tebow*}. On this set, a baseline that always guesses no sentiment reaches an accuracy of 66.9%, compared to Jiang et al.'s 65.5% accuracy on a balanced set (not strictly comparable, but provided for reference). The JOINT models reach an accuracy of 71.04% on this set, demonstrating this approach as potentially useful for topic-dependent targeted sentiment.

Table 8 shows results for the task of predicting the polarity of the sentiment expressed about an entity. In Spanish, the PIPE models significantly outperform the COLL models on sentiment recall, and the JOINT models on sentiment precision ($p<.01$). In English, PIPE significantly outperforms JOINT on precision ($p<.001$).

Model	Spanish			English		
	Joint	Pipe	Coll	Joint	Pipe	Coll
Subj prec	58.3	58.8	**58.9**	46.6	**52.2**	45.9
Subj rec	40.1	**50.9**	19.1	44.5	**48.5**	16.4
Subj spec	**79.6**	77.5	77.8	77.6	**80.8**	74.0

Table 7: Average precision, recall, and specificity (in %) for subjectivity prediction (has/does not have sentiment) along the target entity.

Model	Spanish			English		
	Joint	Pipe	Coll	Joint	Pipe	Coll
Sent prec	36.6	**45.8**	42.5	31.6	**42.9**	38.5
Sent rec	38.0	**40.6**	15.5	**36.6**	34.8	9.7
Sent spec	67.1	**75.2**	73.3	72.3	**82.0**	78.1

Table 8: Average precision, recall, and specificity (in %) for sentiment prediction (positive/negative/no sentiment) along the target entity.

Targeted Subjectivity and Targeted Sentiment The JOINT and PIPE models work reasonably well for the isolated tasks of NER and subjectivity/sentiment prediction. We now examine results for *targeted subjectivity* – labeling an entity and predicting whether there is sentiment directed towards it – in Table 9; and *targeted sentiment* – labeling an entity and predicting what the sentiment directed towards it is – in Table 10.

We evaluate using two accuracy metrics: **Acc-all**, which measures the accuracy of the entire named entity span along with the sentiment span; and **Acc-Bsent**, which measures the accuracy of identifying the start of a named entity (B- labels) along with the sentiment expressed towards it. Acc-all primarily measures the correctness of O labels, while Acc-Bsent focuses on the beginning of named entities.

For the targeted subjectivity task, our JOINT models perform optimally in Spanish, and significantly above their baselines. For the Acc-Bsent task, JOINT models perform best, significantly outperforming their baseline for subjectivity prediction. In English, where our data is half the size, we do not see a statistically significant difference between the predictive models and the no sentiment baselines.

For the targeted sentiment task, the JOINT models again perform relatively well in Spanish (Table 10), labeling volitional entities, predicting whether or not there is sentiment targeted towards them, and

	Model	Joint	Joint Base	Pipe	Pipe Base	Coll	Coll Base
Spa	**Acc-all**	89.5*	89.3	89.3**	89.1	89.5*	89.3
	Acc-Bsent	32.1***	29.5	30.9***	28.3	30.1**	28.1
Eng	**Acc-all**	88.0	88.1	88.6	88.6	87.9	88.1
	Acc-Bsent	30.4	30.8	30.7	30.3	28.1	29.2

***p<.001 **p<.01 *p<.05

Table 9: Average accuracy on Targeted Subjectivity Prediction: Identifying volitional entities and whether they are a sentiment target. In the core task, **Acc-Bsent**, the best model in Spanish is JOINT, significantly outperforming the baseline. In English, the best model (PIPE) does not significantly improve over its baseline.

	Model	Joint	Joint Base	Pipe	Pipe Base	Coll	Coll Base
Spa	**Acc-all**	89.4	89.4	89.0	89.0	89.2	89.3
	Acc-Bsent	29.7*	29.0	30.0	29.2	28.9	29.0
Eng	**Acc-all**	88.0	88.1	88.2	88.4	87.7	88.1
	Acc-Bsent	30.4	30.6	30.5	30.8	27.9	29.8

*p<.05

Table 10: Average accuracy on Targeted Sentiment Prediction: Identifying volitional entities and the polarity of the sentiment expressed towards them. The Spanish JOINT models significantly improve over their baseline for the core task. In English, no models outperform their baseline.

the sentiment polarity above their no sentiment baselines. We find this to be the most difficult task: It may be clear that sentiment is being expressed towards an entity, but it is not always clear what the polarity of that sentiment is. Error analysis is given below in this section. In the smaller English set, the models do not outperform the no sentiment baseline.

7 Discussion

Feature Analysis Examples of some of the top-weighted features in the Spanish models are shown in Table 11. In addition to lexical identity and Brown cluster, we find that positive indicators include positive suffixes such as diminutive forms, whether the word can be syllabized (Section 5), and whether it is three or four letters.

Error Analysis Because it is relatively common for there *not* to be sentiment targeted at a named entity, it is difficult to tease out the polarity in instances where there is targeted sentiment. Similarly, our predictions are most reliable for detecting the absence of a named entity (O labels).

Label confusions are shown in Table 12. Mistakes are often made by confusing B- labels (the start of an entity) with I- labels (inside an entity); and by predicting sentiment polarity when the gold annotations say there is not sentiment targeted at the entity. Some example errors are shown in Figure 13. In (1), "CANSADO" (*"TIRED"*) was predicted to be volitional, while "Matthew" was not. In (2), "Matias del río" was not predicted to be an entity, likely due to the fact that the capitalization patterns we see in this sentence are indicative of the start of a sentence rather than a proper name (similar to 1). In (3),

B-VOLITIONAL FEATURES	
Negative	is a function word; jerboa tags; followed by a word with 3 or 4 letters that cannot be syllabified
Positive	ends in -a, -o, or -s; is capitalized; has one non-initial capital letter; is 3 or 4 letters
B-VOLITIONAL, POS FEATURES	
Negative	preceded by a curse word; followed by a word with a positive suffix; immediately preceded by a word with a negative prefix
Positive	not in a sentiment lexicon; preceded by a happy emoticon; followed by an exclamation or a 'my' word; immediately preceded by a laugh; has two or more sentiment-bearing words in the sentence
B-VOLITIONAL, NEG FEATURES	
Negative	is immediately followed by a question mark or positive abbreviation word
Positive	preceded by a 'bad' word or curse word; has four or more sentiment lexicon items
B-VOLITIONAL, NOT-TARG FEATURES	
Negative	immediately followed by a 'no' word or word with a negative prefix; is preceded by a question mark; is immediately preceded by a curse word or laugh; is followed by an exclamation mark
Positive	not followed by sentiment lexicon word

Table 11: Example strongly weighted features for a Spanish joint sentiment model. In addition to lexical identity, we find that curse words and positive and negative prefixes are used to detect volitional entities and the sentiment directed towards them.

a.

Predicted \ Observed	B	I	O
B	423	21	186
I	36	236	135
O	197	90	7168

b.

Predicted \ Observed	POS	NEG	NEUT
POS	68	24	42
NEG	58	65	102
NEUT	115	61	468

Table 12: Predicted vs. observed values for a joint model. (**a**) For named entities, most common confusions were between B-VOLITIONAL and O labels. (**b**) For sentiment, most common mistakes were to predict that a positive sentiment was neutral (no sentiment), and that a neutral sentiment was negative.

NE prediction errors

1.

Spanish:	Cuando	estoy	CANSADO	,	él	es	mi	DESCANSO	.	Mateo	.	11	:	29	.						
Predicted:	O	O	B-VOLITIONAL	O	O	O	O	O		O	O			O	O	O	O	O			
Gold:	O	O	O	O	O	O	O	O		O	B-VOLITIONAL			O	O	O	O	O			
English:	*When*	*I'm*	*TIRED*	*,*	*he*	*is*	*my*	*REST*	*.*	*Matthew*	*.*	*11*	*:*	*29*	*.*						

2.

Spanish:	Matias	del	río	fue	una	lata	...
Predicted:	O	O	O	O	O	O	...
Gold:	B-VOLITIONAL	I-VOLITIONAL	I-VOLITIONAL	O	O	O	...
English:	*Matias*	*del*	*río*	*was*	*a*	*drag*	...

Sentiment prediction errors

3.

Spanish:	Mario	que	dio	este	contigo
Predicted:	NOT-TARG	-	-	-	-
Gold:	POSITIVE	-	-	-	-
English:	*Mario*	*may*	*God*	*be*	*with you*

4.

Spanish:	... si	de verdad	estas	en cielo	,	ayudame	Superman	!!!
Predicted:	-	- -	-	- -		- -	POSITIVE	-
Gold:	-	- -	-	- -		- -	NOT-TARG	-
English:	*... if*	*you really*	*are*	*in the skies*	*,*	*help me*	*Superman*	*!!!*

Sentiment and NE prediction errors

5.

Spanish:	Salen	del	gobierno	de	Humala	dos	connotados	izquierdistas	,	Giesecke	y	Eiguiguren			
Predicted:	O	O	O	O	B-VOLITIONAL	I-VOLITIONAL	O	O	O	B-VOLITIONAL	O	B-VOLITIONAL			
	-	-	-	-	NOT-TARG	NOT-TARG	-	-	-	NOT-TARG	-	NOT-TARG			
Gold:	O	O	O	O	B-VOLITIONAL	O	O	O	O	B-VOLITIONAL	O	B-VOLITIONAL			
	-	-	-	-	NOT-TARG	-	-	-	-	NEGATIVE	-	NOT-TARG			
English:	*Leaving*	*the*	*Humala*	*government*		*are two*	*notorious*	*leftists*	*,*	*Giesecke*	*and*	*Eiguiguren*			

Table 13: Example errors made by joint models.

sentiment may not be clear without spelling correction: "dio" should be "dios", meaning "God"; otherwise, "dio" is the word for "gave". Humans can easily fix the spelling error, which changes the overall reading of the expression. In (4), the positive polarity item "verdad" (*"believe"*) and the exclamation marks (!!!) were likely used as indicators of positive sentiment; however, in this case the annotators marked the targeted sentiment as neutral. In (5), the "Humala" entity was predicted to be longer than it is ("Hamala dos" or *"Hamala two"*). It was also predicted that both "Giesecke" and "Eiguiguren" had no sentiment expressed towards them; annotators disagreed, with the majority of those who annotated "Giesecke" marking negative sentiment, and the majority of those who annotated "Eiguiguren" marking no sentiment. This highlights some of the difficulty in predicting sentiment discussed in Section 3, where annotators will often disagree as to whether there is no sentiment or positive/negative sentiment.

During development, we found that the collapsed model (COLL) performed best on small amounts of data. However, as we scaled up the amount of data we trained on, the PIPE and JOINT models significantly improved, while the COLL models did not have significant performance gains.

8 Conclusion

We have introduced the task of open domain targeted sentiment: predicting sentiment directed towards an entity along with discovering the entity itself. Our approach is developed to find targeted sentiment towards both person and organization named entities by modeling sentiment as a span along the entity.

We find that by modeling targeted sentiment in this way, we can reliably detect entities and whether or not they are sentiment targets above a no sentiment baseline. How best to determine the *polarity* of the sentiment expressed towards the entity, however, is still an open issue. Our data suggests that it is usually not clear-cut whether sentiment is being expressed or not; the strong disagreement between annotators suggests that detecting sentiment polarity in microblogs is difficult even for humans.

In future work, we hope to explore further methods for teasing apart sentiment polarity expressed towards a target. This research has achieved promising results for detecting sentiment targets without relying on external supervised models, and we hope that the features and approaches developed here can aid in sentiment analysis in noisy text and languages without rich linguistic resources.

References

A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Language in Social Media*.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on Twitter from biased and noisy data. In *Proceedings of Coling: Posters*.

Adam Bermingham and Alan F Smeaton. 2010. Classifying sentiment in microblogs: Is brevity an advantage? In *Proceedings of CIKM-2010*.

Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in Twitter streaming data. In *Proceedings of the International Conference on Discovery Science (DS-2010)*.

Juliette Blevins. 1996. The syllable in phonological theory. In John A. Goldswmith, editor, *The Handbook of Phonological Theory*. Blackwell Publishing, Blackwell Reference Online.

N. N. Bora. 2012. Summarizing public opinions in tweets. In *Proceedings of CICLing-2012*.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs. In *Proceedings of EMNLP-2011*.

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

Pedro Henrique Calais Guerra, Adriano Veloso, Wagner Meira Jr, and Virgílio Almeida. 2011. From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the KDD-2011*.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL:HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Lu Chen, Wenbo Wang, Meenakshi Nagarajan, Shaojun Wang, and Amit P. Sheth. 2012. Extracting diverse sentiment expressions with target-dependent polarity from twitter. In *Proceedings of ICWSM-2012*.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. *Proceedings of EMNLP 2006*.

G. N. Clements. 1990. The role of the sonority cycle in core syllabification. In J. Kingston and M. Beckman, editors, *Papers in Laboratory Phonology*, pages 283–333. CUP, Cambridge.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using Twitter hashtags and smileys. In *Proceedings of Coling: Posters*.

Nicholas A Diakopoulos and David A Shamma. 2010. Characterizing debate performance via aggregated twitter sentiment. In *Proceedings of CHI-2010*.

David Etter, Francis Ferraro, Ryan Cotterell, Olivia Buzek, and Benjamin Van Durme. 2013. Nerit: Named entity recognition for informal text. Technical Report 11, Human Language Technology Center of Excellence, Johns Hopkins University, July.

Jenny Rose Finkel and Christopher D. Manning. 2010. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proceedings of ACL-2010*.

Joan B. Hooper. 1976. The syllable in phonological theory. *Language*, 48(3):525–540.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*.

Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM-2013)*.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of EMNLP*.

Long Jiang, Mo Yu, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL-2011*.

Wei Jin and Hung Hay Ho. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. *Proceedings of ICML 2009*.

Soo-Min Kim and Eduard Hovy. 2006. Identifying and analyzing judgment opinions. *Proceedings of NAACL 2006*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL/HLT*.

Michal Kosinskia, David Stillwell, and Thore Graepel. 2013. Private trains and attributes are predictable from digital records of human behavior. *Proc. of the National Academy of Sciences of the USA*, 110(5).

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *Proceedings of ICWSM-2011*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-2001*.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010a. Structure-aware review mining and summarization. *Proceedings of Coling 2010*.

Guangxia Li, Steven CH Hoi, Kuiyu Chang, and Ramesh Jain. 2010b. Micro-blogging sentiment detection by collaborative online learning. In *Proceedings of ICDM-2010*.

Hao Li, Yu Chen, Heng Ji, Smaranda Muresan, and Dequan Zheng. 2012. Combining social cognitive theories with linguistic features for multi-genre sentiment analysis. In *Proceedings of the Pacific Asia Conference on Language, Information and Computation (PACLIC-2012)*.

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

Frida Morelli. 2003. The relative harmony of /s+stop/ onsets: Obstruent clusters and the sonority sequencing principle. In C. Fery and R. van de Vijver, editors, *The syllable in optimality theory*, pages 356–371. CUP, New York.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC-2010*.

James W. Pennebaker, Roger J. Booth, and Martha E. Francis. 2007. Linguistic inquiry and word count: Liwc2007, operator's manual.

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. *Proceedings of the Conference on Language Resources and Evaluations (LREC 2012)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of Coling:ACL-2006*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT:EMNLP-2005*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1).

Hassan Saif, Yulan He, and Harith Alani. 2012. Alleviating data sparsity for twitter sentiment analysis. *Proceedings of the WWW Workshop on Making Sense of Microposts (# MSM2012)*.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the EMNLP-2011 Workshop on Unsupervised Learning in NLP*.

Veselin Stoyanov and Jason Eisner. 2012. Minimum-risk training of approximate crf-based nlp systems. In *Proceedings of NAACL:HLT-2012*.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. 2011. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *AIStats*.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of the KDD-2011*.

Benjamin Van Durme. 2012. Jerboa: A toolkit for randomized and streaming algorithms. Technical report, Human Language Technology Center of Excellence, Johns Hopkins University.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring sentiment in social media: Bootstrapping subjectivity clues from multilingual twitter streams. In *Association for Computational Linguistics (ACL)*.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in Twitter: A graph-based hashtag sentiment classification approach. In *Proceedings of CIKM-2011*.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffman. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3).

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. *Proceedings of ACL 2013*.

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of ICDM-2003*.

Exploiting Domain Knowledge in Aspect Extraction

Zhiyuan Chen, Arjun Mukherjee, Bing Liu
University of Illinois at Chicago
Chicago, IL 60607, USA
{czyuanacm,arjun4787}@gmail.com, liub@cs.uic.edu

Meichun Hsu, Malu Castellanos, Riddhiman Ghosh
HP Labs
Palo Alto, CA 94304, USA
{meichun.hsu, malu.castellanos, riddhiman.ghosh}@hp.com

Abstract

Aspect extraction is one of the key tasks in sentiment analysis. In recent years, statistical models have been used for the task. However, such models without any domain knowledge often produce aspects that are not interpretable in applications. To tackle the issue, some knowledge-based topic models have been proposed, which allow the user to input some prior domain knowledge to generate coherent aspects. However, existing knowledge-based topic models have several major shortcomings, e.g., little work has been done to incorporate the cannot-link type of knowledge or to automatically adjust the number of topics based on domain knowledge. This paper proposes a more advanced topic model, called *MC-LDA* (LDA with m-set and c-set), to address these problems, which is based on an *Extended generalized Pólya urn* (*E-GPU*) model (which is also proposed in this paper). Experiments on real-life product reviews from a variety of domains show that MC-LDA outperforms the existing state-of-the-art models markedly.

1 Introduction

In sentiment analysis and opinion mining, aspect extraction aims to extract entity aspects or features on which opinions have been expressed (Hu and Liu, 2004; Liu, 2012). For example, in a sentence "The picture looks great," the aspect is "picture." Aspect extraction consists of two sub-tasks: (1) extracting all *aspect terms* (e.g., "picture") from the corpus, and (2) clustering aspect terms with similar meanings (e.g., cluster "picture" and "photo" into one *aspect category* as they mean the same in the domain "Camera"). In this work, we adopt the topic modeling approach as it can perform both sub-tasks simultaneously (see § 2).

Topic models, such as LDA (Blei et al., 2003), provide an unsupervised framework for extracting latent topics in text documents. Topics are aspect categories (or simply aspects) in our context. However, in recent years, researchers have found that fully unsupervised topic models may not produce topics that are very coherent for a particular application. This is because the objective functions of topic models do not always correlate well with human judgments and needs (Chang et al., 2009).

To address the issue, several *knowledge-based topic models* have been proposed. The DF-LDA model (Andrzejewski et al., 2009) incorporates two forms of prior knowledge, also called two types of constraints: *must-links* and *cannot-links*. A must-link states that two words (or terms) should belong to the same topic whereas a cannot-link indicates that two words should not be in the same topic. In (Andrzejewski et al., 2011), more general knowledge can be specified using first-order logic. In (Burns et al., 2012; Jagarlamudi et al., 2012; Lu et al., 2011; Mukherjee and Liu, 2012), seeded models were proposed. They enable the user to specify prior knowledge as seed words/terms for some topics. Petterson et al. (2010) also used word similarity as priors for guidance.

However, none of the existing models is capable of incorporating the cannot-link type of knowledge except DF-LDA (Andrzejewski et al., 2009). Furthermore, none of the existing models, including DF-LDA, is able to automatically adjust the number of topics based on domain knowledge. The domain knowledge, such as cannot-links, may change the number of topics. There are two types of cannot-links: consistent and inconsistent with the domain corpus. For example, in the reviews of

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1655–1667,
Seattle, Washington, USA, 18-21 October 2013.

domain "Computer", a topic model may generate two topics *Battery* and *Screen* that represent two different aspects. A cannot-link {battery, screen} as the domain knowledge is thus consistent with the corpus. However, words *Amazon* and *Price* may appear in the same topic due to their high co-occurrences in the Amazon.com review corpus. To separate them, a cannot-link {amazon, price} can be added as the domain knowledge, which is inconsistent with the corpus as these two words have high co-occurrences in the corpus. In this case, the number of topics needs to be *increased* by 1 since the mixed topic has to be separated into two individual topics *Amazon* and *Price*. Apart from the above shortcoming, earlier knowledge-based topic models also have some major shortcomings:

Incapability of handling multiple senses: A word typically has multiple meanings or senses. For example, *light* can mean "of little weight" or "something that makes things visible." DF-LDA cannot handle multiple senses because its definition of must-link is transitive. That is, if A and B form a must-link, and B and C form a must-link, it implies a must-link between A and C, indicating A, B, and C should be in the same topic. This case also applies to the models in (Andrzejewski et al., 2011), (Petterson et al., 2010), and (Mukherjee and Liu, 2012). Although the model in (Jagarlamudi et al., 2012) allows multiple senses, it requires that each topic has at most one set of seed words (seed set), which is restrictive as the amount of knowledge should not be limited.

Sensitivity to the adverse effect of knowledge: When using must-links or seeds, existing models basically try to ensure that the words in a must-link or a seed set have similar probabilities under a topic. This causes a problem: if a must-link comprises of a frequent word and an infrequent word, due to the redistribution of probability mass, the probability of the frequent word will decrease while the probability of the infrequent word will increase. This can harm the final topics because the attenuation of the frequent (often domain important) words can result in some irrelevant words being ranked higher (with higher probabilities).

To address the above shortcomings, we define *m-set* (for *must-set*) as a set of words that should belong to the same topic and *c-set* (*cannot-set*) as a set of words that should not be in the same topic. They are similar to must-link and cannot-link but m-sets do not enforce transitivity. Transitivity is the main cause of the inability to handle multiple senses. Our m-sets and c-sets are also more concise providing knowledge in the context of a set. As in (Andrzejewski et al., 2009), we assume that there is no conflict between m-sets and c-sets, i.e., if w_1 is a cannot-word of w_2 (i.e., shares a c-set with w_2), any word that shares an m-set with w_1 is also a cannot-word of w_2. Note that knowledge as m-sets has also been used in (Chen et al., 2013a) and (Chen et al., 2013b).

We then propose a new topic model, called *MC-LDA* (LDA with m-set and c-set), which is not only able to deal with c-sets and automatically adjust the number of topics, but also deal with the multiple senses and adverse effect of knowledge problems at the same time. For the issue of multiple senses, a new latent variable s is added to LDA to distinguish multiple senses (§ 3). Then, we employ the *generalized Pólya urn* (GPU) model (Mahmoud, 2008) to address the issue of adverse effect of knowledge (§ 4). Deviating from the standard topic modeling approaches, we propose the *Extended generalized Pólya urn* (E-GPU) model (§ 5). E-GPU extends the GPU model to enable multi-urn interactions. This is necessary for handling c-sets and for adjusting the number of topics. E-GPU is the heart of MC-LDA. Due to the extension, a new inference mechanism is designed for MC-LDA (§ 6). Note that E-GPU is generic and can be used in any appropriate application.

In summary, this paper makes the following three contributions:

1. It proposed a new knowledge-based topic model called MC-LDA, which is able to use both m-sets and c-sets, as well as automatically adjust the number of topics based on domain knowledge. At the same time, it can deal with some other major shortcomings of early existing models. To our knowledge, none of the existing knowledge-based models is as comprehensive as MC-LDA in terms of capabilities.
2. It proposed the E-GPU model to enable multi-urn interactions, which enables c-sets to be naturally integrated into a topic model. To the best of our knowledge, E-GPU has not been proposed and used before.
3. A comprehensive evaluation has been conducted to compare MC-LDA with several state-of-the-art models. Experimental results based on both qualitative and quantitative measures demonstrate the superiority of MC-LDA.

2 Related Work

Sentiment analysis has been studied extensively in recent years (Hu and Liu, 2004; Pang and Lee, 2008; Wiebe and Riloff, 2005; Wiebe et al., 2004). According to (Liu, 2012), there are three main approaches to aspect extraction: 1) Using word frequency and syntactic dependency of aspects and sentiment words for extraction (e.g., Blair-goldensohn et al., 2008; Hu and Liu, 2004; Ku et al., 2006; Popescu and Etzioni, 2005; Qiu et al., 2011; Somasundaran and Wiebe, 2009; Wu et al., 2009; Yu et al., 2011; Zhang and Liu, 2011; Zhuang et al., 2006); 2) Using supervised sequence labeling/classification (e.g., Choi and Cardie, 2010; Jakob and Gurevych, 2010; Kobayashi et al., 2007; Li et al., 2010); 3) Topic models (Branavan et al., 2008; Brody and Elhadad, 2010; Fang and Huang, 2012; Jo and Oh, 2011; Kim et al., 2013; Lazaridou et al., 2013; Li et al., 2011; Lin and He, 2009; Lu et al., 2009, 2012, 2011; Lu and Zhai, 2008; Mei et al., 2007; Moghaddam and Ester, 2011; Mukherjee and Liu, 2012; Sauper et al., 2011; Titov and McDonald, 2008; Wang et al., 2010, 2011; Zhao et al., 2010). Other approaches include shallow semantic parsing (Li et al., 2012b), bootstrapping (Xia et al., 2009), Non-English techniques (Abu-Jbara et al., 2013; Zhou et al., 2012), graph-based representation (Wu et al., 2011), convolution kernels (Wiegand and Klakow, 2010) and domain adaption (Li et al., 2012). Stoyanov and Cardie (2011), Wang and Liu (2011), and Meng et al. (2012) studied opinion summarization outside the reviews. Some other works related with sentiment analysis include (Agarwal and Sabharwal, 2012; Kennedy and Inkpen, 2006; Kim et al., 2009; Mohammad et al., 2009).

In this work, we focus on topic models owing to their advantage of performing both aspect extraction and clustering simultaneously. All other approaches only perform extraction. Although there are several related works on clustering aspect terms (e.g., Carenini et al., 2005; Guo et al., 2009; Zhai et al., 2011), they all assume that the aspect terms have been extracted beforehand. We also notice that some aspect extraction models in sentiment analysis separately discover aspect words and aspect specific sentiment words (e.g., Sauper and Barzilay, 2013; Zhao et al., 2010). Our proposed model does not separate them as most sentiment words also imply aspects and most adjectives modify specific attributes of objects. For example, sentiment words *expensive* and *beautiful* imply aspects *price* and *appearance* respectively.

Regarding the knowledge-based models, besides those discussed in § 1, the model (Hu et al., 2011) enables the user to provide guidance interactively. Blei and McAuliffe (2007) and Ramage et al. (2009) used document labels in supervised setting. In (Chen et al., 2013a), we proposed MDK-LDA to leverage multi-domain knowledge, which serves as the basic mechanism to exploit m-sets in MC-LDA. In (Chen et al., 2013b), we proposed a framework (called GK-LDA) to explicitly deal with the wrong knowledge when exploring the lexical semantic relations as the general (domain independent) knowledge in topic models. But these models above did not consider the knowledge in the form of c-sets (or cannot-links).

The *generalized Pólya urn* (GPU) model (Mahmoud, 2008) was first introduced in LDA by Mimno et al. (2011). However, Mimno et al. (2011) did not use domain knowledge. Our results in § 7 show that using domain knowledge can significantly improve aspect extraction. The GPU model was also employed in topic models in our work of (Chen et al., 2013a, 2013b). In this paper, we propose the Extended GPU (E-GPU) model. The E-GPU model is more powerful in handling complex situations in dealing with c-sets.

3 Dealing with M-sets and Multiple Senses

Since the proposed MC-LDA model is a major extension to our earlier work in (Chen et al., 2013a), which can deal with m-sets, we include this earlier work here as the background.

To incorporate m-sets and deal with multiple senses of a word, the MDK-LDA(b) model was proposed in (Chen et al., 2013a), which adds a new latent variable s into LDA. The rationale here is that this new latent variable s guides the model to choose the right sense represented by an m-set. The generative process of MDK-LDA(b) is (the notations are explained in Table 1):

$$\theta \sim Dirichlet(\alpha)$$
$$z_i|\theta_m \sim Multinomial(\theta_m)$$
$$\varphi \sim Dirichlet(\beta)$$
$$s_i|z_i, \varphi \sim Multinomial(\varphi_{z_i})$$
$$\eta \sim Dirichlet(\gamma)$$
$$w_i|z_i, s_i, \eta \sim Multinomial(\eta_{z_i,s_i})$$

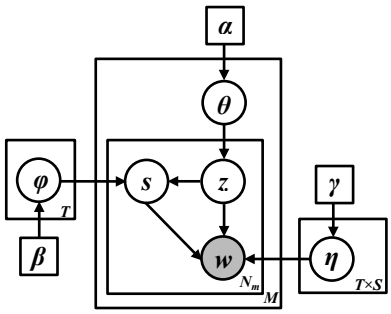


Figure 1. Plate notation for MDK-LDA(b) and MC-LDA.

The corresponding plate is shown in Figure 1. Under MDK-LDA(b), the probability of word w given topic t, i.e., $\pi_t(w)$, is given by:

$$\pi_t(w) = \sum_{s=1}^{S} \varphi_t(s) \cdot \eta_{t,s}(w) \quad (1)$$

where $\varphi_t(s)$ denotes the probability of m-set s occurring under topic t and $\eta_{t,s}(w)$ is the probability of word w appearing in m-set s under topic t.

According to (Chen et al., 2013a), the conditional probability of Gibbs sampler for MDK-LDA(b) is given by (see notations in Table 1):

$$P(z_i = t, s_i = s \,|\, \boldsymbol{z}^{-i}, \boldsymbol{s}^{-i}, \boldsymbol{w}, \alpha, \beta, \gamma) \propto \frac{n_{m,t}^{-i} + \alpha}{\sum_{t'=1}^{T}(n_{m,t'}^{-i} + \alpha)} \times \frac{n_{t,s}^{-i} + \beta}{\sum_{s'=1}^{S}(n_{t,s'}^{-i} + \beta)} \times \frac{n_{t,s,w_i}^{-i} + \gamma_s}{\sum_{v'=1}^{V}(n_{t,s,v'}^{-i} + \gamma_s)} \quad (2)$$

The superscript $-i$ denotes the counts excluding the current assignments (z_i and s_i) for word w_i.

4 Handling Adverse Effect of Knowledge

4.1 Generalized Pólya urn (GPU) Model

The Pólya urn model involves an urn containing balls of different colors. At discrete time intervals, balls are added or removed from the urn according to their color distributions.

In the simple Pólya urn (SPU) model, a ball is first drawn randomly from the urn and its color is recorded, then that ball is put back along with a new ball of the same color. This selection process is repeated and the contents of the urn change over time, with a self-reinforcing property sometimes expressed as "the rich get richer." SPU is actually exhibited in the Gibbs sampling for LDA.

The *generalized Pólya urn* (GPU) model differs from the SPU model in the replacement scheme during sampling. Specifically, when a ball is randomly drawn, certain numbers of additional balls of each color are returned to the urn, rather than just two balls of the same color as in SPU.

Hyperparameters	
α, β, γ	Dirichlet priors for $\theta,\ \varphi,\ \eta$
Latent & Visible Variables	
z	Topic (Aspect)
s	M-set
w	Word
θ	Document-Topic distribution
θ_m	Topic distribution of document m
φ	Topic-M-set distribution
φ_t	M-set distribution of topic t
η	Topic-M-set-Word distribution
$\eta_{t,s}$	Word distribution of topic t, m-set s
Cardinalities	
M	Number of documents
N_m	Number of words in document m
T	Number of topics
S	Number of m-sets
V	The vocabulary size
Sampling & Count Notations	
z_i	Topic assignment for word w_i
s_i	M-set assignment for word w_i
$\boldsymbol{z}^{-i}$	Topic assignments for all words except w_i
$\boldsymbol{s}^{-i}$	M-set assignments for all words except w_i
$n_{m,t}$	Number of times that topic t is assigned to word tokens in document m
$n_{t,s}$	Number of times that m-set s occurs under topic t
$n_{t,s,w}$	Number of times that word w appears in m-set s under topic t

Table 1. Meanings of symbols.

4.2 Promoting M-sets using GPU

To deal with the issue of sensitivity to the adverse effect of knowledge, MDK-LDA(b) is extended to MDK-LDA which employs the generalized Pólya urn (GPU) sampling scheme**.**

As discussed in § 1, due to the problem of the adverse effect of knowledge, important words may suffer from the presence of rare words in the same m-set. This problem can be dealt with the very sampling scheme of the GPU model (Chen et al., 2013a). Specifically, by adding additional $\mathbb{A}_{s,w',w}$ balls of color s into U_t^S while keeping the drawn ball, we increase the proportion (probability) of seeing the m-set s under topic t and thus promote m-set s as a whole. Consequently, each word in s is more likely to be emitted. We define $\mathbb{A}_{s,w',w}$ as:

$$\mathbb{A}_{s,w',w} = \begin{cases} 1 & w = w' \\ \sigma & w \in s, w' \in s, w \neq w' \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

The corresponding Gibbs sampler for MDK-LDA will be introduced in § 6.

5 Incorporating C-sets

5.1 Extended Generalized Pólya urn Model

To handle the complex situation resulted from incorporating c-sets, we propose an *Extended generalized Pólya urn* (E-GPU) model. Instead of involving only one urn as in SPU and GPU, E-GPU model considers a set of urns in the sampling process. The E-GPU model allows a ball to be transferred from one urn to another, enabling multi-urn interactions. Thus, during sampling, the populations of several urns will evolve even if only one ball is drawn from one urn. This capability makes the E-GPU model more powerful as it models relationships among multiple urns.

We define three sets of urns which will be used in the new sampling scheme in the proposed MC-LDA model. The first set of urns is the topic urns $U^T_{m\in\{1\dots M\}}$, where each topic urn contains T colors (topics) and each ball inside has a color $t \in \{1 \dots T\}$. It corresponds to the document-topic distribution θ in Table 1. The second set of urns (m-set urn $U^S_{t\in\{1\dots T\}}$) corresponds to the topic-m-set distribution φ, with balls of colors (m-sets) $s \in \{1 \dots S\}$ in each m-set urn. The third set of urns is the word urns $U^W_{t,s}$,where $t \in \{1 \dots T\}$ and $s \in \{1 \dots S\}$. Each ball inside a word urn has a color (word) $w \in \{1 \dots V\}$. The distribution η can be reflected in this set of urns.

5.2 Handling C-sets using E-GPU

As MDK-LDA can only use m-sets but not c-sets, we now extend MDK-LDA to the MC-LDA model in order to exploit c-sets. As pointed out in § 1, c-sets may be inconsistent with the corpus domain, which makes them considerably harder to deal with. To tackle the issue, we utilize the proposed E-GPU model and incorporate c-sets handling inside the E-GPU sampling scheme, which is also designed to enable automated adjustment of the number of topics based on domain knowledge.

Based on the definition of c-set, each pair of words in a c-set cannot both have large probabilities under the same topic. As the E-GPU model allows multi-urn interactions, when sampling a ball represents word w from a word urn $U^W_{t,s}$, we want to transfer the balls representing cannot-words of w (sharing a c-set with w) to other urns (see Step 3 a below). That is, decrease the probabilities of those cannot-words under this topic while increasing their corresponding probabilities under some other topics. In order to correctly transfer a ball that represents word w, it should be transferred to an urn which has a higher proportion of w and its related words (i.e., words sharing m-sets with w). That is, we randomly sample an urn that has a higher proportion of any m-set of w to transfer w to (Step 3 b below). However, the situation becomes more involved when a c-set is not consistent with the corpus. For example, aspects *price* and *amazon* may be mixed under one topic (say t) in LDA. The user may want to separate them by providing a c-set {price, amazon}. In this case, according to LDA, word *price* has no topic with a higher proportion of it (and its related words) than topic t. To transfer it, we need to increment the number of topics by 1 and then transfer the word to this new topic urn (step 3 c below). Based on these ideas, we propose the E-GPU sampling scheme for the MC-LDA model below:

1. Sample a topic t from U^T_m, an m-set s from U^S_t, and a word w from $U^W_{t,s}$ sequentially, where m is the mth document.
2. Record t, s and w, put back two balls of color t into urn U^T_m, one ball of color s into urn U^S_t, and two balls of color w into urn $U^W_{t,s}$. Given the matrix $\mathbb{A}$ (in Equation 3), for each word $w' \in s$, we put back $\mathbb{A}_{s,w',w}$ number of balls of color s into urn U^S_t.
3. For each word w_c that shares a c-set with w:
   a) Sample an m-set s_c from U^S_t which satisfies $w_c \in s_c$. Draw a ball b of color w_c (to be transferred) from U^W_{t,s_c} and remove it from U^W_{t,s_c}. The document of ball b is denoted by m_c. If no ball of color w_c can be drawn (i.e., there is no ball of color w_c in U^W_{t,s_c}), skip steps b) to d).
   b) Produce an urn set $\{U^W_{t',s'}\}$ such that each urn in it satisfies the following conditions:
      i) $t' \neq t, w_c \in s'$
      ii) The proportion of balls of color s' in $U^S_{t'}$ is higher than that of balls of color s_c in U^S_t.
   c) If $\{U^W_{t',s'}\}$ is not empty, randomly select one urn $U^W_{t',s'}$ from it. If $\{U^W_{t',s'}\}$ is empty, set $T = T + 1, t' = T$, draw an m-set s' from $U^S_{t'}$ which satisfies $w_c \in s'$. Record s' for step d).
   d) Put the ball b drawn from Step a) into $U^W_{t',s'}$, as well as a ball of color s' into $U^S_{t'}$ and a ball of color t' into $U^T_{m_c}$.

Note that the E-GPU model cannot be reflected in the graphical model in Figure 1 as it is essentially

Algorithm 1. GibbsSampling(m, w_i, $\mathbb{A}$, μ, Ω)

Input: Document m, Word w_i, Matrix $\mathbb{A}$,
Transfer cannot-word flag μ,
A set of valid topics Ω to be assigned to w_i

1: $n_{m,z_i} \leftarrow n_{m,z_i} - 1$;
2: **for** each word w' in s_i **do**
3: $n_{z_i,s_i} \leftarrow n_{z_i,s_i} - \mathbb{A}_{s_i,w',w_i}$;
4: **end for**
5: $n_{z_i,s_i,w_i} \leftarrow n_{z_i,s_i,w_i} - 1$;
6: Jointly sample $z_i \in \Omega$ and $s_i \ni w_i$ using Equation 2;
7: $n_{m,z_i} \leftarrow n_{m,z_i} + 1$;
8: **for** each word w' in s_i **do**
9: $n_{z_i,s_i} \leftarrow n_{z_i,s_i} + \mathbb{A}_{s_i,w',w_i}$;
10: **end for**
11: $n_{z_i,s_i,w_i} \leftarrow n_{z_i,s_i,w_i} + 1$;
12: **if** μ is true **then**
13: TransferCannotWords(w_i, z_i);
14: **end if**

Figure 2. Gibbs sampling for MC-LDA.

sampling scheme, and hence MC-LDA shares the same plate as MDK-LDA(b).

6 Collapsed Gibbs Sampling

We now describe the collapsed Gibbs sampler (Griffiths and Steyvers, 2004) with the detailed conditional distributions and algorithms for MC-LDA. Inference of z and s can be computationally expensive due to the non-exchangeability of words under the E-GPU models. We take the approach of (Mimno et al., 2011) which approximates the true Gibbs sampling distribution by treating each word as if it were the last.

For each word w_i, we perform hierarchical sampling consisting of the following three steps (the detailed algorithms are given in Figures 2 and 3):

Step 1 (Lines 1-11 in Figure 2): We jointly sample a topic z_i and an m-set s_i (containing w_i) for w_i, which gives us a blocked Gibbs sampler (Ishwaran and James, 2001), with the conditional probability given by:

$$P(z_i = t, s_i = s | \boldsymbol{z}^{-i}, \boldsymbol{s}^{-i}, \boldsymbol{w}, \alpha, \beta, \gamma, \mathbb{A}) \propto \frac{n_{m,t}^{-i} + \alpha}{\sum_{t'=1}^{T}\left(n_{m,t'}^{-i} + \alpha\right)} \times \frac{\sum_{w'=1}^{V}\sum_{v'=1}^{V} \mathbb{A}_{s,v',w'} \cdot n_{t,s,v'}^{-i} + \beta}{\sum_{s'=1}^{S}\left(\sum_{w'=1}^{V}\sum_{v'=1}^{V} \mathbb{A}_{s',v',w'} \cdot n_{t,s',v'}^{-i} + \beta\right)} \times \frac{n_{t,s,w_i}^{-i} + \gamma_s}{\sum_{v'=1}^{V}\left(n_{t,s,v'}^{-i} + \gamma_s\right)} \quad (4)$$

This step is the same as the Gibbs sampling for the MDK-LDA model.

Algorithm 2.TransferCannotWords(w_i, z_i)

Input: Word w_i, Topic z_i,

1: **for** each cannot-word w_c of w_i **do**
2: Randomly select an m-set s_c from all m-sets of w_c;
3: Build a set Ψ containing all the instances of w_c from the corpus with topic and m-set assignments being z_i and s_c;
4: **if** Ψ is not empty **then**
5: Draw an instance of w_c from Ψ (denoting the document of this instance by m_c) using Equation 5;
6: Generate a topic set Ω' that each topic t' inside satisfies $max_{s' \ni w_c}(\varphi_{t'}(s')) > \varphi_{z_i}(s_c)$.
7: **if** Ω' is not empty **then**
8: GibbsSampling(m_c, w_c, $\mathbb{A}$, false, Ω');
9: **else**
10: $Dummy = T + 1$; // T is #Topics.
11: GibbsSampling(m_c, w_c, $\mathbb{A}$, false, $\{Dummy\}$);
12: **end if**
13: **end if**
14: **end for**

Figure 3. Transfer cannot-words in Gibbs sampling.

Step 2 (lines 1-5 in Figure 3): For every cannot-word (say w_c) of w_i, randomly pick an urn U_{z_i,s_c}^{W} from the urn set $\{U_{z_i,\bar{s}}^{W}\}$ where $\bar{s} \ni w_c$. If there exists at least one ball of color w_c in urn U_{z_i,s_c}^{W}, we sample one ball (say b_c) of color w_c from urn U_{z_i,s_c}^{W}, based on the following conditional distribution:

$$P(b = b_c | \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{w}, \alpha, \beta, \gamma, \mathbb{A}) \propto \frac{n_{m_c,t} + \alpha}{\sum_{t'=1}^{T}\left(n_{m_c,t'} + \alpha\right)} \quad (5)$$

where m_c denotes the document of the ball b_c of color w_c.

Step 3 (lines 6-12 in Figure 3): For each drawn ball b from Step 2, resample a topic t and an m-set s (containing w_c) based on the following conditional distribution:

$$\begin{aligned} & P(z_b = t, s_b = s | \boldsymbol{z}^{-b}, \boldsymbol{s}^{-b}, \boldsymbol{w}, \alpha, \beta, \gamma, \mathbb{A}, b = b_c) \\ & \propto \mathbf{I}_{\left[0, \max\limits_{s' \ni w_c}(\varphi_t(s'))\right]}\left(\varphi_{z_c}(s_c)\right) \times \frac{n_{m,t}^{-b} + \alpha}{\sum_{t'=1}^{T}\left(n_{m,t'}^{-b} + \alpha\right)} \\ & \times \frac{\sum_{w'=1}^{V}\sum_{v'=1}^{V} \mathbb{A}_{s,v',w'} \cdot n_{t,s,v'}^{-b} + \beta}{\sum_{s'=1}^{S}\left(\sum_{w'=1}^{V}\sum_{v'=1}^{V} \mathbb{A}_{s',v',w'} \cdot n_{t,s',v'}^{-b} + \beta\right)} \\ & \times \frac{n_{t,s,w_b}^{-b} + \gamma_s}{\sum_{v'=1}^{V}\left(n_{t,s,v'}^{-b} + \gamma_s\right)} \end{aligned} \quad (6)$$

where z_c (same as z_i in Figure 3) and s_c are the original topic and m-set assignments. The superscript $-b$ denotes the counts excluding the original

assignments. **I**() is an indicator function, which restricts the ball to be transferred only to an urn that contains a higher proportion of its m-set. When no topic t can be successfully sampled and the current sweep (iteration) of Gibbs sampling has the same number of topic (T) as the previous sweep, we increment T by 1. And then assign T to z_b. The counts and parameters are also updated accordingly.

7 Experiments

We now evaluate the proposed MC-LDA model and compare it with state-of-the-art existing models. Two unsupervised baseline models that we compare with are:

- **LDA**: LDA is the basic unsupervised topic model (Blei et al., 2003).
- **LDA-GPU**: LDA with GPU (Mimno et al., 2011). Specifically, LDA-GPU applies GPU in LDA using co-document frequency.

As for knowledge-based models, we focus on comparing with DF-LDA model (Andrzejewski et al., 2009), which is perhaps the best known knowledge-based model and it allows both must-links and cannot-links.

For a comprehensive evaluation, we consider the following variations of MC-LDA and DF-LDA:

- **MC-LDA:** MC-LDA with both m-sets and c-sets. This is the newly proposed model.
- **M-LDA**: MC-LDA with m-sets only. This is the MDK-LDA model in (Chen et al., 2013a).
- **DF-M**: DF-LDA with must-links only.
- **DF-MC**: DF-LDA with both must-links and cannot-links. This is the full DF-LDA model in (Andrzejewski et al., 2009).

We do not compare with seeded models in (Burns et al., 2012; Jagarlamudi et al., 2012; Lu et al., 2011; Mukherjee and Liu, 2012) as seed sets are special cases of must-links and they also do not allow c-sets (or cannot-links).

7.1 Datasets and Settings

Datasets: We use product reviews from four domains (types of products) from Amazon.com for evaluation. The corpus statistics are shown in Table 2 (columns 2 and 3). The domains are "Camera," "Food," "Computer," and "Care" (short for "Personal Care"). We have made the datasets publically available at the website of the first author.

Domain	#Reviews	#Sentences	#M-sets	#C-sets
Camera	500	5171	173	18
Food	500	2416	85	10
Computer	500	2864	92	6
Care	500	3008	119	13
Average	500	3116	103	9

Table 2. Corpus statistics with #m-sets and #c-sets having at least two words.

Pre-processing: We ran the Stanford Core NLP Tools[1] to perform sentence detection and lemmatization. Punctuations, stopwords[2], numbers and words appearing less than 5 times in each corpus were removed. The domain name was also removed, e.g., word *camera* in the domain "Camera", since it co-occurs with most words in the corpus, leading to high similarity among topics/aspects.

Sentences as documents: As noted in (Titov and McDonald, 2008), when standard topic models are applied to reviews as documents, they tend to produce topics that correspond to global properties of products (e.g., brand name), which make topics overlapping with each other. The reason is that all reviews of the same type of products discuss about the same aspects of these products. Only the brand names and product names are different. Thus, using individual reviews for modeling is not very effective. Although there are approaches which model sentences (Jo and Oh, 2011; Titov and McDonald, 2008), we take the approach of (Brody and Elhadad, 2010), dividing each review into sentences and treating each sentence as an independent document. Sentences can be used by all three baselines without any change to their models. Although the relationships between sentences are lost, the data is fair to all models.

Parameter settings: For all models, posterior inference was drawn using 1000 Gibbs iterations with an initial burn-in of 100 iterations. For all models, we set $\alpha = 1$ and $\beta = 0.1$. We found that small changes of α and β did not affect the results much, which was also reported in (Jo and Oh, 2011) who also used online reviews. For the number of topics T, we tried different values (see §7.2) as it is hard to know the exact number of topics. While non-parametric Bayesian approaches (Teh et al., 2006) aim to estimate T from the corpus, they are often sensitive to the hyper-parameters (Heinrich, 2009).

[1] http://nlp.stanford.edu/software/corenlp.shtml
[2] http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list

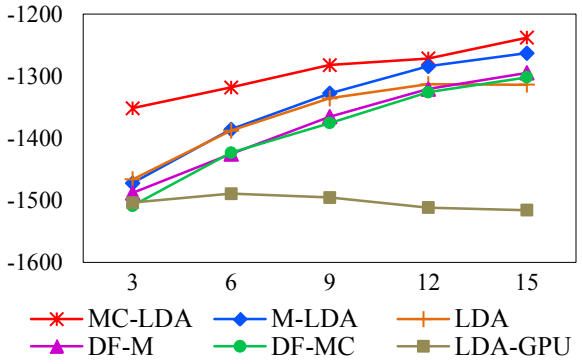


Figure 4. Avg. Topic Coherence score of each model across different number of topics.

For DF-LDA, we followed (Andrzejewski et al., 2009) to generate must-links and cannot-links from our domain knowledge. We then ran DF-LDA[3] while keeping its parameters as proposed in (Andrzejewski et al., 2009) (we also experimented with different parameter settings but they did not produce better results). For our proposed model, we estimated the thresholds using cross validation in our pilot experiments. Estimated value $\sigma = 0.2$ in equation 3 yielded good results. The second stage (steps 2 and 3) of the Gibbs sampler for MC-LDA (for dealing with c-sets) is applied after burn-in phrase.

Domain knowledge: User knowledge about a domain can vary a great deal. Different users may have very different knowledge. To reduce this variance for a more reliable evaluation, instead of asking a human user to provide m-sets, we obtain the synonym sets and the antonym sets of each word that is a noun or adjective (as words of other parts-of-speech usually do not indicate aspects) from WordNet (Miller, 1995) and manually verify the words in those sets for the domain. Note that if a word w is not provided with any m-set, it is treated as a singleton m-set $\{w\}$. For c-sets, we ran LDA in each domain and provide c-sets based on the wrong results of LDA as in (Andrzejewski et al., 2009). Then, the knowledge is provided to each model in the format required by each model. The numbers of m-sets and c-sets are listed in columns 4 and 5 of Table 2. Duplicate sets have been removed.

7.2 Objective Evaluation

In this section, we evaluate our proposed MC-LDA model objectively. Topic models are often evaluated using perplexity on held-out test data. However, the perplexity metric does not reflect the semantic coherence of individual topics learned by a topic model (Newman et al., 2010). Recent research has shown potential issues with perplexity as a measure: (Chang et al., 2009) suggested that the perplexity can sometimes be contrary to human judgments. Also, perplexity does not really reflect our goal of finding coherent aspects with accurate semantic clustering. It only provides a measure of how well the model fits the data.

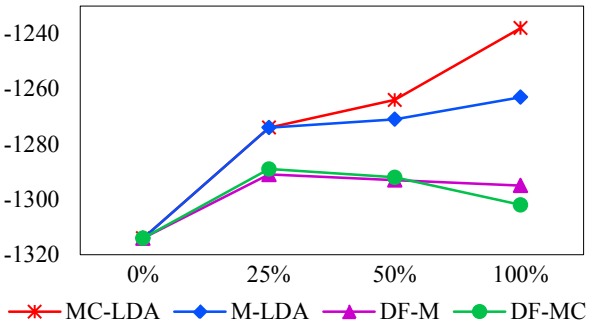


Figure 5. Avg. Topic Coherence score for different proportions of knowledge.

The *Topic Coherence* metric (Mimno et al., 2011) (also called the "UMass" measure (Stevens and Buttler, 2012)) was proposed as a better alternative for assessing topic quality. This metric relies upon word co-occurrence statistics within the documents, and does not depend on external resources or human labeling. It was shown that topic coherence is highly consistent with human expert labeling by Mimno et al. (2011). Higher topic coherence score indicates higher quality of topics, i.e., better topic interpretability.

Effects of Number of Topics

Since our proposed models and the baseline models are all parametric models, we first compare each model given different numbers of topics. Figure 4 shows the average Topic Coherence score of each model given different numbers of topics. From Figure 4, we note the following:

1. MC-LDA consistently achieves the highest Topic Coherence scores given different numbers of topics. M-LDA also works better than the other baseline models, but not as well as MC-LDA. This shows that both m-sets and c-sets are beneficial in producing coherent aspects.
2. DF-LDA variants, DF-M and DF-MC, do not perform well due to the shortcomings discussed

[3] http://pages.cs.wisc.edu/~andrzeje/research/df_lda.html

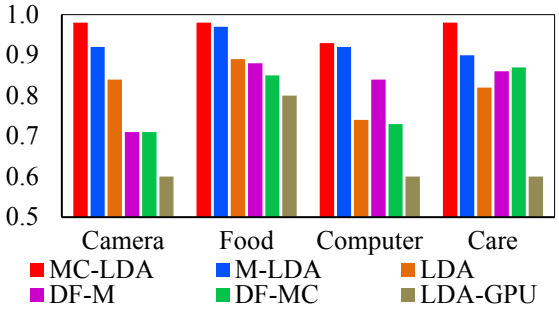


Figure 6. Avg. *p@5* of good topics for each model across different domains.

The models of each bar from left to rights are MC-LDA, M-LDA, LDA, DF-M, DF-MC, LDA-GPU. (Same for Figure 7)

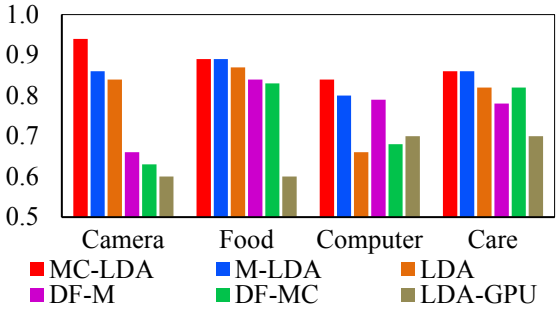


Figure 7. Avg. *p@10* of good topics for each model across different domains.

in § 1. It is slightly better than LDA when $T = 15$, but worse than LDA in other cases. We will further analyze the effects of knowledge on MC-LDA and DF-LDA shortly.

3. LDA-GPU does not perform well due to its use of co-document frequency. As frequent words usually have high co-document frequency with many other words, the frequent words are ranked top in many topics. This shows that the guidance using domain knowledge is more effective than using co-document frequency.

In terms of improvements, MC-LDA outperforms M-LDA significantly ($p < 0.03$) and all other baseline models significantly ($p < 0.01$) based on a paired *t*-test. It is important to note that by no means do we say that LDA-GPU and DF-LDA are not effective. We only say that for the task of aspect extraction and leveraging domain knowledge, these models do not generate as coherent aspects as ours because of their shortcomings discussed in § 1. In general, with more topics, the Topic Coherence scores increase. We found that when T is larger than 15, aspects found by each model became more and more overlapping, with several aspects expressing the same features of products. So we fix $T = 15$ in the subsequent experiments.

Effects of Knowledge

To further analyze the effects of knowledge on models, in each domain, we randomly sampled different proportions of knowledge (i.e., different numbers of m-sets/must-links and c-sets/cannot-links) as shown in Figure 5, where 0% means no knowledge (same as LDA and LDA-GPU, which do not incorporate knowledge) and 100% means all knowledge. From Figure 5, we see that MC-LDA and M-LDA both perform consistently better than DF-MC and DF-M across different proportions of knowledge. With the increasing number of knowledge sets, MC-LDA and M-LDA achieve higher Topic Coherence scores (i.e., produce more coherent aspects). In general, MC-LDA performs the best. For both DF-MC and DF-M, the Topic Coherence score increases from 0% to 25% knowledge, but decreases with more knowledge (50% and 100%). This shows that with limited amount of knowledge, the shortcomings of DF-LDA are not very obvious, but with more knowledge, these issues become more serious and thus degrade the performance of DF-LDA.

7.3 Human Evaluation

Since our aim is to make topics more interpretable and conformable to human judgments, we worked with two judges who are familiar with Amazon products and reviews to evaluate the models subjectively. Since topics from topic models are rankings based on word probability and we do not know the number of correct topical words, a natural way to evaluate these rankings is to use *Precision@n* (or *p@n*) which was also used in (Mukherjee and Liu, 2012; Zhao et al., 2010), where *n* is the rank position. We give *p@n* for $n = 5$ and 10. There are two steps in human evaluation: topic labeling and word labeling.

Topic Labeling: We followed the instructions in (Mimno et al., 2011) and asked the judges to label each topic as *good* or *bad*. Each topic was presented as a list of 10 most probable words in descending order of their probabilities under that topic. The models which generated the topics for labeling were obscure to the judges. In general, each topic was annotated as *good* if it had more than half of its words coherently related to each other representing a semantic concept together; otherwise *bad*. Agreement of human judges on topic

labeling using Cohen's Kappa yielded a score of 0.92 indicating almost perfect agreements according to the scale in (Landis and Koch, 1977). This is reasonable as topic labeling is an easy task and semantic coherence can be judged well by humans.

Word Labeling: After topic labeling, we chose the topics, which were labeled as good by both judges, as good topics. Then, we asked the two judges to label each word of the top 10 words in these good topics. Each word was annotated as *correct* if it was coherently related to the concept represented by the topic; otherwise *incorrect*. Since judges already had the conception of each topic in mind when they were labeling topics, labeling each word was not difficult which explains the high Kappa score for this labeling task (score = 0.892).

Quantitative Results

Figures 6 and 7 give the average *p@5* and *p@10* of all good topics over all four domains. The numbers of good topics generated by each model are shown in Table 3. We can see that the human evaluation results are highly consistent with Topic Coherence results in §7.2. MC-LDA improves over M-LDA significantly ($p < 0.01$) and both MC-LDA and M-LDA outperforms the other baseline models significantly ($p < 0.005$) using a paired *t*-test. We also found that when the domain knowledge is simple with one word usually expressing only one meaning/sense (e.g., in the domain "Computer"), DF-LDA performs better than LDA. In other domains, it performs similarly or worse than LDA. Again, it shows that DF-LDA is not effective to handle complex knowledge, which is consistent with the results of effects of knowledge on DF-LDA in §7.2.

Qualitative Results

We now show some qualitative results to give an intuitive feeling of the outputs from different models. There are a large number of aspects that are dramatically improved by MC-LDA. Due to space constraints, we only show some examples. To further focus, we just show some results of MC-LDA, M-LDA and LDA. The results from LDA-GPU and DF-LDA were inferior and hard for the human judges to match them with aspects found by the other models for qualitative comparison.

Table 4 shows three aspects Amazon, Price, Battery generated by each model in the domain

#Good Topics	MC-LDA	M-LDA	LDA	DF-M	DF-MC	LDA-GPU
Camera	**15/18**	12	11	9	7	3
Food	**8/16**	7	7	5	4	5
Computer	**12/16**	10	7	9	6	4
Care	**11/16**	10	9	10	9	3
Average	**11.5/16.5**	9.75/15	8.5/15	8.25/15	6.5/15	3.75/15

Table 3. Number of good topics of each model.

In x/y, x is the number of discovered good topics, and y is the total number of topics generated.

MC-LDA			M-LDA		LDA	
Amazon	**Price**	**Battery**	**Price**	**Battery**	**Amazon**	**Battery**
review	price	battery	price	battery	*card*	battery
amazon	*perform*	life	*lot*	*review*	*day*	*screen*
software	money	*day*	money	*amazon*	amazon	life
customer	expensive	extra	*big*	life	*memory*	*lcd*
month	cost	charger	expensive	extra	product	*water*
support	*week*	*water*	*point*	*day*	*sd*	usb
warranty	cheap	time	cost	power	*week*	*cable*
package	purchase	power	*photo*	time	*month*	*case*
product	deal	hour	*dot*	*support*	item	charger
hardware	product	aa	purchase	*customer*	*class*	hour

Table 4. Example aspects in the domain "Camera"; errors are marked in red/italic.

"Camera". Both LDA and M-LDA can only discover two aspects but M-LDA has a higher average precision. Given the c-set {amazon, price, battery}, MC-LDA can discover all three aspects with the highest average precision.

8 Conclusion

This paper proposed a new model to exploit domain knowledge in the form of m-sets and c-sets to generate coherent aspects (topics) from online reviews. The paper first identified and characterized some shortcomings of the existing knowledge-based models. A new model called MC-LDA was then proposed, whose sampling scheme was based on the proposed Extended GPU (E-GPU) model enabling multi-urn interactions. A comprehensive evaluation using real-life online reviews from multiple domains shows that MC-LDA outperforms the state-of-the-art models significantly and discovers aspects with high semantic coherence. In our future work, we plan to incorporate aspect specific sentiments in the MC-LDA model.

Acknowledgments

This work was supported in part by a grant from National Science Foundation (NSF) under grant no. IIS-1111092, and a grant from HP Labs Innovation Research Program.

References

Amjad Abu-Jbara, Ben King, Mona Diab, and Dragomir Radev. 2013. Identifying Opinion Subgroups in Arabic Online Discussions. In *Proceedings of ACL*.

Apoorv Agarwal and Jasneet Sabharwal. 2012. End-to-End Sentiment Analysis of Twitter Data. In *Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data, at the 24th International Conference on Computational Linguistics (IEEASMD-COLING 2012)*, Vol. 2.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet Forest priors. In *Proceedings of ICML*, pages 25–32.

David Andrzejewski, Xiaojin Zhu, Mark Craven, and Benjamin Recht. 2011. A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *Proceedings of IJCAI*, pages 1171–1177.

Sasha Blair-goldensohn, Tyler Neylon, Kerry Hannan, George A. Reis, Ryan Mcdonald, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *Proceedings of In NLP in the Information Explosion Era*.

David M. Blei and Jon D. McAuliffe. 2007. Supervised Topic Models. In *Proceedings of NIPS*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

S. R. K. Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. 2008. Learning Document-Level Semantic Properties from Free-Text Annotations. In *Proceedings of ACL*, pages 263–271.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Proceedings of NAACL*, pages 804–812.

Nicola Burns, Yaxin Bi, Hui Wang, and Terry Anderson. 2012. Extended Twofold-LDA Model for Two Aspects in One Sentence. *Advances in Computational Intelligence*, Vol. 298, pages 265–275. Springer Berlin Heidelberg.

Giuseppe Carenini, Raymond T. Ng, and Ed Zwart. 2005. Extracting knowledge from evaluative text. In *Proceedings of K-CAP*, pages 11–18.

Jonathan Chang, Jordan Boyd-Graber, Wang Chong, Sean Gerrish, and David Blei, M. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Proceedings of NIPS*, pages 288–296.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Leveraging Multi-Domain Prior Knowledge in Topic Models. In *Proceedings of IJCAI*, pages 2071–2077.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013b. Discovering Coherent Topics Using General Knowledge. In *Proceedings of CIKM*.

Yejin Choi and Claire Cardie. 2010. Hierarchical Sequential Learning for Extracting Opinions and their Attributes, pages 269–274.

Lei Fang and Minlie Huang. 2012. Fine Granular Aspect Analysis using Latent Structural Models. In *Proceedings of ACL*, pages 333–337.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding Scientific Topics. *PNAS*, *101 Suppl*, 5228–5235.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, and Zhong Su. 2009. Product feature categorization with multilevel latent semantic association. In *Proceedings of CIKM*, pages 1087–1096.

Gregor Heinrich. 2009. A Generic Approach to Topic Models. In *Proceedings of ECML PKDD*, pages 517 – 532.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of KDD*, pages 168–177.

Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. 2011. Interactive Topic Modeling. In *Proceedings of ACL*, pages 248–257.

Hemant Ishwaran and LF James. 2001. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, *96*(453), 161–173.

Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. 2012. Incorporating Lexical Priors into Topic Models. In *Proceedings of EACL*, pages 204–213.

Niklas Jakob and Iryna Gurevych. 2010. Extracting Opinion Targets in a Single- and Cross-Domain Setting with Conditional Random Fields. In *Proceedings of EMNLP*, pages 1035–1045.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of WSDM*, pages 815–824.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment Classification of Movie Reviews Using Contextual Valence Shifters. *Computational Intelligence*, *22*(2), 110–125.

Jungi Kim, Jinji Li, and Jong-Hyeok Lee. 2009. Discovering the Discriminative Views: Measuring Term Weights for Sentiment Analysis. In *Proceedings of ACL/IJCNLP*, pages 253–261.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice Oh, and Shixia Liu. 2013. A Hierarchical Aspect-Sentiment Model for Online Reviews. In *Proceedings of AAAI*.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting Aspect-Evaluation and Aspect-of Relations in Opinion Mining. In *Proceedings of EMNLP*, pages 1065–1074.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion Extraction, Summarization and Tracking in

News and Blog Corpora. In *Proceedings of AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 100–107.

JR Landis and GG Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, *33*.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A Bayesian Model for Joint Unsupervised Induction of Sentiment, Aspect and Discourse Representations. In *Proceedings of ACL*.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Yingju Xia, Shu Zhang, and Hao Yu. 2010. Structure-Aware Review Mining and Summarization. In *Proceedings of COLING*, pages 653–661.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012a. Cross-Domain Co-Extraction of Sentiment and Topic Lexicons. In *Proceedings of ACL (1)*, pages 410–419.

Peng Li, Yinglin Wang, Wei Gao, and Jing Jiang. 2011. Generating Aspect-oriented Multi-Document Summarization with Event-aspect model. In *Proceedings of EMNLP*, pages 1137–1146.

Shoushan Li, Rongyang Wang, and Guodong Zhou. 2012b. Opinion Target Extraction Using a Shallow Semantic Parsing Framework. In *Proceedings of AAAI*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*, pages 375–384.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K. Tsou. 2011. Multi-aspect Sentiment Analysis with Topic Models. In *Proceedings of ICDM Workshops*, pages 81–88.

Yue Lu, Hongning Wang, ChengXiang Zhai, and Dan Roth. 2012. Unsupervised discovery of opposing opinion networks from forum discussions. In *Proceedings of CIKM*, pages 1642–1646.

Yue Lu and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *Proceedings of WWW*, pages 121–130.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of WWW*, pages 131–140.

Hosam Mahmoud. 2008. *Polya Urn Models*. Chapman & Hall/CRC Texts in Statistical Science.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of WWW*, pages 171–180.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. 2012. Entity-centric topic-oriented opinion summarization in twitter. In *Proceedings of KDD*, pages 379–387.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, *38*(11), 39–41.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of EMNLP*, pages 262–272.

Samaneh Moghaddam and Martin Ester. 2011. ILDA: interdependent LDA model for learning latent aspects and their ratings from online product reviews. In *Proceedings of SIGIR*, pages 665–674.

Saif Mohammad, Cody Dunne, and Bonnie J. Dorr. 2009. Generating High-Coverage Semantic Orientation Lexicons From Overtly Marked Words and a Thesaurus. In *Proceedings of EMNLP*, pages 599–608.

Arjun Mukherjee and Bing Liu. 2012. Aspect Extraction through Semi-Supervised Modeling. In *Proceedings of ACL*, pages 339–348.

David Newman, Youn Noh, Edmund Talley, Sarvnaz Karimi, and Timothy Baldwin. 2010. Evaluating topic models for digital libraries. In *Proceedings of JCDL*, pages 215–224.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2), 1–135.

James Petterson, Alex Smola, Tibério Caetano, Wray Buntine, and Shravan Narayanamurthy. 2010. Word Features for Latent Dirichlet Allocation. In *Proceedings of NIPS*, pages 1921–1929.

AM Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT*, pages 339–346.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion Word Expansion and Target Extraction through Double Propagation. *Computational Linguistics*, *37*(1), 9–27.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of EMNLP*, pages 248–256.

Christina Sauper and Regina Barzilay. 2013. Automatic Aggregation by Joint Modeling of Aspects and Values. *J. Artif. Intell. Res. (JAIR)*, *46*, 89–127.

Christina Sauper, Aria Haghighi, and Regina Barzilay. 2011. Content Models with Attitude. In *Proceedings of ACL*, pages 350–358.

Swapna Somasundaran and J. Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL*, pages 226–234.

Keith Stevens and PKDAD Buttler. 2012. Exploring Topic Coherence over many models and many topics. In *Proceedings of EMNLP-CoNLL*, pages 952–961.

Veselin Stoyanov and Claire Cardie. 2011. Automatically Creating General-Purpose Opinion

Summaries from Text. In *Proceedings of RANLP*, pages 202–209.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 1–30.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*, pages 111–120.

Dong Wang and Yang Liu. 2011. A Pilot Study of Opinion Summarization in Conversations. In *Proceedings of ACL*, pages 331–339.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD*, pages 783–792.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of KDD*, pages 618–626.

Janyce Wiebe and Ellen Riloff. 2005. Creating Subjective and Objective Sentence Classifiers from Unannotated Texts. In *Proceedings of CICLing*, pages 486–497.

Janyce Wiebe, Theresa Wilson, Rebecca F. Bruce, Matthew Bell, and Melanie Martin. 2004. Learning Subjective Language. *Computational Linguistics*, *30*(3), 277–308.

Michael Wiegand and Dietrich Klakow. 2010. Convolution Kernels for Opinion Holder Extraction. In *Proceedings of HLT-NAACL*, pages 795–803.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of EMNLP*, pages 1533–1541.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2011. Structural Opinion Mining for Graph-based Sentiment Representation. In *Proceedings of EMNLP*, pages 1332–1341.

Yunqing Xia, Boyi Hao, and Kam-Fai Wong. 2009. Opinion Target Network and Bootstrapping Method for Chinese Opinion Target Extraction. In *Proceedings of AIRS*, pages 339–350.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. 2011. Aspect Ranking: Identifying Important Product Aspects from Online Consumer Reviews. In *Proceedings of ACL*, pages 1496–1505.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011. Constrained LDA for grouping product features in opinion mining. In *Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 448–459.

Lei Zhang and Bing Liu. 2011. Identifying Noun Product Features that Imply Opinions. In *Proceedings of ACL (Short Papers)*, pages 575–580.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly Modeling Aspects and Opinions with a MaxEnt-LDA Hybrid. In *Proceedings of EMNLP*, pages 56–65.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2012. Cross-Language Opinion Target Extraction in Review Texts. In *Proceedings of ICDM*, pages 1200–1205.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of CIKM*, pages 43–50. ACM Press.

Dependency-Based Decipherment for Resource-Limited Machine Translation

Qing Dou and Kevin Knight
Information Sciences Institute
Department of Computer Science
University of Southern California
{qdou,knight}@isi.edu

Abstract

We introduce dependency relations into deciphering foreign languages and show that dependency relations help improve the state-of-the-art deciphering accuracy by over 500%. We learn a translation lexicon from large amounts of genuinely non parallel data with decipherment to improve a phrase-based machine translation system trained with limited parallel data. In experiments, we observe BLEU gains of 1.2 to 1.8 across three different test sets.

1 Introduction

State-of-the-art machine translation (MT) systems apply statistical techniques to learn translation rules from large amounts of parallel data. However, parallel data is limited for many language pairs and domains.

In general, it is easier to obtain non parallel data. The ability to build a machine translation system using monolingual data could alleviate problems caused by insufficient parallel data. Towards building a machine translation system without a parallel corpus, Klementiev et al. (2012) use non parallel data to estimate parameters for a large scale MT system. Other work tries to learn full MT systems using only non parallel data through decipherment (Ravi and Knight, 2011; Ravi, 2013). However, the performance of such systems is poor compared with those trained with parallel data.

Given that we often have some parallel data, it is more practical to improve a translation system trained on parallel corpora with non parallel data. Dou and Knight (2012) successfully apply decipherment to learn a domain specific translation lexicon from monolingual data to improve out-of-domain machine translation. Although their approach works well for Spanish/French, they do not show whether their approach works for other language pairs. Moreover, the non parallel data used in their experiments is created from a parallel corpus. Such highly comparable data is difficult to obtain in reality.

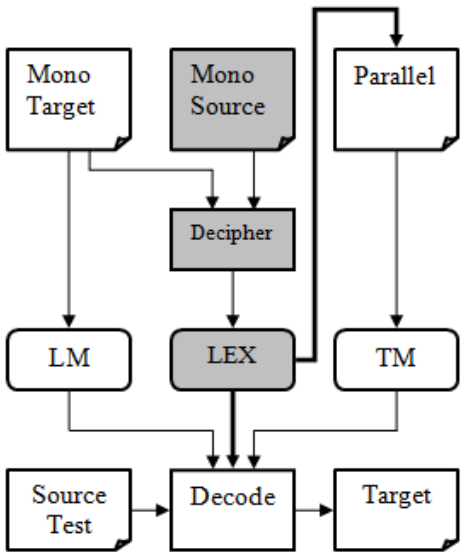


Figure 1: Improving machine translation with decipherment (Grey boxes represent new data and process). Mono: monolingual; LM: language model; LEX: translation lexicon; TM: translation model.

In this work, we improve previous work by Dou and Knight (2012) using genuinely non parallel data,

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1668–1676,
Seattle, Washington, USA, 18-21 October 2013.

and propose a framework to improve a machine translation system trained with a small amount of parallel data. As shown in Figure 1, we use a lexicon learned from decipherment to improve translations of both observed and out-of-vocabulary (OOV) words. The main contributions of this work are:

- We extract bigrams based on dependency relations for decipherment, which improves the state-of-the-art deciphering accuracy by over 500%.
- We demonstrate how to improve translations of words observed in parallel data by using a translation lexicon obtained from large amounts of non parallel data.
- We show that decipherment is able to find correct translations for OOV words.
- We use a translation lexicon learned by deciphering large amounts of non parallel data to improve a phrase-based MT system trained with limited amounts of parallel data. In experiments, we observe 1.2 to 1.8 BLEU gains across three different test sets.

2 Previous Work

Motivated by the idea that a translation lexicon induced from non parallel data can be applied to MT, a variety of prior research has tried to build a translation lexicon from non parallel or comparable data (Rapp, 1995; Fung and Yee, 1998; Koehn and Knight, 2002; Haghighi et al., 2008; Garera et al., 2009; Bergsma and Van Durme, 2011; Daumé and Jagarlamudi, 2011; Irvine and Callison-Burch, 2013b; Irvine and Callison-Burch, 2013a). Although previous work is able to build a translation lexicon without parallel data, little has used the lexicon to improve machine translation.

There has been increasing interest in learning translation lexicons from non parallel data with decipherment techniques (Ravi and Knight, 2011; Dou and Knight, 2012; Nuhn et al., 2012). Decipherment views one language as a cipher for another and learns a translation lexicon that produces a good decipherment.

In an effort to build a MT system without a parallel corpus, Ravi and Knight (2011) view Spanish as a cipher for English and apply Bayesian learning to directly decipher Spanish into English. Unfortunately, their approach can only work on small data with limited vocabulary. Dou and Knight (2012) propose two techniques to make Bayesian decipherment scalable.

First, unlike Ravi and Knight (2011), who decipher whole sentences, Dou and Knight (2012) decipher bigrams. Reducing a ciphertext to a set of bigrams with counts significantly reduces the amount of cipher data. According to Dou and Knight (2012), a ciphertext bigram F is generated through the following generative story:

- Generate a sequence of two plaintext tokens e_1e_2 with probability $P(e_1e_2)$ given by a language model built from large numbers of plaintext bigrams.
- Substitute e_1 with f_1 and e_2 with f_2 with probability $P(f_1|e_1) \cdot P(f_2|e_2)$.

The probability of any cipher bigram F is:

$$P(F) = \sum_{e_1e_2} P(e_1e_2) \prod_{i=1}^{2} P(f_i|e_i)$$

Given a corpus of N cipher bigrams $F_1...F_N$, the probability of the corpus is:

$$P(corpus) = \prod_{j=1}^{N} P(F_j)$$

Given a plaintext bigram language model, the goal is to manipulate $P(f|e)$ to maximize $P(corpus)$. Theoretically, one can directly apply EM to solve the problem (Knight et al., 2006). However, EM has time complexity $O(N \cdot V_e^2)$ and space complexity $O(V_f \cdot V_e)$, where V_f, V_e are the sizes of ciphertext and plaintext vocabularies respectively, and N is the number of cipher bigrams.

Ravi and Knight (2011) apply Bayesian learning to reduce the space complexity. Instead of estimating probabilities $P(f|e)$, Bayesian learning tries to draw samples from plaintext sequences given ciphertext bigrams. During sampling, the probability of any possible plaintext sample e_1e_2 is given as:

$$P_{sample}(e_1e_2) = P(e_1e_2) \prod_{i=1}^{2} P_{bayes}(f_i|e_i)$$

misión de naciones unidas en oriente medio	
misión de	misión naciones
de naciones	naciones unidas
naciones unidas	misión en
unidas en	en oriente
en oriente	oriente medio
oriente medio	

Table 1: Comparison of adjacent bigrams (left) and dependency bigrams (right) extracted from the same Spanish text

with $P_{bayes}(f_i|e_i)$ defined as:

$$P_{bayes}(f_i|e_i) = \frac{\alpha P_0(f_i|e_i) + count(f_i, e_i)}{\alpha + count(e_i)}$$

where P_0 is a base distribution, and α is a parameter that controls how much we trust P_0. $count(f_i, e_i)$ and $count(e_i)$ record the number of times f_i, e_i and e_i appear in previously generated samples respectively.

At the end of sampling, $P(f_i|e_i)$ is estimated by:

$$P(f_i|e_i) = \frac{count(f_i, e_i)}{count(e_i)}$$

However, Bayesian decipherment is still very slow with Gibbs sampling (Geman and Geman, 1987), as each sampling step requires considering V_e possibilities. Dou and Knight (2012) solve the problem by introducing slice sampling (Neal, 2000) to Bayesian decipherment.

3 From Adjacent Bigrams to Dependency Bigrams

A major limitation of work by Dou and Knight (2012) is their monotonic generative story for deciphering adjacent bigrams. While the generation process works well for deciphering similar languages (e.g. Spanish and French) without considering reordering, it does not work well for languages that are more different in grammar and word order (e.g. Spanish and English). In this section, we first look at why adjacent bigrams are bad for decipherment. Then we describe how to use syntax to solve the problem.

The left column in Table 1 contains adjacent bigrams extracted from the Spanish phrase "misión de naciones unidas en oriente medio". The correct decipherment for the bigram "naciones unidas" should be "united nations". Since the deciphering model described by Dou and Knight (2012) does not consider word reordering, it needs to decipher the bigram into "nations united" in order to get the right word translations "naciones"→"nations" and "unidas"→"united". However, the English language model used for decipherment is built from English adjacent bigrams, so it strongly disprefers "nations united" and is not likely to produce a sensible decipherment for "naciones unidas". The Spanish bigram "oriente medio" poses the same problem. Thus, without considering word reordering, the model described by Dou and Knight (2012) is not a good fit for deciphering Spanish into English.

However, if we extract bigrams based on dependency relations for both languages, the model fits better. To extract such bigrams, we first use dependency parsers to parse both languages, and extract bigrams by putting head word first, followed by the modifier.[1] We call these dependency bigrams. The right column in Table 1 lists examples of Spanish dependency bigrams extracted from the same Spanish phrase. With a language model built with English dependency bigrams, the same model used for deciphering adjacent bigrams is able to decipher Spanish dependency bigram "naciones(head) unidas(modifier)" into "nations(head) united(modifier)".

We might instead propose to consider word reordering when deciphering adjacent bigrams (e.g. add an operation to swap tokens in a bigram). However, using dependency bigrams has the following advantages:

- First, using dependency bigrams avoids complicating the model, keeping deciphering efficient and scalable.
- Second, it addresses the problem of long distance reordering, which can not be modeled by swapping tokens in bigrams.

Furthermore, using dependency bigrams allows us to use dependency types to further

[1] As use of "del" and "de" in Spanish is much more frequent than the use of "of" in English, we skip those words by using their head words as new heads if any of them serves as a head.

improve decipherment. Suppose we have a Spanish dependency bigram "aceptó(verb) solicitud(object)". Then all of the following English dependency bigrams are possible decipherments: "accepted(verb) UN(subject)", "accepted(verb) government(subject)", "accepted(verb) request(object)". However, if we know the type of the Spanish dependency bigram and use a language model built with the same type in English, the only possible decipherment is "accepted(verb) request(object)". If we limit the search space, a system is more likely to find a better decipherment.

4 Deciphering Spanish Gigaword

In this section, we compare dependency bigrams with adjacent bigrams for deciphering Spanish into English.

4.1 Data

We use the Gigaword corpus for our decipherment experiments. The corpus contains news articles from different news agencies and is available in Spanish and English. We use only the AFP (Agence France-Presse) section of the corpus in decipherment experiments. We tokenize the corpus using tools that come with the Europarl corpus (Koehn, 2005). To shorten the time required for running different systems on large amounts of data, we keep only the top 5000 most frequent word types in both languages and replace all other word types with UNK. We also throw away lines with more than 40 tokens, as the Spanish parser (Bohnet, 2010) we use is slow when processing long sentences. After preprocessing, the corpus contains approximately 440 million tokens in Spanish and 350 million tokens in English. To obtain dependency bigrams, we use the Bohnet parsers (Bohnet, 2010) to parse both the Spanish and English version of the corpus.

4.2 Systems

Three systems are evaluated in the experiments. We implement a baseline system, **Adjacent**, based on Dou and Knight (2012). The baseline system collects adjacent bigrams and their counts from Spanish and English texts. It then builds an English bigram language model using the English adjacent bigrams and uses it to decipher the Spanish adjacent bigrams.

	Dependency Types
Group 1	Verb/Subject
Group 2	Preposition/Preposition-Object, Noun/Noun-Modifier
Group 3	Verb/Noun-Object

Table 2: Dependency relations divided into three groups

We build the second system, **Dependency**, using dependency bigrams for decipherment. As the two parsers do not output the same set of dependency relations, we cannot extract all types of dependency bigrams. Instead, we select a subset of dependency bigrams whose dependency relations are shared by the two parser outputs. The selected dependency relations are: Verb/Subject, Verb/Noun-Object, Preposition/Object, Noun/Modifier. Decipherment runs the same way as in the baseline system.

The third system, **DepType**, is built using both dependent bigrams and their dependency types. We first extract dependency bigrams for both languages, then group them based on their dependency types. As both parsers treat noun phrases dependent on "del", "de", and "of" as prepositional phrases, we choose to divide the dependency bigrams into 3 groups and list them in Table 2. A separate language model is built for each group of English dependency bigrams and used to decipher the group of Spanish dependency bigrams with same dependency type.

For all the systems, language models are built using the SRILM toolkit (Stolcke, 2002). For the Adjacent system, we use Good-Turing smoothing. For the other systems, we use a mix of Witten-Bell and Good-Turing smoothing.

4.3 Sampling Procedure

In experiments, we find that the iterative sampling method described by Dou and Knight (2012) helps improve deciphering accuracy. We also find that combining results from different decipherments helps find more correct translations at each iteration. Thus, instead of using a single sampling process, we use 10 different sampling processes at each iteration. The details of the new sampling procedure are provided here:

- Extract dependency bigrams from parsing outputs and collect their counts.

- Keep bigrams whose counts are greater than a threshold α. Then start 10 different randomly seeded and initialized sampling processes. Perform sampling.
- At the end of sampling, extract word translation pairs (f, e) from the final sample. Estimate translation probabilities $P(e|f)$ for each pair. Then construct a translation table by keeping translation pairs (f, e) seen in more than one decipherment and use the average $P(e|f)$ as the new translation probability.
- Lower the threshold α to include more bigrams into the sampling process. Start 10 different sampling processes again and initialize the first sample using the translation pairs obtained from the previous step (for each Spanish token f, choose an English token e whose $P(e|f)$ is the highest). Perform sampling again.
- Repeat until $\alpha = 1$.

4.4 Deciphering Accuracy

We choose the first 1000 lines of the monolingual Spanish texts as our test data. The data contains 37,505 tokens and 6556 word types. We use type accuracy as our evaluation metric: Given a word type f in Spanish, we find a translation pair (f, e) with the highest average $P(e|f)$ from the translation table learned through decipherment. If the translation pair (f, e) can also be found in a gold translation lexicon T_{gold}, we treat the word type f as correctly deciphered. Let $|C|$ be the number of word types correctly deciphered, and $|V|$ be the total number of word types evaluated. We define type accuracy as $\frac{|C|}{|V|}$.

To create T_{gold}, we use GIZA (Och and Ney, 2003) to align a small amount of Spanish-English parallel text (1 million tokens for each language), and use the lexicon derived from the alignment as our gold translation lexicon. T_{gold} contains a subset of 4408 types seen in the test data, among which, 2878 are also top 5000 frequent word types.

4.5 Results

During decipherment, we gradually increase the size of Spanish texts and compare the learning curves of three deciphering systems in Figure 2.

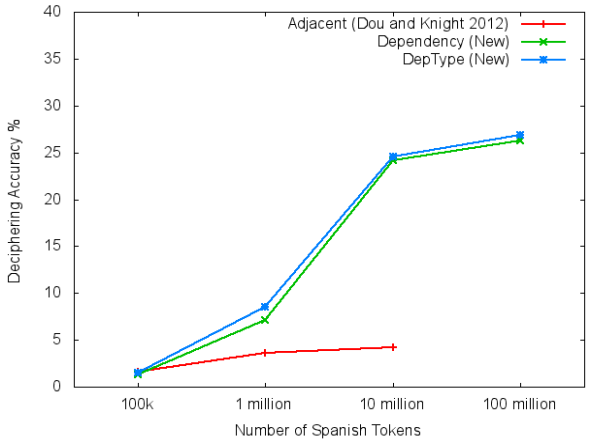


Figure 2: Learning curves for three decipherment systems. Compared with Adjacent (previous state of the art), systems that use dependency bigrams improve deciphering accuracy by over 500%.

With 100k tokens of Spanish text, the performance of the three systems are similar. However, the learning curve of Adjacent plateaus quickly, while those of the dependency based systems soar up as more data becomes available and still rise sharply when the size of Spanish texts increases to 10 million tokens, where the DepType system improves deciphering accuracy of the Adjacent system from 4.2% to 24.6%. In the end, with 100 million tokens, the accuracy of the DepType system rises to 27.0%. The accuracy is even higher (41%), when evaluated against the top 5000 frequent word types only.

5 Improving Machine Translation with Decipherment

In this section, we demonstrate how to use a translation lexicon learned by deciphering large amounts of in-domain (news) monolingual data to improve a phrase-based machine translation system trained with limited out-of-domain (politics) parallel data.

5.1 Data

We use approximately one million tokens of the Europarl corpus (Koehn, 2005) as our small out-of-domain parallel training data and Gigaword as our large in-domain monolingual training data to build language models and a new translation lexicon to improve a phrase-based MT baseline system. For tuning and testing, we use the development data

Parallel		
	Spanish	English
Europarl	1.1 million	1.0 million
Tune-2008	52.6k	49.8k
Test-2009	68.1k	65.6k
Test-2010	65.5k	61.9k
Test-2011	79.4k	74.7k
Non Parallel		
	Spanish	English
Gigaword	894 million	940 million

Table 3: Size of training, tuning, and testing data in number of tokens

from the NAACL 2012 workshop on statistical machine translation. The data contains test data in the news domain from the 2008, 2009, 2010, and 2011 workshops. We use the 2008 test data for tuning and the rest for testing. The sizes of the training, tuning, and testing sets are listed in Table 3.

5.2 Systems

5.2.1 Baseline Machine Translation System

We build a state-of-the-art phrase-based MT system, PBMT, using Moses (Koehn et al., 2007). PBMT has 3 models: a translation model, a distortion model, and a language model. We build a 5-gram language model using the AFP section of the English Gigaword. We train the other models using the Europarl corpus. By default, Moses uses the following 8 features to score a candidate translation:

- direct and inverse translation probabilities
- direct and inverse lexical weighting
- a language model score
- a distortion score
- phrase penalty
- word penalty

The 8 features have weights adjusted on the tuning data using minimum error rate training (MERT) (Och, 2003). PBMT has a phrase table T_{phrase}. During decoding, Moses copies out-of-vocabulary (OOV) words, which can not be found in T_{phrase}, directly to output. In the following sections, we describe how to use a translation lexicon learned from large amounts of non parallel data to improve translation of OOV words, as well as words observed in T_{phrase}.

5.2.2 Decipherment for Machine Translation

To achieve better decipherment, we:

- Increase the size of Spanish ciphertext from 100 million tokens to 894 million tokens.
- Keep top 50k instead of top 5k most frequent word types of the ciphertext.
- Instead of seeding the sampling process randomly, we use a translation lexicon learned from a limited amount of parallel data as seed: For each Spanish dependency bigram f_1, f_2, where both f_1 and f_2 are found in the seed lexicon, we find the English sequence e_1, e_2 that maximizes $P(e_1, e_2)P(e_1|f_1)P(e_2|f_2)$. Otherwise, for any Spanish token f that can be found in the seed lexicon, we choose English word e, where $P(e|f)$ is the highest as the initial sample; for any f that are not seen in the seed lexicon, we do random initialization.

We perform 20 random restarts with 10k iterations on each and build a word-to-word translation lexicon $T_{decipher}$ by collecting translation pairs seen in at least 3 final decipherments with either $P(f|e) \geq 0.2$ or $P(e|f) \geq 0.2$.

5.2.3 Improving Translation of Observed Words with Decipherment

To improve translation of words observed in our parallel corpus, we simply use $T_{decipher}$ as an additional parallel corpus. First, we filter $T_{decipher}$ by keeping only translation pairs (f, e), where f is observed in the Spanish part and e is observed in the English part of the parallel corpus. Then we append all the Spanish and English words in the filtered $T_{decipher}$ to the end of Spanish part and English part of the parallel corpus respectively. The training and tuning process is the same as the baseline machine translation system PBMT. We denote this system as **Decipher-OBSV**.

5.2.4 Improving OOV translation with Decipherment

As $T_{decipher}$ is learned from large amounts of in-domain monolingual data, we expect that $T_{decipher}$ contains a number of useful translations for words not seen in the limited amount of parallel data (OOV words). Instead of copying OOV words directly to output, which is what Moses does by default, we try to find translations from $T_{decipher}$ to improve translation.

During decoding, if a source word f is in T_{phrase}, its translation options are collected from T_{phrase} exclusively. If f is not in T_{phrase} but in $T_{decipher}$, the decoder will find translations from $T_{decipher}$. If f is not in either translation table, the decoder just copies it directly to the output. We call this system **Decipher-OOV**.

However, when an OOV's correct translation is same as its surface form and all its possible translations in $T_{decipher}$ are wrong, it is better to just copy OOV words directly to output. This scenario happens frequently, as Spanish and English share many common words. To avoid over trusting $T_{decipher}$, we add a new translation pair (f, f) for each source word f in $T_{decipher}$ if the translation pair (f, f) is not originally in $T_{decipher}$. For each newly added translation pair, both of its log translation probabilities are set to 0. To distinguish the added translation pairs from the others learned through decipherment, we add a binary feature θ to each translation pair in $T_{decipher}$. The final version of $T_{decipher}$ has three feature scores: $P(e|f)$, $P(f|e)$, and θ. Finally, we tune weights of the features in $T_{decipher}$ using MERT (Och, 2003) on the tuning set.

5.2.5 A Combined Approach

In the end, we build a system **Decipher-COMB**, which uses $T_{decipher}$ to improve translation of both observed and OOV words with methods described in sections 5.2.3 and 5.2.4.

5.3 Results

We tune each system three times with MERT and choose the best weights based on BLEU scores on tuning set.

Table 4 shows that the translation lexicon learned from decipherment helps achieve higher BLEU scores across tuning and testing sets. **Decipher-OBSV** improves BLEU scores by as much as 1.2 points. We analyze the results and find the gain mainly comes from two parts. First, adding $T_{decipher}$ to small amounts of parallel corpus improves word level translation probabilities, which lead to better lexical weighting; second, $T_{decipher}$ contains new alternative translations for words observed in the parallel corpus.

Moreover, **Decipher-OOV** also achieves better BLEU scores compared with PBMT across all tuning and test sets. We also observe that systems using $T_{decipher}$ learned by deciphering dependency bigrams leads to larger gains in BLEU scores. When decipherment is used to improve translation of both observed and OOV words, we see improvement in BLEU score as high as 1.8 points on the 2010 news test set.

The consistent improvement on the tuning and different testing data suggests that decipherment is capable of learning good translations for a number of OOV words. To further demonstrate that our decipherment approach finds useful translations for OOV words, we list the top 10 most frequent OOV words from both the tuning set and testing set as well as their translations (up to three most likely translations) in Table 5. $P(e|f)$ and $P(f|e)$ are average scores over different decipherment runs.

From the table, we can see that decipherment finds correct translations (bolded) for 7 out of the 10 most frequent OOV words. Moreover, many OOVs and their correct translations are homographs , which makes copying OOVs directly to the output a strong baseline to beat. Nonetheless, decipherment still finds enough correct translations to improve the baseline.

6 Conclusion

We introduce syntax for deciphering Spanish into English. Experiment results show that using dependency bigrams improves decipherment accuracy by over 500% compared with the state-of-the-art approach. Moreover, we learn a domain specific translation lexicon by deciphering large amounts of monolingual data and show that the lexicon can improve a baseline machine translation system trained with limited parallel data.

Decipherment	System	$Tune_{2008}$	$Test_{2009}$	$Test_{2010}$	$Test_{2011}$
None	PBMT (Baseline)	19.1	19.6	21.3	22.1
Adjacent	Decipher-OBSV	19.5	20.1	22.2	22.6
	Decipher-OOV	19.4	19.9	21.7	22.5
	Decipher-COMB	19.5	20.2	22.3	22.5
Dependency	Decipher-OBSV	19.7	20.5	22.5	23.0
	Decipher-OOV	19.9	20.4	22.4	22.9
	Decipher-COMB	20.0	20.8	23.1	23.4

Table 4: Systems that use translation lexicons learned from decipherment show consistent improvement over the baseline system across tuning and testing sets. The best system, Decipher-COMB, achieves as much as 1.8 BLEU point gain on the 2010 news test set.

Spanish	English	$P(e\|f)$	$P(f\|e)$
obama	his	0.33	0.01
	bush	0.27	0.07
	clinton	0.23	0.11
bush	**bush**	0.47	0.45
	yeltsin	0.28	0.81
	he	0.24	0.05
festival	event	0.68	0.35
	festival	0.61	0.72
wikileaks	zeta	0.03	0.33
venus	**venus**	0.61	0.74
	serena	0.47	0.62
colchones	**mattresses**	0.55	0.73
	cars	0.31	0.01
helado	frigid	0.52	0.44
	chill	0.37	0.14
	sandwich	0.42	0.27
google	microsoft	0.67	0.18
	google	0.59	0.69
cantante	**singer**	0.44	0.92
	jackson	0.14	0.33
	artists	0.14	0.77
mccain	**mccain**	0.66	0.92
	it	0.22	0.00
	he	0.21	0.00

Table 5: Decipherment finds correct translations for 7 out of 10 most frequent OOV word types.

7 Acknowledgments

This work was supported by NSF Grant 0904684 and ARO grant W911NF-10-1-0533. The authors would like to thank David Chiang, Malte Nuhn, Victoria Fossum, Ashish Vaswani, Ulf Hermjakob, Yang Gao, and Hui Zhang (in no particular order) for their comments and suggestions.

References

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*. AAAI Press.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Coling.

Hal Daumé, III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational*

Linguistics - Volume 1. Association for Computational Linguistics.

Nikesh Garera, Chris Callison-Burch, and David Yarowsky. 2009. Improving translation lexicon induction from monolingual corpora via dependency contexts and part-of-speech equivalences. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1987. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in computer vision: issues, problems, principles, and paradigms*. Morgan Kaufmann Publishers Inc.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2013a. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, August.

Ann Irvine and Chris Callison-Burch. 2013b. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without parallel corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition*. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: a parallel corpus for statistical machine translation. In *In Proceedings of the Tenth Machine Translation Summit*, Phuket, Thailand. Asia-Pacific Association for Machine Translation.

Radford Neal. 2000. Slice sampling. *Annals of Statistics*, 31.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*.

Translating into Morphologically Rich Languages with Synthetic Phrases

Victor Chahuneau Eva Schlinger Noah A. Smith Chris Dyer
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{vchahune,eschling,nasmith,cdyer}@cs.cmu.edu

Abstract

Translation into morphologically rich languages is an important but recalcitrant problem in MT. We present a simple and effective approach that deals with the problem in two phases. First, a discriminative model is learned to predict inflections of target words from rich source-side annotations. Then, this model is used to create additional sentence-specific word- and phrase-level translations that are added to a standard translation model as "synthetic" phrases. Our approach relies on morphological analysis of the target language, but we show that an unsupervised Bayesian model of morphology can successfully be used in place of a supervised analyzer. We report significant improvements in translation quality when translating from English to Russian, Hebrew and Swahili.

1 Introduction

Machine translation into morphologically rich languages is challenging, due to lexical sparsity and the large variety of grammatical features expressed with morphology. In this paper, we introduce a method that uses target language morphological grammars (either hand-crafted or learned unsupervisedly) to address this challenge and demonstrate its effectiveness at improving translation from English into several morphologically rich target languages.

Our approach decomposes the process of producing a translation for a word (or phrase) into two steps. First, a meaning-bearing **stem** is chosen and then an appropriate **inflection** is selected using a feature-rich discriminative model that conditions on the source context of the word being translated.

Rather than attempting to directly produce full-sentence translations using such an elementary process, we use our model to generate translations of individual words and short phrases that *augment*—on a sentence-by-sentence basis—the inventory of translation rules obtained using standard translation rule extraction techniques (Chiang, 2007). We call these **synthetic phrases**.

The major advantages of our approach are: (i) synthesized forms are targeted to a specific translation context; (ii) multiple, alternative phrases may be generated with the final choice among rules left to the global translation model; (iii) virtually no language-specific engineering is necessary; (iv) any phrase- or syntax-based decoder can be used without modification; and (v) we can generate forms that were not attested in the bilingual training data.

The paper is structured as follows. We first present our "translate-and-inflect" model for predicting lexical translations into morphologically rich languages given a source word and its context (§2). Our approach requires a morphological grammar to relate surface forms to underlying $\langle \text{stem}, \text{inflection} \rangle$ pairs; we discuss how either a standard morphological analyzer or a simple Bayesian unsupervised analyzer can be used (§3). After describing an efficient parameter estimation procedure for the inflection model (§4), we employ the translate-and-inflect model in an MT system. We describe how we use our model to synthesize translation options (§5) and then evaluate translation quality on English–Russian, English–Hebrew, and English–

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1677–1687,
Seattle, Washington, USA, 18-21 October 2013.

Swahili translation tasks, finding significant improvements in all language pairs (§6). We finally review related work (§7) and conclude (§8).

2 Translate-and-Inflect Model

The task of the translate-and-inflect model is illustrated in Fig. 1 for an English–Russian sentence pair. The input will be a sentence $\boldsymbol{e}$ in the source language (in this paper, always English) and any available linguistic analysis of $\boldsymbol{e}$. The output $\boldsymbol{f}$ will be composed of (i) a sequence of stems, each denoted σ and (ii) one morphological inflection pattern for each stem, denoted μ. When the information is available, a stem σ is composed of a lemma and an inflectional class. Throughout, we use Ω_σ to denote the set of possible morphological inflection patterns for a given stem σ. Ω_σ might be defined by a grammar; our models restrict Ω_σ to be the set of inflections observed anywhere in our monolingual or bilingual training data as a realization of σ.[1]

We assume the availability of a deterministic function that maps a stem σ and morphological inflection μ to a target language surface form f. In some cases, such as our unsupervised approach in §3.2, this will be a concatenation operation, though finite-state transducers are traditionally used to define such relations (§3.1). We abstractly denote this operation by $\star$: $f = \sigma \star \mu$.

Our approach consists in defining a probabilistic model over target words $\boldsymbol{f}$. The model assumes independence between each target word f conditioned on the source sentence $\boldsymbol{e}$ and its aligned position i in this sentence.[2] This assumption is further relaxed in §5 when the model is integrated in the translation system.

We decompose the probability of generating each target word f in the following way:

$$p(f \mid \boldsymbol{e}, i) = \sum_{\sigma \star \mu = f} \underbrace{p(\sigma \mid e_i)}_{\text{gen. stem}} \times \underbrace{p(\mu \mid \sigma, \boldsymbol{e}, i)}_{\text{gen. inflection}}$$

Here, each stem is generated independently from a single aligned source word e_i, but in practice we use a standard phrase-based model to generate sequences of stems and only the inflection model operates word-by-word. We turn next to the inflection model.

2.1 Modeling Inflection

In morphologically rich languages, each stem may be combined with one or more inflectional morphemes to express many different grammatical features (e.g., case, definiteness, mood, tense, etc.).

Since the inflectional morphology of a word generally expresses multiple grammatical features, we would like a model that naturally incorporates rich, possibly overlapping features in its representation of both the input (i.e., conditioning context) and output (i.e., the inflection pattern). We therefore use the following parametric form to model inflectional probabilities:

$$u(\mu, \boldsymbol{e}, i) = \exp\Big[\boldsymbol{\varphi}(\boldsymbol{e}, i)^\top \mathbf{W} \boldsymbol{\psi}(\mu) + \boldsymbol{\psi}(\mu)^\top \mathbf{V} \boldsymbol{\psi}(\mu)\Big],$$

$$p(\mu \mid \sigma, \boldsymbol{e}, i) = \frac{u(\mu, \boldsymbol{e}, i)}{\sum_{\mu' \in \Omega_\sigma} u(\mu', \boldsymbol{e}, i)}. \quad (1)$$

Here, $\boldsymbol{\varphi}$ is an m-dimensional *source context* feature vector function, $\boldsymbol{\psi}$ is an n-dimensional *morphology* feature vector function, $\mathbf{W} \in \mathbb{R}^{m \times n}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are parameter matrices. As with the more familiar log-linear parametrization that is written with a single feature vector, single weight vector and single bias vector, this model is linear in its parameters (it can be understood as working with a feature space that is the outer product of the two feature spaces). However, using two feature vectors allows to define overlapping features of both the input *and* the output, which is important for modeling morphology in which output variables are naturally expressed as bundles of features. The second term in the sum in u enables correlations among output features to be modeled independently of input, and as such can be understood as a generalization of the bias terms in multi-class logistic regression (on the diagonal $\mathbf{V}_{ii}$) and interaction terms between output variables in a conditional random field (off the diagonal $\mathbf{V}_{ij}$).

[1] This prevents the model from generating words that would be difficult for the language model to reliably score.

[2] This is the same assumption that Brown et al. (1993) make in, for example, IBM Model 1.

Figure 1: The inflection model predicts a form for the target verb lemma σ =пытаться (*pytat'sya*) based on its source *attempted* and the linear and syntactic source context. The correct inflection string for the observed Russian form in this particular training instance is μ = mis-sfm-e (equivalent to the more traditional morphological string: +MAIN+IND+PAST+SING+FEM+MEDIAL+PERF).

$$\left\{\begin{array}{c}\text{source aligned word } e_i \\ \text{parent word } e_{\pi_i} \text{ with its dependency } \pi_i \rightarrow i \\ \text{all children } e_j \mid \pi_j = i \text{ with their dependency } i \rightarrow j \\ \text{source words } e_{i-1} \text{ and } e_{i+1}\end{array}\right\} \left\{\begin{array}{c}\text{token} \\ \text{part-of-speech tag} \\ \text{word cluster}\end{array}\right\}$$
$$\text{– are } e_i, e_{\pi_i} \text{ at the root of the dependency tree?}$$
$$\text{– number of children, siblings of } e_i$$

Figure 2: Source features $\boldsymbol{\varphi}(\boldsymbol{e}, i)$ extracted from $\boldsymbol{e}$ and its linguistic analysis. π_i denotes the parent of the token in position i in the dependency tree and $\pi_i \rightarrow i$ the typed dependency link.

2.2 Source Context Features: $\boldsymbol{\varphi}(\boldsymbol{e}, i)$

In order to select the best inflection of a target-language word, given the source word it translates and the context of that source word, we seek to exploit as many features of the context as are available. Consider the example shown in Fig. 1, where most of the inflection features of the Russian word (past tense, singular number, and feminine gender) can be inferred from the context of the English word it is aligned to. Indeed, many grammatical functions expressed morphologically in Russian are expressed syntactically in English. Fortunately, high-quality parsers and other linguistic analyzers are available for English.

On the source side, we apply the following processing steps:

- Part-of-speech tagging with a CRF tagger trained on sections 02–21 of the Penn Treebank.
- Dependency parsing with TurboParser (Martins et al., 2010), a non-projective dependency parser trained on the Penn Treebank to produce basic Stanford dependencies.
- Assignment of tokens to one of 600 Brown clusters, trained on 8G words of English text.[3]

We then extract binary features from $\boldsymbol{e}$ using this information, by considering the aligned source word e_i, its preceding and following words, and its syntactic neighbors. These are detailed in Figure 2.

3 Morphological Grammars and Features

We now describe how to obtain morphological analyses and convert them into feature vectors ($\boldsymbol{\psi}$) for our target languages, Russian, Hebrew, and Swahili, using supervised and unsupervised methods.

3.1 Supervised Morphology

The state-of-the-art in morphological analysis uses unweighted morphological transduction rules (usu-

[3]The entire monolingual data available for the translation task of the 8th ACL Workshop on Statistical Machine Translation was used.

ally in the form of an FST) to produce candidate analyses for each word in a sentence and then statistical models to disambiguate among the analyses in context (Hakkani-Tür et al., 2000; Hajič et al., 2001; Smith et al., 2005; Habash and Rambow, 2005, *inter alia*). While this technique is capable of producing high quality linguistic analyses, it is expensive to develop, requiring hand-crafted rule-based analyzers and annotated corpora to train the disambiguation models. As a result, such analyzers are only available for a small number of languages, and, as a practical matter, each analyzer (which resulted from different development efforts) operates differently from the others.

We therefore focus on using supervised analysis for a single target language, Russian. We use the analysis tool of Sharoff et al. (2008) which produces for each word in context a lemma and a fixed-length morphological tag encoding the grammatical features. We process the target side of the parallel data with this tool to obtain the information necessary to extract $\langle \text{lemma}, \text{inflection} \rangle$ pairs, from which we compute σ and morphological feature vectors $\boldsymbol{\psi}(\mu)$.

Supervised morphology features: $\boldsymbol{\psi}(\mu)$. Since a positional tag set is used, it is straightforward to convert each fixed-length tag μ into a feature vector by defining a binary feature for each key-value pair (e.g., Tense=past) composing the tag.

3.2 Unsupervised Morphology

Since many languages into which we might want to translate do not have supervised morphological analyzers, we now turn to the question of how to generate morphological analyses and features using an unsupervised analyzer. We hypothesize that perfect decomposition into rich linguistic structures may not be required for accurate generation of new inflected forms. We will test this hypothesis by experimenting with a simple, *unsupervised* model of morphology that segments words into sequences of morphemes, assuming a (naïve) concatenative generation process and a single analysis per type.

Unsupervised morphological segmentation. We assume that each word can be decomposed into any number of prefixes, a stem, and any number of suffixes. Formally, we let M represent the set of all possible morphemes and define a regular grammar M^*MM^* (i.e., zero or more prefixes, a stem, and zero or more suffixes). To infer the decomposition structure for the words in the target language, we assume that the vocabulary was generated by the following process:

1. Sample morpheme distributions from symmetric Dirichlet distributions: $\theta_p \sim \text{Dir}_{|M|}(\alpha_p)$ for prefixes, $\theta_\sigma \sim \text{Dir}_{|M|}(\alpha_\sigma)$ for stems, and $\theta_s \sim \text{Dir}_{|M|}(\alpha_s)$ for suffixes.

2. Sample length distribution parameters $\lambda_p \sim \text{Beta}(\beta_p, \gamma_p)$ for prefix sequences and $\lambda_s \sim \text{Beta}(\beta_s, \gamma_s)$ for suffix sequences.

3. Sample a vocabulary by creating each word type w using the following steps:
   (a) Sample affix sequence lengths: $l_p \sim \text{Geometric}(\lambda_p)$; $l_s \sim \text{Geometric}(\lambda_s)$.
   (b) Sample l_p prefixes $p_1, \ldots, p_{l_p}$ independently from θ_p; l_s suffixes $s_1, \ldots, s_{l_s}$ independently from θ_s; and a stem $\sigma \sim \theta_\sigma$.
   (c) Concatenate prefixes, the stem, and suffixes: $w = p_1 + \cdots + p_{l_p} + \sigma + s_1 + \cdots + s_{l_s}$.

We use blocked Gibbs sampling to sample segmentations for each word in the training vocabulary. Because of our particular choice of priors, it possible to approximately decompose the posterior over the arcs of a compact finite-state machine. Sampling a segmentation or obtaining the most likely segmentation *a posteriori* then reduces to familiar FST operations. This model is reminiscent of work on learning morphology using adaptor grammars (Johnson et al., 2006; Johnson, 2008).

The inferred morphological grammar is very sensitive to the Dirichlet hyperparameters $(\alpha_p, \alpha_s, \alpha_\sigma)$ and these are, in turn, sensitive to the number of types in the vocabulary. Using $\alpha_p, \alpha_s \ll \alpha_\sigma \ll 1$ tended to recover useful segmentations, but we have not yet been able to find reliable generic priors for these values. Therefore, we selected them empirically to obtain a stem vocabulary size on the parallel data that is one-to-one with English.[4] Future work

[4] Our default starting point was to use $\alpha_p = \alpha_s = 10^{-6}, \alpha_\sigma = 10^{-4}$ and then to adjust all parameters by factors of 10.

Table 1: Corpus statistics.

	Parallel					Parallel+Monolingual		
	Sentences	EN-tokens	TRG-tokens	EN-types	TRG-types	Sentences	TRG-tokens	TRG-types
Russian	150k	3.5M	3.3M	131k	254k	20M	360M	1,971k
Hebrew	134k	2.7M	2.0M	48k	120k	806k	15M	316k
Swahili	15k	0.3M	0.3M	23k	35k	596k	13M	334k

will involve a more direct method for specifying or inferring these values.

Unsupervised morphology features: $\psi(\mu)$. For the unsupervised analyzer, we do not have a mapping from morphemes to structured morphological attributes; however, we can create features from the affix sequences obtained after morphological segmentation. We produce binary features corresponding to the content of each potential affixation position relative to the stem:

For example, the unsupervised analysis μ = wa+ki+wa+STEM of the Swahili word *wakiwapiga* will produce the following features:

$$\psi_{\text{prefix}[-3][\text{wa}]}(\mu) = 1,$$
$$\psi_{\text{prefix}[-2][\text{ki}]}(\mu) = 1,$$
$$\psi_{\text{prefix}[-1][\text{wa}]}(\mu) = 1.$$

4 Inflection Model Parameter Estimation

To set the parameters $\mathbf{W}$ and $\mathbf{V}$ of the inflection prediction model (Eq. 1), we use stochastic gradient descent to maximize the conditional log-likelihood of a training set consisting of pairs of source (English) sentence contextual features ($\boldsymbol{\varphi}$) and target word inflectional features ($\boldsymbol{\psi}$). The training instances are extracted from the word-aligned parallel corpus with the English side preprocessed as discussed in §2.2 and the target side disambiguated as discussed in §3. When morphological category information is available, we train an independent model for each open-class category (in Russian, nouns, verbs, adjectives, numerals, adverbs); otherwise a single model is used for all words (excluding words less than four characters long, which are ignored).

Statistics of the parallel corpora used to train the inflection model are summarized in Table 1. It is important to note here that our richly parameterized model is trained on the full parallel training corpus, not just on a handful of development sentences (which are typically used to tune MT system parameters). Despite this scale, training is simple: the inflection model is trained to discriminate among different inflectional paradigms, not over all possible target language sentences (Blunsom et al., 2008) or learning from all observable rules (Subotin, 2011). This makes the training problem relatively tractable: all experiments in this paper were trained on a single processor using a Cython implementation of the SGD optimizer. For our largest model, trained on 3.3M Russian words, $n = 231K \times m = 336$ features were produced, and 10 SGD iterations were performed in less than 16 hours.

4.1 Intrinsic Evaluation

Before considering the broader problem of integrating the inflection model in a machine translation system, we perform an artificial evaluation to verify that the model learns sensible source sentence-target inflection patterns. To do so, we create an inflection test set as follows. We preprocess the source (English) sentences exactly as during training (§2.2), and using the target language morphological analyzer, we convert each aligned target word to $\langle$stem, inflection$\rangle$ pairs. We perform word alignment on the held-out MT development data for each language pair (cf. Table 1), exactly as if it were going to produce training instances, but instead we use them for testing.

Although the resulting dataset is noisy (e.g., due to alignment errors), this becomes our intrinsic evaluation test set. Using this data, we measure inflection quality using two measurements:[5]

[5]Note that we are not evaluating the stem translation model,

			acc.	ppl.	$\lvert\Omega_\sigma\rvert$
Supervised	Russian	N	64.1%	3.46	9.16
		V	63.7%	3.41	20.12
		A	51.5%	6.24	19.56
		M	73.0%	2.81	9.14
		average	63.1%	3.98	14.49
Unsup.	Russian	all	71.2%	2.15	4.73
	Hebrew	all	85.5%	1.49	2.55
	Swahili	all	78.2%	2.09	11.46

Table 2: Intrinsic evaluation of inflection model (N: nouns, V: verbs, A: adjectives, M: numerals).

- the accuracy of predicting the inflection given the source, source context and target stem, and
- the inflection model perplexity on the same set of test instances.

Additionally, we report the average number of possible inflections for each stem, an upper bound to the perplexity that indicates the inherent difficulty of the task. The results of this evaluation are presented in Table 2 for the three language pairs considered. We remark on two patterns in these results. First, perplexity is substantially lower than the perplexity of a uniform model, indicating our model is overall quite effective at predicting inflections using source context only. Second, in the supervised Russian results, we see that predicting the inflections of adjectives is relatively more difficult than for other parts-of-speech. Since adjectives agree with the nouns they modify in gender and case, and gender is an idiosyncratic feature of Russian nouns (and therefore not directly predictable from the English source), this difficulty is unsurprising.

We can also inspect the weights learned by the model to assess the effectiveness of the features in relating source-context structure with target-side morphology. Such an analysis is presented in Fig. 3.

4.2 Feature Ablation

Our inflection model makes use of numerous feature types. Table 3 explores the effect of removing different kinds of (source) features from the model, evaluated on predicting Russian inflections using supervised morphological grammars.[6] Rows 2–3 show the effect of removing either linear or dependency context. We see that both are necessary for good performance; however removing dependency context substantially degrades performance of the model (we interpret this result as evidence that Russian morphological inflection captures grammatical relationships that would be expressed structurally in English). The bottom four rows explore the effect of source language word representation. The results indicate that lexical features are important for accurate prediction of inflection, and that POS tags and Brown clusters are likewise important, but they seem to capture similar information (removing one has little impact, but removing both substantially degrades performance).

Table 3: Feature ablation experiments using supervised Russian classification experiments.

Features ($\boldsymbol{\varphi}(\boldsymbol{e}, i)$)	acc.
all	54.7%
−linear context	52.7%
−dependency context	44.4%
−POS tags	54.5%
−Brown clusters	54.5%
−POS tags, −Brown cl.	50.9%
−lexical items	51.2%

5 Synthetic Phrases

We turn now to translation; recall that our translate-and-inflect model is used to augment the set of rules available to a conventional statistical machine translation decoder. We refer to the phrases it produces as *synthetic* phrases.

Our baseline system is a standard hierarchical phrase-based translation model (Chiang, 2007). Following Lopez (2007), the training data is compiled into an efficient binary representation which allows extraction of sentence-specific grammars just before decoding. In our case, this also allows the creation of synthetic inflected phrases that are produced conditioning on the sentence to translate.

To generate these synthetic phrases with new inflections possibly unseen in the parallel training

just the inflection prediction model.

[6]The models used in the feature ablation experiment were trained on fewer examples, resulting in overall lower accuracies than seen in Table 2, but the pattern of results is the relevant datapoint here.

Russian supervised

Verb: 1st Person
`child(nsubj)=I child(nsubj)=we`
Verb: Future tense
`child(aux)=MD child(aux)=will`
Noun: Animate
`source=animals/victims/...`
Noun: Feminine gender
`source=obama/economy/...`
Noun: Dative case
`parent(iobj)`
Adjective: Genitive case
`grandparent(poss)`

Hebrew

Suffix ים (masculine plural)
`parent=NNS after=NNS`
Prefix א (first person sing. + future)
`child(nsubj)=I child(aux)='ll`
Prefix כ (preposition like/as)
`child(prep)=IN parent=as`
Suffix י (possesive mark)
`before=my child(poss)=my`
Suffix ה (feminine mark)
`child(nsubj)=she before=she`
Prefix כש (when)
`before=when before=WRB`

Swahili

Prefix *li* (past)
`source=VBD source=VBN`
Prefix *nita* (1st person sing. + future)
`child(aux) child(nsubj)=I`
Prefix *ana* (3rd person sing. + present)
`source=VBZ`
Prefix *wa* (3rd person plural)
`before=they child(nsubj)=NNS`
Suffix *tu* (1st person plural)
`child(nsubj)=she before=she`
Prefix *ha* (negative tense)
`source=no after=not`

Figure 3: Examples of highly weighted features learned by the inflection model. We selected a few frequent morphological features and show their top corresponding source context features.

data, we first construct an additional phrase-based translation model on the parallel corpus preprocessed to replace inflected surface words with their stems. We then extract a set of non-gappy phrases for each sentence (e.g., X → <attempted, пытаться_V>). The target side of each such phrase is re-inflected, conditioned on the source sentence, using the inflection model from §2. Each stem is given its most likely inflection.[7]

The original features extracted for the stemmed phrase are conserved, and the following features are added to help the decoder select good synthetic phrases:

- a binary feature indicating that the phrase is synthetic,
- the log-probability of the inflected forms according to our model,
- the count of words that have been inflected, with a separate feature for each morphological category in the supervised case.

Finally, these synthetic phrases are combined with the original translation rules obtained for the baseline system to produce an extended sentence-specific grammar which is used as input to the decoder. If a phrase already existing in the standard phrase table happens to be recreated, both phrases are kept and will compete with each other with different features in the decoder.

For example, for the large EN→RU system, 6% of all the rules used for translation are synthetic phrases, with 65% of these phrases being entirely new rules.

6 Translation Experiments

We evaluate our approach in the standard discriminative MT framework. We use cdec (Dyer et al., 2010) as our decoder and perform MIRA training to learn feature weights of the sentence translation model (Chiang, 2012). We compare the following configurations:

- A baseline system, using a 4-gram language model trained on the entire monolingual and bilingual data available.
- An enriched system with a class-based n-gram language model[8] trained on the monolingual data mapped to 600 Brown clusters. Class-based language modeling is a strong baseline for scenarios with high out-of-vocabulary rates but in which large amounts of monolingual target-language data are available.
- The enriched system further augmented with our inflected synthetic phrases. We expect the class-based language model to be especially

[7] Several reviewers asked about what happens when k-best inflections are added. The results for $k \in \{2, 4, 8\}$ range from no effect to an improvement over $k = 1$ of about 0.2 BLEU (absolute). We hypothesize that larger values of k could have a greater impact, perhaps in a more "global" model of the target string; however, exploration of this question is beyond the scope of this paper.

[8] For Swahili and Hebrew, $n = 6$; for Russian, $n = 7$.

helpful here and capture some basic agreement patterns that can be learned more easily on dense clusters than from plain word sequences.

Detailed corpus statistics are given in Table 1:

- The Russian data consist of the News Commentary parallel corpus and additional monolingual data crawled from news websites.[9]
- The Hebrew parallel corpus is composed of transcribed TED talks (Cettolo et al., 2012). Additional monolingual news data is also used.
- The Swahili parallel corpus was obtained by crawling the Global Voices project website[10] for parallel articles. Additional monolingual data was taken from the Helsinki Corpus of Swahili.[11]

We evaluate translation quality by translating and measuring the BLEU score of a 2000–3000 sentence-long evaluation corpus, averaging the results over 3 MIRA runs to control for optimizer instability (Clark et al., 2011). Table 4 reports the results. For all languages, using class language models improves over the baseline. When synthetic phrases are added, significant additional improvements are obtained. For the English–Russian language pair, where both supervised and unsupervised analyses can be obtained, we notice that expert-crafted morphological analyzers are more efficient at improving translation quality. Globally, the amount of improvement observed varies depending on the language; this is most likely indicative of the quality of unsupervised morphological segmentations produced and the kinds of grammatical relations expressed morphologically.

Finally, to confirm the effectiveness of our approach as corpus size increases, we use our technique on top of a state-of-the art English–Russian system trained on data from the 8th ACL Workshop on Machine Translation (30M words of bilingual text and 410M words of monolingual text). The setup is identical except for the addition of sparse rule shape indicator features and bigram cluster features. In these large scale conditions, the BLEU score improves from 18.8 to 19.6 with the addition of word clusters and reaches 20.0 with synthetic phrases. Details regarding this system are reported in Ammar et al. (2013).

Table 4: Translation quality (measured by BLEU) averaged over 3 MIRA runs.

	EN→RU	EN→HE	EN→SW
Baseline	14.7±0.1	15.8±0.3	18.3±0.1
+Class LM	15.7±0.1	16.8±0.4	18.7±0.2
+Synthetic			
unsupervised	16.2±0.1	17.6±0.1	19.0±0.1
supervised	16.7±0.1	—	—

7 Related Work

Translation into morphologically rich languages is a widely studied problem and there is a tremendous amount of related work. Our technique of synthesizing translation options to improve generation of inflected forms is closely related to the factored translation approach proposed by Koehn and Hoang (2007); however, an important difference to that work is that we use a discriminative model that conditions on source context to make "local" decisions about what inflections may be used before combining the phrases into a complete sentence translation.

Combination pre-/post-processing solutions are also frequently proposed. In these, the target language is generally transformed from multi-morphemic surface words into smaller units more amenable to direct translation, and then a post-processing step is applied independent of the translation model. For example, Oflazer and El-Kahlout (2007) experiment with partial morpheme groupings to produce novel inflected forms when translating into Turkish; Al-Haj and Lavie (2010) compare different processing schemes for Arabic. A related but different approach is to *enrich* the source language items with grammatical features (e.g., a source sentence like *John saw Mary* is preprocessed into, e.g., John+subj saw+msubj+fobj Mary+obj) so as to make the source and target lexicons have similar morphological contrasts (Avramidis and Koehn, 2008; Yeniterzi and Oflazer, 2010; Chang et al.,

[9] http://www.statmt.org/wmt13/translation-task.html

[10] http://sw.globalvoicesonline.org

[11] http://www.aakkl.helsinki.fi/cameel/corpus/intro.htm

2009). In general, this work suffers from the problem that it is extremely difficult to know *a priori* what the right preprocessing is for a given language pair, data size, and domain.

Several post-processing approaches have relied on supervised classifiers to predict the optimal complete inflection for an incomplete or lemmatized translation. Minkov et al. (2007) present a method for predicting the inflection of Russian and Arabic sentences aligned to English sentences. They train a sequence model to predict target morphological features from the lemmas and the syntactic structures of both aligned sentences and demonstrate its ability to recover accurately inflections on reference translations. Toutanova et al. (2008) apply this method to generate inflections after translation in two different ways: by rescoring inflected n-best outputs or by translating lemmas and re-inflecting them *a posteriori*. El Kholy and Habash (2012) follow a similar method and compare different approaches for generating rich morphology in Arabic after a translation step. Fraser et al. (2012) observe improvements for translation into German with a similar method. As in that work, we model morphological features rather than directly inflected forms. However, that work may be criticized for providing no mechanism to translate surface forms directly, even when evidence for a direct translation is available in the parallel data.

Unsupervised morphology has begun to play a role in translation between morphologically complex languages. Stallard et al. (2012) show that an unsupervised approach to Arabic segmentation performs as well as a supervised segmenter for source-side preprocessing (in terms of English translation quality). For translation *into* morphological rich languages, Clifton and Sarkar (2011) use an unsupervised morphological analyzer to produce morphological affixes in Finnish, injecting some linguistic knowledge in the generation process.

Several authors have proposed using conditional models to predict the probability of phrase translation in context (Gimpel and Smith, 2008; Chan et al., 2007; Carpuat and Wu, 2007; Jeong et al., 2010). Of particular note is the work of Subotin (2011), who use a conditional model to predict morphological features conditioned on rich linguistic features; however, this latter work also conditions on target context, which substantially complicates decoding.

Finally, synthetic phrases have been used for different purposes than generating morphology. Callison-Burch et al. (2006) expanded the coverage of a phrase table by adding synthesized phrases by paraphrasing source language phrases, Chen et al. (2011) produced "fabricated" phrases by paraphrasing both source and target phrases, and Habash (2009) created new rules to handle out-of-vocabulary words. In related work, Tsvetkov et al. (2013) used synthetic phrases to improve generation of (in)definite articles when translating into English from Russian and Czech, two languages which do not lexically mark definiteness.

8 Conclusion

We have presented an efficient technique that exploits morphologically analyzed corpora to produce new inflections possibly unseen in the bilingual training data. Our method decomposes into two simple independent steps involving well-understood discriminative models.

By relying on source-side context to generate additional local translation options and by leaving the choice of the full sentence translation to the decoder, we sidestep the difficulty of computing features on target translations hypotheses. However, many morphological processes (most notably, agreement) are most best modeled using target language context. To capture target context effects, we depend on strong target language models. Therefore, an important extension of our work is to explore the interaction of our approach with more sophisticated language models that more directly model morphology, e.g., the models of Bilmes and Kirchhoff (2003), or, alternatively, ways to incorporate target language context in the inflection model.

We also achieve language independence by exploiting unsupervised morphological segmentations in the absence of linguistically informed morphological analyses.

Code for replicating the experiments is available from `https://github.com/eschling/morphogen`; further details are available in (Schlinger et al., 2013).

Acknowledgments

This work was supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533. We would like to thank Kim Spasaro for curating the Swahili development and test sets, Yulia Tsvetkov for assistance with Russian, and the anonymous reviewers for their helpful comments.

References

Hassan Al-Haj and Alon Lavie. 2010. The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation. In *Proc. of AMTA*.

Waleed Ammar, Victor Chahuneau, Michael Denkowski, Greg Hanneman, Wang Ling, Austin Matthews, Kenton Murray, Nicola Segall, Yulia Tsvetkov, Alon Lavie, and Chris Dyer. 2013. The CMU machine translation systems at WMT 2013: Syntax, synthetic translation options, and pseudo-references. In *Proc. of WMT*.

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proc. of ACL*.

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proc. of NAACL*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL*.

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

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Improved statistical machine translation using paraphrases. In *Proc. of NAACL*.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proc. of EMNLP*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proc. of EAMT*.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. of ACL*.

Pi-Chuan Chang, Dan Jurafsky, and Christopher D. Manning. 2009. Disambiguating "DE" for Chinese–English machine translation. In *Proc. of WMT*.

Boxing Chen, Roland Kuhn, and George Foster. 2011. Semantic smoothing and fabrication of phrase pairs for SMT. In *Proc. of IWSLT*.

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

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *JMLR*, 13:1159–1187.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL*.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proc. of ACL*.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL*.

Ahmed El Kholy and Nizar Habash. 2012. Translate, predict or generate: Modeling rich morphology in statistical machine translation. In *Proc. of EAMT*.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in SMT. In *Proc. of EACL*.

Kevin Gimpel and Noah A. Smith. 2008. Rich source-side context for statistical machine translation. In *Proc. of WMT*.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proc. of ACL*.

Nizar Habash. 2009. REMOOV: A tool for online handling of out-of-vocabulary words in machine translation. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*.

Jan Hajič, Pavel Krbec, Pavel Květoň, Karel Oliva, and Vladimír Petkevič. 2001. Serial combination of rules and statistics: A case study in Czech tagging. In *Proc. of ACL*.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proc. of COLING*.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. 2010. A discriminative lexicon model for complex morphology. In *Proc. of AMTA*.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. *NIPS*, pages 641–648.

Mark Johnson. 2008. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proc. SIGMORPHON*.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proc. of EMNLP*.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proc. of EMNLP*.

André F.T. Martins, Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proc. of ACL*.

Kemal Oflazer and İlknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proc. of WMT*.

Eva Schlinger, Victor Chahuneau, and Chris Dyer. 2013. morphogen: Translation into morphologically rich languages with synthetic phrases. *Prague Bulletin of Mathematical Linguistics*, (100).

Serge Sharoff, Mikhail Kopotev, Tomaz Erjavec, Anna Feldman, and Dagmar Divjak. 2008. Designing and evaluating a Russian tagset. In *Proc. of LREC*.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proc. of EMNLP*.

David Stallard, Jacob Devlin, Michael Kayser, Yoong Keok Lee, and Regina Barzilay. 2012. Unsupervised morphology rivals supervised morphology for Arabic MT. In *Proc. of ACL*.

Michael Subotin. 2011. An exponential translation model for target language morphology. In *Proc. ACL*.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proc. of ACL*.

Yulia Tsvetkov, Chris Dyer, Lori Levin, and Archna Bhatia. 2013. Generating English determiners in phrase-based translation with synthetic translation options. In *Proc. of WMT*.

Reyyan Yeniterzi and Kemal Oflazer. 2010. Syntax-to-morphology mapping in factored phrase-based statistical machine translation from English to Turkish. In *Proc. of ACL*.

Boosting Cross-Language Retrieval by Learning Bilingual Phrase Associations from Relevance Rankings

Artem Sokolov and **Laura Jehl** and **Felix Hieber** and **Stefan Riezler**
Department of Computational Linguistics
Heidelberg University, 69120 Heidelberg, Germany
{sokolov,jehl,hieber,riezler}@cl.uni-heidelberg.de

Abstract

We present an approach to learning bilingual n-gram correspondences from relevance rankings of English documents for Japanese queries. We show that directly optimizing cross-lingual rankings rivals and complements machine translation-based cross-language information retrieval (CLIR). We propose an efficient boosting algorithm that deals with very large cross-product spaces of word correspondences. We show in an experimental evaluation on patent prior art search that our approach, and in particular a consensus-based combination of boosting and translation-based approaches, yields substantial improvements in CLIR performance. Our training and test data are made publicly available.

1 Introduction

The central problem addressed in Cross-Language Information Retrieval (CLIR) is that of translating or projecting a query into the language of the document repository across which retrieval is performed. There are two main approaches to tackle this problem: The first approach leverages the standard Statistical Machine Translation (SMT) machinery to produce a single best translation that is used as search query in the target language. We will henceforth call this the *direct translation* approach. This technique is particularly useful if large amounts of data are available in domain-specific form.

Alternative approaches avoid to solve the hard problem of word reordering, and instead rely on token-to-token translations that are used to project the query terms into the target language with a probabilistic weighting of the standard term tf-idf scheme. Darwish and Oard (2003) termed this method the *probabilistic structured query* approach. The advantage of this technique is an implicit query expansion effect due to the use of probability distributions over term translations (Xu et al., 2001). Recent research has shown that leveraging query context by extracting term translation probabilities from n-best direct translations of queries instead of using context-free translation tables outperforms both direct translation and context-free projection (Ture et al., 2012b; Ture et al., 2012a).

While direct translation as well as probabilistic structured query approaches use machine learning to optimize the SMT module, retrieval is done by standard search algorithms in both approaches. For example, Google's CLIR approach uses their standard proprietary search engine (Chin et al., 2008). Ture et al. (2012b; 2012a) use standard retrieval algorithms such as BM25 (Robertson et al., 1998). That means, machine learning in SMT-based approaches concentrates on the cross-language aspect of CLIR and is agnostic of the ultimate ranking task.

In this paper, we present a method to project search queries into the target language that is complementary to SMT-based CLIR approaches. Our method learns a table of n-gram correspondences by direct optimization of a ranking objective on relevance rankings of English documents for Japanese queries. Our model is similar to the approach of Bai et al. (2010) who characterize their technique as "Learning to rank with (a Lot of) Word Features". Given a set of search queries $\mathbf{q} \in \mathbb{R}^Q$ and docu-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1688–1699,
Seattle, Washington, USA, 18-21 October 2013.

ments $\mathbf{d} \in \mathbb{R}^D$, where the j^{th} dimension of a vector indicates the occurrence of the j^{th} word for dictionaries of size Q and D, we want to learn a score $f(\mathbf{q}, \mathbf{d})$ between a query and a given document using the model[1]

$$f(\mathbf{q}, \mathbf{d}) = \mathbf{q}^\top W \mathbf{d} = \sum_{i=1}^{Q} \sum_{j=1}^{D} q_i W_{ij} d_j.$$

We take a pairwise ranking approach to optimization. That is, given labeled data in the form of a set $\mathcal{R}$ of tuples $(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$, where $\mathbf{d}^+$ is a relevant (or higher ranked) document and $\mathbf{d}^-$ an irrelevant (or lower ranked) document for query $\mathbf{q}$, the goal is to find a weight matrix $W \in \mathbb{R}^{Q \times D}$ such that $f(\mathbf{q}, \mathbf{d}^+) > f(\mathbf{q}, \mathbf{d}^-)$ for all data tuples from $\mathcal{R}$. The scoring model learns weights for all possible correspondences of query terms and document terms by directly optimizing the ranking objective at hand. Such a phrase table contains domain-specific word associations that are useful to discern relevant from irrelevant documents, something that is orthogonal and complementary to standard SMT models.

The challenge of our approach can be explained by constructing a joint feature map ϕ from the outer product of the vectors $\mathbf{q}$ and $\mathbf{d}$ where

$$\phi_{((i-1)D+j)}(\mathbf{q}, \mathbf{d}) = (\mathbf{q} \otimes \mathbf{d})_{ij} = (\mathbf{q}\mathbf{d}^\top)_{ij}. \quad (1)$$

Using this feature map, we see that the score function f can be written in the standard form of a linear model that computes the inner product between a weight vector w and a feature vector ϕ where $w, \phi \in \mathbb{R}^{Q \times D}$ and

$$f(\mathbf{q}, \mathbf{d}) = \langle w, \phi(\mathbf{q}, \mathbf{d}) \rangle. \quad (2)$$

While various standard algorithms exist to optimize linear models, the difficulty lies in the memory footprint and capacity of the word-based model. A full-sized model includes $Q \times D$ parameters which is easily in the billions even for moderately sized dictionaries. Clearly, an efficient implementation and remedies against overfitting are essential.

The main contribution of our paper is the presentation of algorithms that make learning a phrase table by direct rank optimization feasible, and an experimental verification of the benefits of this approach, especially with regard to a combination of the orthogonal information sources of ranking-based and SMT-based CLIR approaches. Our approach builds upon a boosting framework for pairwise ranking (Freund et al., 2003) that allows the model to grow incrementally, thus avoiding having to deal with the full matrix W. Furthermore, we present an implementation of boosting that utilizes parallel estimation on bootstrap samples from the training set for increased efficiency and reduced error (Breiman, 1996). Our "bagged boosting" approach allows to combine incremental feature selection, parallel training, and efficient management of large data structures.

We show in an experimental evaluation on large-scale retrieval on patent abstracts that our boosting approach is comparable in MAP and improves significantly by 13-15 PRES points over very competitive translation-based CLIR systems that are trained on 1.8 million parallel sentence pairs from Japanese-English patent documents. Moreover, a combination of the orthogonal information learned in ranking-based and translation-based approaches improves over 7 MAP points and over 15 PRES points over the respective translation-based system in a consensus-based voting approach following the Borda Count technique (Aslam and Montague, 2001).

2 Related Work

Recent research in CLIR follows the two main paradigms of direct translation and probabilistic structured query approaches. An example for the first approach is the work of Magdy and Jones (2011) who presented an efficient technique to adapt off-the-shelf SMT systems for CLIR by training them on data pre-processed for retrieval (case folding, stopword removal, stemming). Nikoulina et al. (2012) presented an approach to direct translation-based CLIR where the n-best list of an SMT system is re-ranked according to the MAP performance of the translated queries. The probabilistic structured query approach has seen a lot of work on context-aware query expansion across languages, based on various similarity statistics (Ballesteros and Croft, 1998; Gao et al., 2001; Lavrenko et al., 2002; Gao

[1] With bold letters we denote vectors for query $\mathbf{q}$ and document $\mathbf{d}$. Vector components are denoted with normal font letters and indices (e.g., q_i).

et al., 2007). At the time of writing this paper, the most recent extension to this paradigm is Ture et al. (2012a). In addition to projecting terms from n-best translations, they propose a projection extracted from the hierarchical phrase- based grammar models, and a scoring method based on multi-token terms. Since the latter techniques achieved only marginal improvements over the context-sensitive query translation from n-best lists, we did not pursue them in our work.

CLIR in the context of patent prior art search was done as extrinsic evaluation at the NTCIR PatentMT[2] workshops until 2010, and has been ongoing in the CLEF-IP[3] benchmarking workshops since 2009. However, most workshop participants did either not make use of automatic translation at all, or they used an off-the-shelf translation tool. This is due to the CLEF-IP data collection where parts of patent documents are provided as manual translations into three languages. In order to evaluate CLIR in a truly cross-lingual scenario, we created a large patent CLIR dataset where queries and documents are Japanese and English patent abstracts, respectively.

Ranking approaches to CLIR have been presented by Guo and Gomes (2009) who use pairwise ranking for patent retrieval. Their method is a classical learning-to-rank setup where retrieval scores such as tf-idf or BM25 are combined with domain knowledge on patent class, inventor, date, location, etc. into a dense feature vector of a few hundred features. Methods to learn word-based translation correspondences from supervised ranking signals have been presented by Bai et al. (2010) and Chen et al. (2010). These approaches tackle the problem of complexity and capacity of the cross product matrix of word correspondences from different directions. The first proposes to learn a low rank representation of the matrix; the second deploys sparse online learning under ℓ_1 regularization to keep the matrix small. Both approaches are mainly evaluated in a monolingual setting. The cross-lingual evaluation presented in Bai et al. (2010) uses weak translation-based baselines and non-public data such that a direct comparison is not possible.

A combination of bagging and boosting in the context of retrieval has been presented by Pavlov et al. (2010) and Ganjisaffar et al. (2011). This work is done in a standard learning-to-rank setup using a few hundred dense features trained on hundreds of thousands of pairs. Our setup deals with billions of sparse features (from the cross-product of the unrestricted dictionaries) trained on millions of pairs (sampled from a much larger space). Parallel boosting where all feature weights are updated simultaneously has been presented by Collins et al. (2002) and Canini et al. (2010). The first method distributes the gradient calculation for different features among different compute nodes. This is not possible in our approach because we construct the cross-product matrix on-the-fly. The second approach requires substantial efforts in changing the data representation to use the MapReduce framework. Overall, one of the goals of our work is sequential updating for implicit feature selection, something that runs contrary to parallel boosting.

3 CLIR Approaches

3.1 Direct translation approach

For direct translation, we use the SCFG decoder cdec (Dyer et al., 2010)[4] and build grammars using its implementation of the suffix array extraction method described in Lopez (2007). Word alignments are built from all parallel data using mgiza[5] and the Moses scripts[6]. SCFG models use the same settings as described in Chiang (2007). Training and querying of a modified Kneser-Ney smoothed 5-gram language model are done on the English side of the training data using KenLM (Heafield, 2011)[7]. Model parameters were optimized using cdec's implementation of MERT (Och (2003)).

At retrieval time, all queries are translated sentence-wise and subsequently re-joined to form one query per patent. Our baseline retrieval system uses the Okapi BM25 scores for document ranking.

[2] http://research.nii.ac.jp/ntcir/ntcir/

[3] http://www.ifs.tuwien.ac.at/~clef-ip/

[4] https://github.com/redpony/cdec

[5] http://www.kyloo.net/software/doku.php/mgiza:overview

[6] http://www.statmt.org/moses/?n=Moses.SupportTools

[7] http://kheafield.com/code/kenlm/estimation/

3.2 Probabilistic structured query approach

Early Probabilistic Structured Query approaches (Xu et al., 2001; Darwish and Oard, 2003) represent translation options by lexical, i.e., token-to-token translation tables that are estimated using standard word alignment techniques (Och and Ney, 2000). Later approaches (Ture et al., 2012b; Ture et al., 2012a) extract translation options from the decoder's n-best list for translating a particular query. The central idea is to let the language model choose fluent, context-aware translations for each query term during decoding. This retains the desired query-expansion effect of probabilistic structured models, but it reduces query drift by filtering translations with respect to the context of the full query.

A projection of source language query terms $f \in F$ into the target language is achieved by representing each source token f by its probabilistically weighted translations. The score of target document E, given source language query F, is computed by calculating the BM25 rank over projected term frequency and document frequency weights as follows:

$$score(E|F) = \sum_{f \in F} BM25(tf(f,E), df(f)) \quad (3)$$

$$tf(f,E) = \sum_{e \in E_f} tf(e,E)p(e|f)$$

$$df(f) = \sum_{e \in E_f} df(e)p(e|f)$$

where $E_f = \{e \in E | p(e|f) > p_L\}$ is the set of translation options for query term f with probability greater than p_L. We also use a cumulative threshold p_C so that only the most probable options are added until p_C is reached.

Ture et al. (2012b; 2012a) achieved best retrieval performance by interpolating between (context-free) lexical translation probabilities p_{lex} estimated on symmetrized word alignments, and (context-aware) translation probabilities p_{nbest} estimated on the n-best list of an SMT decoder:

$$p(e|f) = \lambda p_{nbest}(e|f) + (1-\lambda)p_{lex}(e|f) \quad (4)$$

$p_{nbest}(e|f)$ is estimated by calculating expectations of term translations from k-best translations:

$$p_{nbest}(e|f) = \frac{\sum_{k=1}^{n} a_k(e,f)\mathcal{D}(k,F)}{\sum_{k=1}^{n}\sum_{e'} a_k(e',f)\mathcal{D}(k,F)}$$

where $a_k(e,f)$ is a function indicating an alignment of target term e to source term f in the k^{th} derivation of query F, and $\mathcal{D}(k,F)$ is the model score of the k^{th} derivation in the n-best list for query F.

We use the same hierarchical phrase-based system that was used for direct translation to calculate n-best translations for the probabilistic structured query approach. This allows us to extract word alignments between source and target text for F from the SCFG rules used in the derivation. The concept of self-translation is covered by the decoder's ability to use pass-through rules if words or phrases cannot be translated.

Probabilistic structured queries that include context-aware estimates of translation probabilities require a preservation of sentence-wise context-sensitivity also in retrieval. Thus, unlike the direct translation approach, we compute weighted term and document frequencies for each sentence s in query F separately. The scoring (3) of a target document for a multiple sentence query then becomes:

$$score(E|F) = \sum_{s \text{ in } F} \sum_{f \in s} BM25(tf(f,E), df(f))$$

3.3 Direct Phrase Table Learning from Relevance Rankings

Pairwise Ranking using Boosting The general form of the RankBoost algorithm (Freund et al., 2003; Collins and Koo, 2005) defines a scoring function $f(\mathbf{q},\mathbf{d})$ on query $\mathbf{q}$ and document $\mathbf{d}$ as a weighted linear combination of T *weak learners* h_t such that $f(\mathbf{q},\mathbf{d}) = \sum_{t=1}^{T} w_t h_t(\mathbf{q},\mathbf{d})$. Weak learners can belong to an arbitrary family of functions, but in our case they are restricted to the simplest case of unparameterized indicator functions selecting components of the feature vector $\phi(\mathbf{q},\mathbf{d})$ in (1) such that f is of the standard linear form (2). In our experiments, these features indicate the presence of pairs of uni- and bi-grams from the source-side vocabulary of query terms and the target-side vocabulary of document-terms, respectively. Furthermore, in order to simulate the pass-through behavior of SMT, we introduce additional features to the model that indicate the identity of terms in source and target. All identity features have the same fixed weight β, which is found on the development set.

For training, we are given labeled data in the form

of a set $\mathcal{R}$ of tuples $(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$, where $\mathbf{d}^+$ is a relevant (or higher ranked) document and $\mathbf{d}^-$ an irrelevant (or lower ranked) document for query $\mathbf{q}$. RankBoost's objective is to correctly rank query-document pairs such that $f(\mathbf{q}, \mathbf{d}^+) > f(\mathbf{q}, \mathbf{d}^-)$ for all data tuples from $\mathcal{R}$. RankBoost achieves this by optimizing the following convex exponential loss:

$$\mathcal{L}_{exp} = \sum_{(\mathbf{q},\mathbf{d}^+,\mathbf{d}^-)\in\mathcal{R}} D(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-) e^{f(\mathbf{q},\mathbf{d}^-)-f(\mathbf{q},\mathbf{d}^+)},$$

where $D(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$ is a non-negative importance function on pairs of documents for a given $\mathbf{q}$.

We optimize $\mathcal{L}_{exp}$ in a greedy iterative fashion, which closely follows an efficient algorithm of Collins and Koo (2005) for the case of binary-valued h. In each step, the single feature h is selected that provides the largest decrease of $\mathcal{L}_{exp}$, i.e., that has the largest projection on the direction of the gradient $\nabla_h \mathcal{L}_{exp}$. Because of the sequential nature of the algorithm, RankBoost implicitly performs automatic feature selection and regularization (Rosset et al., 2004), which is crucial to reduce complexity and capacity for our application.

Parallelization and Bagging To achieve parallelization we use a variant of bagging (Breiman, 1996) on top of boosting, which has been observed to improve performance, reduce variance and is trivial to parallelize. The procedure is described as part of Algorithm 1: From the set of preference pairs $\mathcal{R}$, draw S equal-sized samples with replacement and distribute to nodes. Then, using each of the samples as a training set, separate boosting models $\{w_t^s, h_t^s\}, s = 1 \dots S$ are trained that contain the same number of features $t = 1 \dots T$. Finally the models are averaged: $f(\mathbf{q}, \mathbf{d}) = \frac{1}{S} \sum_t \sum_s w_t^s h_t^s(\mathbf{q}, \mathbf{d})$.

Algorithm The entire training procedure is outlined in Algorithm 1. For each possible feature h we maintain auxiliary variables W_h^+ and W_h^-:

$$W_h^{\pm} = \sum_{(\mathbf{q},\mathbf{d}^+,\mathbf{d}^-):h(\mathbf{q},\mathbf{d}^+)-h(\mathbf{q},\mathbf{d}^-)=\pm 1} D(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-),$$

which are the cumulative weights of correctly and incorrectly ranked instances by a candidate feature h. The absolute value of $\partial \mathcal{L}_{exp} / \partial h$ can be expressed as $\left|\sqrt{W_h^+} - \sqrt{W_h^-}\right|$ which is used as feature selection criterion (Collins and Koo, 2005). The optimum of minimizing $\mathcal{L}_{exp}$ over w (with fixed h) can be shown to be $w = \frac{1}{2} \ln \frac{W_h^+ + \epsilon Z}{W_h^- + \epsilon Z}$, where ϵ is a smoothing parameter to avoid problems with small $W_h^{\pm}$ (Schapire and Singer, 1999), and $Z = \sum_{(\mathbf{q},\mathbf{d}^+,\mathbf{d}^-)\in\mathcal{R}} D(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$. Furthermore, for each step t of the learning process, values of D are updated to concentrate on pairs that have not been correctly ranked so far:

$$D_{t+1} = D_t \cdot e^{w_t \left(h_t(\mathbf{q},\mathbf{d}^-) - h_t(\mathbf{q},\mathbf{d}^+)\right)}. \qquad (5)$$

Finally, to speed up learning, on iteration t we recalculate $W_h^{\pm}$ only for those h that cooccur with previously selected h_t and keep the rest unchanged (Collins and Koo, 2005).

Algorithm 1: Bagged Boosting

Input: training tuples $\mathcal{R}$, max number of features T, initial D_0, smoothing param. $\epsilon \simeq 10^{-5}$
Initialize:
from $\mathcal{R}$ draw S samples with replacement and distribute to nodes
Learn:
for *all samples* $s = 1 \dots S$ *in parallel* **do**
  calculate W_h^+,W_h^-, Z on sample's data
  for *all* $t = 1 \dots T$ **do**
    choose $h_t = \arg\max_h \left|\sqrt{W_h^+} - \sqrt{W_h^-}\right|$
    and $w_t = \frac{1}{2} \ln \frac{W_h^+ + \epsilon Z}{W_h^- + \epsilon Z}$
    update D_t according to (5)
    update $W_h^{\pm}$ for all h that cooccur with h_t
  end
  return to master $\{h_t^s, w_t^s\}, t = 1 \dots T$
end
Bagging:
return scoring function
$f(\mathbf{q}, \mathbf{d}) = \frac{1}{S} \sum_t \sum_s w_t^s h_t^s(\mathbf{q}, \mathbf{d})$

Implementation Because of the total number of features (billions) there are several obstacles for the straight-forward implementation of Algorithm 1.

First, we cannot directly access all pairs $(\mathbf{q}, \mathbf{d})$ containing a particular feature h needed for calculating $W_h^{\pm}$. Building an inverted index is complicated as it needs to fit into memory for fast fre-

quent access[8]. We resort to the on-the-fly creation of the cross-product space of features, following prior work by Grangier and Bengio (2008) and Goel et al. (2008). That is, while processing a pair $(\mathbf{q}, \mathbf{d})$, we update $W_h^{\pm}$ for all h found for the pair.

Second, even if the explicit representation of all features is avoided by on-the-fly feature construction, we still need to keep all $W_h^{\pm}$ in addressable RAM. To achieve that we use hash kernels (Shi et al., 2009) and map original features into b-bit integer hashes. The values $W_{h'}^{\pm}$ for new, "hashed", features h' become $W_{h'}^{\pm} = \sum_{h:HASH(h)=h'} W_h^{\pm}$. We used the MurmurHash3 function on the UTF-8 representations of features and $b = 30$ (resulting in more than 1 billion distinct hashes).

4 Model Combination by Borda Counts

SMT-based approaches to CLIR and our boosting approach have different strengths. The SMT-based approaches produce fluent translations that are useful for matching general passages written in natural language. Both baseline SMT-based approaches presented above are agnostic of the ultimate retrieval task and are not specifically adapted for it. The boosting method, on the other hand, learns domain-specific word associations that are useful to discern relevant from irrelevant documents. In order to combine these orthogonal sources of information in a way that democratically respects each approach we use Borda Counts, i.e., a consensus-based voting procedure that has been successfully employed to aggregate ranked lists of documents for metasearch (Aslam and Montague, 2001).

We implemented a weighted version of the Borda Count method where each voter has a fixed amount of voting points which she is free to distribute among the candidates to indicate the amount of preference she is giving to each of them. In the case of retrieval, for each $\mathbf{q}$, the candidates are the scored documents in the retrieved subset of the whole document set. The aggregate score f_{agg} for two rankings $f_1(\mathbf{q}, \mathbf{d})$ and $f_2(\mathbf{q}, \mathbf{d})$ for all $(\mathbf{q}, \mathbf{d})$ in the test set is then:

$$f_{agg}(\mathbf{q}, \mathbf{d}) = \kappa \frac{f_1(\mathbf{q}, \mathbf{d})}{\sum_{\mathbf{d}} f_1(\mathbf{q}, \mathbf{d})} + (1-\kappa) \frac{f_2(\mathbf{q}, \mathbf{d})}{\sum_{\mathbf{d}} f_2(\mathbf{q}, \mathbf{d})}.$$

In practice, the normalizations sum over the top K retrieved documents. If a document is present only in the top-K list of one system, its score is considered zero for the other system. The aggregated scores $f_{agg}(\mathbf{q}, \mathbf{d})$ are sorted in descending order and top K scores are kept for evaluation.

Using the terminology proposed by Belkin et al. (1995), combining several systems' scores with Borda Counts can be viewed as the "data fusion" approach to IR, that merges outputs of the systems, while the PSQ baseline is an example of the "query combination" approach that extends the query at the input. Both techniques were earlier found to have similar performance in CLIR tasks based on direct translation, with a preference for the data fusion approach (Jones and Lam-Adesina, 2002).

5 Translation and Ranking Data

5.1 Parallel Translation Data

For Japanese-to-English patent translation we used data provided by the organizers of the NTCIR[9] workshop for the JP-EN PatentMT subtask. In particular, we used the data provided for NTCIR-7 (Fujii et al., 2008), consisting of 1.8 million parallel sentence pairs from the years 1993-2002 for training. For parameter tuning we used the development set of the NTCIR-8 test collection, consisting of 2,000 sentence pairs. The data were extracted from the description section of patents published by the Japanese Patent Office (JPO) and the United States Patent and Trademark Office (USPTO) by the method described in Utiyama and Isahara (2007).

Japanese text was segmented using the MeCab[10] toolkit. Following Feng et al. (2011), we applied a modified version of the compound splitter described in Koehn and Knight (2003) to katakana terms, which are often transliterations of English compound words. As these are usually not split by MeCab, they can cause a large number of out-of-vocabulary terms.

[8] It is possible to construct separate query and document inverted indices and intersect them on the fly to determine the set of documents that contains some pair of words. In practice, however, we found the overhead of set intersection during each feature access prohibitive.

[9] http://research.nii.ac.jp/ntcir/ntcir/

[10] https://code.google.com/p/mecab/

	#queries	#relevant	#unique docs
train	107,061	1,422,253	888,127
dev	2,000	26,478	25,669
test	2,000	25,173	24,668

Table 1: Statistics of ranking data.

For the English side of the training data, we applied a modified version of the tokenizer included in the Moses scripts. This tokenizer relies on a list of non-breaking prefixes which mark expressions that are usually followed by a "." (period). We customized the list of prefixes by adding some abbreviations like "Chem", "FIG" or "Pat", which are specific to patent documents.

5.2 Ranking Data from Patent Citations

Graf and Azzopardi (2008) describe a method to extract relevance judgements for patent retrieval from patent citations. The key idea is to regard patent documents that are cited in a query patent, either by the patent applicant, or by the patent examiner or in a patent office's search report, as relevant for the query patent. Furthermore, patent documents that are related to the query patent via a patent family relationship, i.e., patents granted by different patent authorities but related to the same invention, are regarded as relevant. We assign three integer relevance levels to these three categories of relationships, with highest relevance (3) for family patents, lower relevance for patents cited in search reports by patent examiners (2), and lowest relevance level (1) for applicants' citations. We also include all patents which are in the same patent family as an applicant or examiner citation to avoid false negatives. This methodology has been used to create patent retrieval data at CLEF-IP[11] and proved very useful to automatically create a patent retrieval dataset for our experiments.

For the creation of our dataset, we used the MAREC[12] citation graph to extract patents in citation or family relation. Since the Japanese portion of the MAREC corpus only contains English abstracts, but not the Japanese full texts, we merged the patent documents in the NTCIR-10 test collection described above with the Japanese (JP) section of MAREC. Title, abstract, description and claims were added to the MAREC-JP data if the document was available in NTCIR. In order to keep parallel data for SMT training separate from ranking data, we used only data from the years 2003-2005 to extract training data for ranking, and two small datasets of 2,000 queries each from the years 2006-2007 for development and testing. Table 1 gives an overview over the data used for ranking. For development and test data, we randomly added irrelevant documents from the NTCIR-10 collection until we obtained two pools of 100,000 documents. The necessary information to reproduce the exact train, development and test data samples is downloadable from authors' webpage[13].

The experiments reported here use only the abstract of the Japanese and English patents in our training, development and test collection.

6 Experiments

6.1 System Development

System development and evaluation in our experiments was done on the ranking data described in the previous section (see Table 1). We report Mean Average Precision (MAP) scores, using the `trec_eval` (ver. 8.1) script from the TREC evaluation campaign[14], with a limit of top $K = 1,000$ retrieved documents for each query. Furthermore, we use the Patent Retrieval Evaluation Score (PRES)[15] introduced by Magdy and Jones (2010). This metric accounts for both precision and recall. In the study by Magdy and Jones (2010), PRES agreed with MAP in almost 80% of cases, and both agreed on the ranks of the best and the worst IR system. Both MAP and PRES scores are reported in the same range $[0, 1]$, and 0.01 stands for 1 MAP (PRES) point. Statistical significance of pairwise system comparisons was assessed using the paired randomization test (Noreen, 1989; Smucker et al., 2007).

For each system, optimal meta-parameter settings were found by choosing the configuration with highest MAP score on the development set. These results

[11] http://www.ifs.tuwien.ac.at/~clef-ip/

[12] http://www.ifs.tuwien.ac.at/imp/marec.shtml

[13] http://www.cl.uni-heidelberg.de/statnlpgroup/boostclir

[14] http://trec.nist.gov/trec_eval

[15] http://www.computing.dcu.ie/~wmagdy/Scripts/PRESeval.htm

method	MAP		PRES	
	dev	test	dev	test
1 DT	0.2636	0.2555	0.5669	0.5681
2 PSQ lexical table	0.2520	0.2444	0.5445	0.5498
3 PSQ n-best table	0.2698	0.2659	0.5789	0.5851
Boost-1g	0.2064	^{123}0.1982	0.5850	^{12}0.6122
Boost-2g	0.2526	^{3}0.2474	0.6900	^{123}0.7196

Table 2: MAP and PRES scores for CLIR methods (best configurations) on the development and test sets. Prefixed numbers denote statistical significance of a pairwise comparison with the baseline indicated by the superscript. For example, the bottom right result shows that Boost-2g is significantly better than DT (method 1), PSQ lexical table (method 2) and PSQ n-best table (method 3).

(together with PRES results) are shown in the second and fourth column of Table 2.

The direct translation approach (DT) was developed in three configurations: no stopword filtering, small stopword list (52 words) and a large stopword list (543 words). The last configuration achieved the highest score (MAP 0.2636).

The probabilistic structured query (PSQ) approach was developed using the lexical translation table and the translation table estimated on the decoder's n-best list, both optionally pruned with a variable lower p_L and cumulative p_C threshold on the word pair probability in the table (Section 3.2). A further meta-parameter of PSQ was whether to use standard or unique n-best lists. Finally, all variants were coupled with the same stopword filters as in the DT approach. The configurations that achieved the highest scores were: MAP 0.2520 for PSQ with a lexical table ($p_L = 0.01, p_C = 0.95$, no stopword filtering), and MAP 0.2698 for PSQ with a translation table estimated on the n-best list ($p_L = 0.005, p_C = 0.95$, large stopword list). Interpolating between lexical and n-best tables did not improve results in our experiments, thus we set $\lambda = 1$ in equation (4).

Each SMT-based system was run with 4 different MERT optimizations, leading to variations of less than 1 MAP point for each system. The best configurations for DT and PSQ on the development set were fixed and used for evaluation on the test set.

Training of the boosting approach (Boost) was done in parallel on bootstrap samples from the training data. First, a query $\mathbf{q}$ (i.e., a Japanese abstract) was sampled uniformly from all training queries.

method	MAP		PRES	
	dev	test	dev	test
DT + PSQ n-best	0.2778	*0.2726	0.5884	*0.5942
DT + Boost-1g	0.2778	*0.2728	0.6157	*0.6225
DT + Boost-2g	0.3309	***0.3300**	0.7132	***0.7279**
PSQ lexical + Boost-1g	0.2695	*0.2653	0.6068	*0.6131
PSQ lexical + Boost-2g	0.3215	***0.3187**	0.7071	***0.7240**
PSQ n-best + Boost-1g	0.2863	*0.2850	0.6309	*0.6402
PSQ n-best + Boost-2g	0.3439	***0.3416**	0.7212	***0.7376**

Table 3: MAP and PRES scores of the aggregated models on the development and test sets. Development scores correspond to peaks in Figures 1 and 3, respectively, for MAP and PRES; test scores are given for the κ's delivering these peaks on the development set. Prefixed $*$ indicates statistical significance of the result difference between aggregated system and the respective translation-based system used in the aggregation.

Then we sampled independently and uniformly a relevant document $\mathbf{d}^+$ (i.e., an English abstract) from the English patents marked relevant for the Japanese patent, and a random document $\mathbf{d}^-$ from the whole pool of English patent abstracts. If $\mathbf{d}^-$ had a relevance score greater or equal to the relevance score of $\mathbf{d}^+$, it was resampled. The initial importance weight D_0 for a triplet $(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$ was set to the positive difference in relevance scores for $\mathbf{d}^+$ and $\mathbf{d}^-$. Each bootstrap sample consisted of 10 pairs of documents for each of $10,000$ queries, resulting in $100,000$ training instances per sample.

The Boost approach was developed for uni-gram and combined uni- and bi-gram versions. We observed that the performance of the Boost method continuously improved with the number of iterations T and with the number of samples S, but saturated at about 15-20 samples without visible over-fitting in the tested range of T. Therefore we arbitrarily stopped training after obtaining $5,000$ features per sample, and used 35 samples for uni-gram version and 65 samples for the combined bi-gram version, resulting in models with 104K and 172K unique features, respectively. The optimal values for the pass-through weight β were found to be 0.3 and 0.2 for the uni-gram and bi-gram models on the development set. The best configuration of uni-gram and bi-gram model achieved MAP scores of 0.2064 and 0.2526 the development set. Using stopword filters during training did not improve the results here.

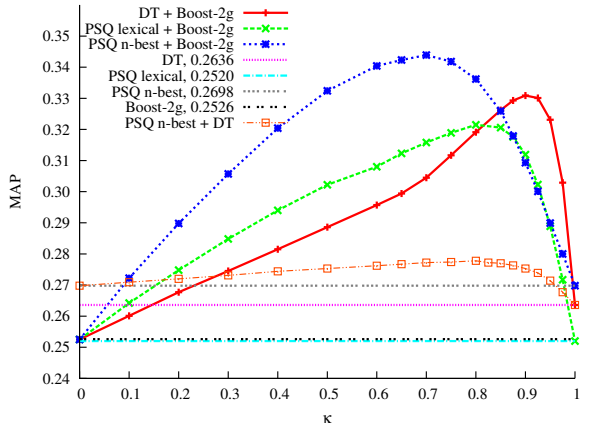


Figure 1: MAP rank aggregation for combinations of the bi-gram boosting and the baselines on the dev set.

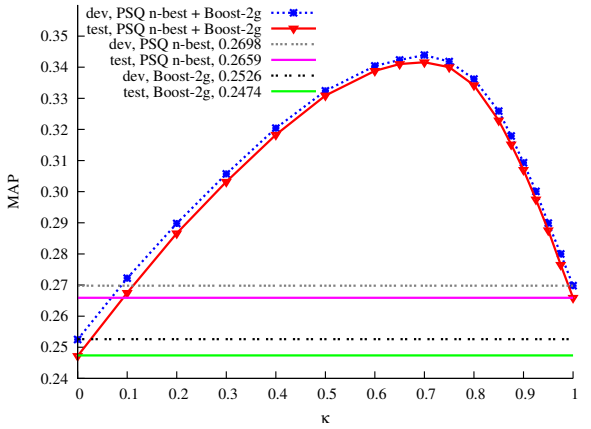


Figure 2: MAP rank aggregation for the bi-gram boosting and the "PSQ n-best table" approach on dev and test sets.

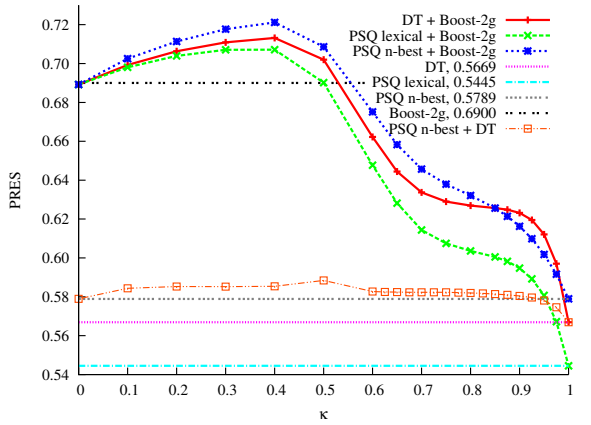


Figure 3: PRES rank aggregation for combinations of the bi-gram boosting and the baselines on the dev set.

6.2 Testing and Model Combination

The third and the fifth columns of Table 2 give a comparison of the MAP scores of the baseline approaches and the Boost model evaluated individually on the test set. Each score corresponds to the best configuration found on the development set. We see that the PSQ approach using n-best lists for projection outperforms all other methods in terms of MAP, but loses to both Boost approaches when evaluated with PRES. Direct translation is about 1 MAP point lower than PSQ n-best; Boost with combined uni- and bi-grams is another 0.8 MAP points worse, but is better in terms of PRES, especially for the bi-gram version. Given the fact that the complex SMT system behind the direct translation and PSQ approach is trained and tuned on very large in-domain datasets, the performance of the bare phrase table induced by the Boost method is respectable.

Our best results are obtained by a combination of the orthogonal information sources of the SMT and the Boost approaches. We evaluated the Borda Count aggregation scheme on the development data in order to find the optimal value for $\kappa \in [0, 1]$. The interpolation was done for the best combined uni- and bi-gram boosting model with the best variants of the DT and PSQ approaches. As can be seen from Figures 1 and 3, rank aggregation by Borda Count outperforms both individual approaches by a large margin. Figure 2 verifies that the results are transferable from the development set to the test set. The best performing system combination on the development data is also optimal on the test data.

Table 3 shows the retrieval performance of the best baseline model (PSQ n-best) combined with the best Boost model (bi-gram), with an impressive gain of over 7 MAP points (15 PRES points) over the best individual baseline result from Table 2. Even when, according to the PRES measure (Figure 3), the Boost-2g system is better on its own, injecting complementary information from the PSQ or DT approach still contributes several points. Similar gains are obtained by model combination of the DT approach with the best Boost model. However, a combination of the SMT-based CLIR approaches DT and PSQ barely improved results over the best input model. In summary, aggregating rankings is helpful for orthogonal systems, but not for systems including similar information.

6.3 Analysis

Table 4 lists some of the top-200 selected features for the boosting approach (the most common translation of the Japanese term is put in subscript).

We see that the direct ranking approach is able to penalize uni- and bi-gram cooccurrences that are harmful for retrieval by assigning them a negative weight, e.g., the pairing of 解決$_{\text{resolution}}$ with *image*. Pairs of uni- and bi-grams that are useful for retrieval are boosted by positive weights, e.g., the pair 圧縮$_{\text{compression}}$,機$_{\text{machine}}$ and *compressor* captures an important compound. Further examples, not shown in the table, are matches of the same source (target) n-gram with several different target (source) n-grams, e.g., the Japanese term 画像$_{\text{image}}$ is paired not only with its main translation, but also with dozens of related notions: *video*, *picture*, *scanning*, *printing*, *photosensitive*, *pixel*, *background* etc. This has a query expansion effect that is not possible in systems that use one translation or a small list of n-best translations. In addition, associations of source n-grams with overlapping target n-grams help boost the final score: e.g., the same term 画像$_{\text{image}}$ is positively paired with target bi-grams as {*an,original*}, {*original,image*} and {*image,for*}. This has the effect of compensating for the lack of handling phrase overlaps in an SMT decoder.

t	h_t **(uni- & bi-grams)**	w_t
1	層$_{\text{layer}}$ - layer	1.29
2	データ$_{\text{data}}$ - data	1.13
3	回路$_{\text{circuit}}$ - circuit	1.13
76	で$_{\text{in}}$ - voltage	-0.39
77	導$_{\text{guide}}$,電$_{\text{power}}$ - conductive	1.25
81	解決$_{\text{resolution}}$ - image	-0.25
99	変速$_{\text{speed}}$ - transmission	1.68
100	液晶$_{\text{LCD}}$ - liquid,crystal	1.73
123	力$_{\text{power}}$ - force	0.91
124	圧縮$_{\text{compression}}$,機$_{\text{machine}}$ - compressor	2.83
132	ケーブル$_{\text{cable}}$ - cable	1.81
133	超$_{\text{hyper}}$,音波$_{\text{sound wave}}$ - ultrasonic	3.34
169	粒子$_{\text{particle}}$ - particles	1.57
170	算出$_{\text{calculation}}$ - for,each	1.14
184	ロータ$_{\text{rotor}}$ - rotor	2.01
185	検出$_{\text{detection}}$,器$_{\text{vessel}}$ - detector	1.43

Table 4: Examples of the features found by boosting.

7 Conclusion

We presented a boosting approach to induce a table of bilingual n-gram correspondences by direct preference learning on relevance rankings. This table can be seen as a phrase table that encodes word-based information that is orthogonal and complementary to the information in standard translation-based CLIR approaches. We compared our boosting approach to very competitive CLIR baselines that use a complex SMT system trained and tuned on large in-domain datasets. Furthermore, our patent retrieval setup gives SMT-based approaches an advantage in that queries consist of several normal-length sentences, as opposed to the short queries common to web search. Despite this and despite the tiny size (about 170K parameters) of the boosting phrase table, compared to standard SMT phrase tables, this approach reached performance similar to direct translation using a full SMT model in terms of MAP, and was significantly better in terms of PRES. Overall, we obtained the best results by a model combination using consensus- based voting where the best SMT-based approach was combined with the boosting phrase table (gaining more than 7 MAP or 15 PRES points). We attribute this to the fact that the boosting approach augments SMT approaches with valuable information that is hard to get in approaches that are agnostic about the ranking data and the ranking task at hand.

The experimental setup presented in this paper uses relevance links between patent abstracts as ranking data. While this technique is useful to develop patent retrieval systems, it would be interesting to see if our results transfer to patent retrieval scenarios where full patent documents are used instead of only abstracts, or to standard CLIR scenarios that use short search queries in retrieval.

Acknowledgements

The research presented in this paper was supported in part by DFG grant "Cross-language Learning-to-Rank for Patent Retrieval". We would like to thank Eugen Ruppert for his contribution to the ranking data construction.

References

Javed A. Aslam and Mark Montague. 2001. Models for metasearch. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, New Orleans, LA.

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. 2010. Learning to rank with (a lot of) word features. *Information Retrieval Journal*, 13(3):291–314.

Lisa Ballesteros and W. Bruce Croft. 1998. Resolving ambiguity for cross-language retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, Melbourne, Australia.

Nicholas J. Belkin, Paul Kantor, Edward A. Fox, and Joseph A. Shaw. 1995. Combining the evidence of multiple query representations for information retrieval. *Inf. Process. Manage.*, 31(3):431–448.

Leo Breiman. 1996. Bagging predictors. *Journal of Machine Learning Research*, 24:123–140.

Kevin Canini, Tushar Chandra, Eugene Ie, Jim McFadden, Ken Goldman, Mike Gunter, Jeremiah Harmsen, Kristen LeFevre, Dmitry Lepikhin, Tomas Lloret Llinares, Indraneel Mukherjee, Fernando Pereira, Josh Redstone, Tal Shaked, and Yoram Singer. 2010. Sibyl: A system for large scale machine learning. In *LADIS: The 4th ACM SIGOPS/SIGACT Workshop on Large Scale Distributed Systems and Middleware*, Zurich, Switzerland.

Xi Chen, Bing Bai, Yanjun Qi, Qihang Ling, and Jaime Carbonell. 2010. Learning preferences with millions of parameters by enforcing sparsity. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'10)*, Sydney, Australia.

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

Jeffrey Chin, Maureen Heymans, Alexandre Kojoukhov, Jocelyn Lin, and Hui Tan. 2008. Cross-language information retrieval. Patent Application. US 2008/0288474 A1.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Michael Collins, Robert E. Schapire, and Yoram Singer. 2002. Logistic regression, AdaBoost and Bregman distances. *Journal of Machine Learning Research*, 48(1-3):253–285.

Kareem Darwish and Douglas W. Oard. 2003. Probabilistic structured query methods. In *Proceedings. of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*, Toronto, Canada.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, Uppsala, Sweden.

Minwei Feng, Christoph Schmidt, Joern Wuebker, Stephan Peitz, Markus Freitag, and Hermann Ney. 2011. The RWTH Aachen system for NTCIR-9 PatentMT. In *Proceedings of the NTCIR-9 Workshop*, Tokyo, Japan.

Yoav Freund, Ray Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Proceedings of NTCIR-7 Workshop Meeting*, Tokyo, Japan.

Yasser Ganjisaffar, Rich Caruana, and Cristina Videira Lopes. 2011. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'11)*, Beijing, China.

Jianfeng Gao, Jian-Yun Nie, Endong Xun, Jian Zhang, Ming Zhou, and Changning Huang. 2001. Improving query translation for cross-language information retrieval using statistical models. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, New Orleans, LA.

Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Jian Hu, Kam-Fai Wong, and Hsiao-Wuen Hon. 2007. Cross-lingual query suggestion using query logs of different languages. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, Amsterdam, The Netherlands.

Sharad Goel, John Langford, and Alexander L. Strehl. 2008. Predictive indexing for fast search. In *Advances in Neural Information Processing Systems*, Vancouver, Canada.

Erik Graf and Leif Azzopardi. 2008. A methodology for building a patent test collection for prior art search. In *Proceedings of the 2nd International Workshop on Evaluating Information Access (EVIA)*, Tokyo, Japan.

David Grangier and Samy Bengio. 2008. A discriminative kernel-based approach to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1371–1384.

Yunsong Guo and Carla Gomes. 2009. Ranking structured documents: A large margin based approach for

patent prior art search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'09)*, Pasadena, CA.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation (WMT'11)*, Edinburgh, UK.

Gareth J.F. Jones and Adenike M. Lam-Adesina. 2002. Combination methods for improving the reliability of machine translation based cross-language information retrieval. In *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS'02)*, Limerick, Ireland.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL'03)*, Budapest, Hungary.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'02)*, Tampere, Finland.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic.

Walid Magdy and Gareth J.F. Jones. 2010. PRES: a score metric for evaluating recall-oriented information retrieval applications. In *Proceedings of the ACM SIGIR conference on Research and development in information retrieval (SIGIR'10)*, New York, NY.

Walid Magdy and Gareth J. F. Jones. 2011. An efficient method for using machine translation technologies in cross-language patent search. In *Proceedings of the 20th ACM Conference on Informationand Knowledge Management (CIKM'11)*, Glasgow, Scotland, UK.

Vassilina Nikoulina, Bogomil Kovachev, Nikolaos Lagos, and Christof Monz. 2012. Adaptation of statistical machine translation model for cross-lingual information retrieval in a service context. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL'12)*, Avignon, France.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley, New York.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Meeting of the Association for Computational Linguistics (ACL'00)*, Hongkong, China.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Meeting on Association for Computational Linguistics (ACL'03)*, Sapporo, Japan.

Dmitry Pavlov, Alexey Gorodilov, and Cliff A. Brunk. 2010. Bagboo: a scalable hybrid bagging-the-boosting model. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*, Toronto, Canada.

Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. 1998. Okapi at TREC-7. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD.

Saharon Rosset, Ji Zhu, and Trevor Hastie. 2004. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973.

Robert E. Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Journal of Machine Learning Research*, 37(3):297–336.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alexander J. Smola, Alexander L. Strehl, and Vishy Vishwanathan. 2009. Hash Kernels. In *Proceedings of the 12th Int. Conference on Artificial Intelligence and Statistics (AISTATS'09)*, Irvine, CA.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management (CIKM '07)*, New York, NY.

Ferhan Ture, Jimmy Lin, and Douglas W. Oard. 2012a. Combining statistical translation techniques for cross-language information retrieval. In *Proceedings of the International Conference on Computational Linguistics (COLING 2012)*, Bombay, India.

Ferhan Ture, Jimmy Lin, and Douglas W. Oard. 2012b. Looking inside the box: Context-sensitive translation for cross-language information retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012)*, Portland, OR.

Masao Utiyama and Hitoshi Isahara. 2007. A Japanese-English patent parallel corpus. In *Proceedings of MT Summit XI*, Copenhagen, Denmark.

Jinxi Xu, Ralph Weischedel, and Chanh Nguyen. 2001. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, New York, NY.

Recurrent Continuous Translation Models

Nal Kalchbrenner **Phil Blunsom**
Department of Computer Science
University of Oxford
{nal.kalchbrenner,phil.blunsom}@cs.ox.ac.uk

Abstract

We introduce a class of probabilistic continuous translation models called Recurrent Continuous Translation Models that are purely based on continuous representations for words, phrases and sentences and do not rely on alignments or phrasal translation units. The models have a generation and a conditioning aspect. The generation of the translation is modelled with a target Recurrent Language Model, whereas the conditioning on the source sentence is modelled with a Convolutional Sentence Model. Through various experiments, we show first that our models obtain a perplexity with respect to gold translations that is $> 43\%$ lower than that of state-of-the-art alignment-based translation models. Secondly, we show that they are remarkably sensitive to the word order, syntax, and meaning of the source sentence despite lacking alignments. Finally we show that they match a state-of-the-art system when rescoring n-best lists of translations.

1 Introduction

In most statistical approaches to machine translation the basic units of translation are phrases that are composed of one or more words. A crucial component of translation systems are models that estimate translation probabilities for pairs of phrases, one phrase being from the source language and the other from the target language. Such models count phrase pairs and their occurrences as distinct if the surface forms of the phrases are distinct. Although distinct phrase pairs often share significant similarities, linguistic or otherwise, they do not share statistical weight in the models' estimation of their translation probabilities. Besides ignoring the similarity of phrase pairs, this leads to general sparsity issues. The estimation is sparse or skewed for the large number of rare or unseen phrase pairs, which grows exponentially in the length of the phrases, and the generalisation to other domains is often limited.

Continuous representations have shown promise at tackling these issues. Continuous representations for words are able to capture their morphological, syntactic and semantic similarity (Collobert and Weston, 2008). They have been applied in continuous language models demonstrating the ability to overcome sparsity issues and to achieve state-of-the-art performance (Bengio et al., 2003; Mikolov et al., 2010). Word representations have also shown a marked sensitivity to conditioning information (Mikolov and Zweig, 2012). Continuous representations for characters have been deployed in character-level language models demonstrating notable language generation capabilities (Sutskever et al., 2011). Continuous representations have also been constructed for phrases and sentences. The representations are able to carry similarity and task dependent information, e.g. sentiment, paraphrase or dialogue labels, significantly beyond the word level and to accurately predict labels for a highly diverse range of unseen phrases and sentences (Grefenstette et al., 2011; Socher et al., 2011; Socher et al., 2012; Hermann and Blunsom, 2013; Kalchbrenner and Blunsom, 2013).

Phrase-based continuous translation models were first proposed in (Schwenk et al., 2006) and re-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1700–1709,
Seattle, Washington, USA, 18-21 October 2013.

cently further developed in (Schwenk, 2012; Le et al., 2012). The models incorporate a principled way of estimating translation probabilities that robustly extends to rare and unseen phrases. They achieve significant Bleu score improvements and yield semantically more suggestive translations. Although wide-reaching in their scope, these models are limited to fixed-size source and target phrases and simplify the dependencies between the target words taking into account restricted target language modelling information.

We describe a class of continuous translation models called Recurrent Continuous Translation Models (RCTM) that map without loss of generality a sentence from the source language to a probability distribution over the sentences in the target language. We define two specific RCTM architectures. Both models adopt a recurrent language model for the generation of the target translation (Mikolov et al., 2010). In contrast to other n-gram approaches, the recurrent language model makes no Markov assumptions about the dependencies of the words in the target sentence.

The two RCTMs differ in the way they condition the target language model on the source sentence. The first RCTM uses the convolutional sentence model (Kalchbrenner and Blunsom, 2013) to transform the source word representations into a representation for the source sentence. The source sentence representation in turn constraints the generation of each target word. The second RCTM introduces an intermediate representation. It uses a truncated variant of the convolutional sentence model to first transform the source word representations into representations for the target words; the latter then constrain the generation of the target sentence. In both cases, the convolutional layers are used to generate combined representations for the phrases in a sentence from the representations of the words in the sentence.

An advantage of RCTMs is the lack of latent alignment segmentations and the sparsity associated with them. Connections between source and target words, phrases and sentences are learnt only implicitly as mappings between their continuous representations. As we see in Sect. 5, these mappings often carry remarkably precise morphological, syntactic and semantic information. Another advantage is that the probability of a translation under the models is efficiently computable requiring a small number of matrix-vector products that is linear in the length of the source and the target sentence. Further, translations can be generated directly from the probability distribution of the RCTM without any external resources.

We evaluate the performance of the models in four experiments. Since the translation probabilities of the RCTMs are tractable, we can measure the perplexity of the models with respect to the reference translations. The perplexity of the models is significantly lower than that of IBM Model 1 and is $> 43\%$ lower than the perplexity of a state-of-the-art variant of the IBM Model 2 (Brown et al., 1993; Dyer et al., 2013). The second and third experiments aim to show the sensitivity of the output of the RCTM II to the linguistic information in the source sentence. The second experiment shows that under a random permutation of the words in the source sentences, the perplexity of the model with respect to the reference translations becomes significantly worse, suggesting that the model is highly sensitive to word position and order. The third experiment inspects the translations generated by the RCTM II. The generated translations demonstrate remarkable morphological, syntactic and semantic agreement with the source sentence. Finally, we test the RCTMs on the task of rescoring n-best lists of translations. The performance of the RCTM probabilities joined with a single word penalty feature matches the performance of the state-of-the-art translation system `cdec` that makes use of twelve features including five alignment-based translation models (Dyer et al., 2010).

We proceed as follows. We begin in Sect. 2 by describing the general modelling framework underlying the RCTMs. In Sect. 3 we describe the RCTM I and in Sect. 4 the RCTM II. Section 5 is dedicated to the four experiments and we conclude in Sect. 6.[1]

2 Framework

We begin by describing the modelling framework underlying RCTMs. An RCTM estimates the probability $P(\mathsf{f}|\mathsf{e})$ of a target sentence $\mathsf{f} = \mathsf{f}_1, ..., \mathsf{f}_m$ being a translation of a source sentence $\mathsf{e} = \mathsf{e}_1, ..., \mathsf{e}_k$. Let

[1] Code and models available at `nal.co`

us denote by $\mathsf{f}_{i:j}$ the substring of words $\mathsf{f}_i, ..., \mathsf{f}_j$. Using the following identity,

$$P(\mathsf{f}|\mathsf{e}) = \prod_{i=1}^{m} P(\mathsf{f}_i|\mathsf{f}_{1:i-1}, \mathsf{e}) \quad (1)$$

an RCTM estimates $P(\mathsf{f}|\mathsf{e})$ by directly computing for each target position i the conditional probability $P(\mathsf{f}_i|\mathsf{f}_{1:i-1}, \mathsf{e})$ of the target word f_i occurring in the translation at position i, given the preceding target words $\mathsf{f}_{1:i-1}$ and the source sentence e. We see that an RCTM is sensitive not just to the source sentence e but also to the preceding words $\mathsf{f}_{1:i-1}$ in the target sentence; by doing so it incorporates a model of the target language itself.

To model the conditional probability $P(\mathsf{f}|\mathsf{e})$, an RCTM comprises both a generative architecture for the target sentence and an architecture for conditioning the latter on the source sentence. To fully capture Eq. 1, we model the generative architecture with a recurrent language model (RLM) based on a recurrent neural network (Mikolov et al., 2010). The prediction of the i-th word f_i in a RLM depends on all the preceding words $\mathsf{f}_{1:i-1}$ in the target sentence ensuring that conditional independence assumptions are not introduced in Eq. 1. Although the prediction is most strongly influenced by words closely preceding f_i, long-range dependencies from across the whole sentence can also be exhibited. The conditioning architectures are model specific and are treated in Sect. 3-4. Both the generative and conditioning aspects of the models deploy continuous representations for the constituents and are trained as a single joint architecture. Given the modelling framework underlying RCTMs, we now proceed to describe in detail the recurrent language model underlying the generative aspect.

2.1 Recurrent Language Model

A RLM models the probability $P(\mathsf{f})$ that the sequence of words f occurs in a given language. Let $\mathsf{f} = \mathsf{f}_1, ..., \mathsf{f}_m$ be a sequence of m words, e.g. a sentence in the target language. Analogously to Eq. 1, using the identity,

$$P(\mathsf{f}) = \prod_{i=1}^{m} P(\mathsf{f}_i|\mathsf{f}_{1:i-1}) \quad (2)$$

the model explicitly computes without simplifying assumptions the conditional distributions

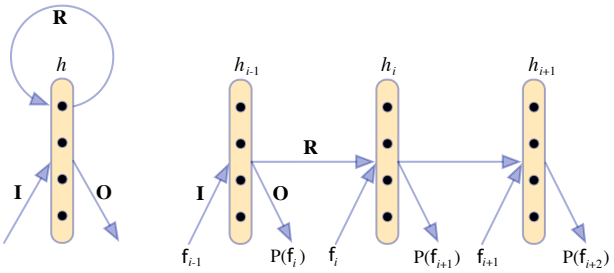


Figure 1: A RLM (left) and its unravelling to depth 3 (right). The recurrent transformation is applied to the hidden layer h_{i-1} and the result is summed to the representation for the current word f_i. After a non-linear transformation, a probability distribution over the next word f_{i+1} is predicted.

$P(\mathsf{f}_i|\mathsf{f}_{1:i-1})$. The architecture of a RLM comprises a vocabulary V that contains the words f_i of the language as well as three transformations: an input vocabulary transformation $\mathbf{I} \in \mathbb{R}^{q \times |V|}$, a recurrent transformation $\mathbf{R} \in \mathbb{R}^{q \times q}$ and an output vocabulary transformation $\mathbf{O} \in \mathbb{R}^{|V| \times q}$. For each word $\mathsf{f}_k \in V$, we indicate by $\mathsf{i}(\mathsf{f}_k)$ its index in V and by $\mathsf{v}(\mathsf{f}_k) \in \mathbb{R}^{|V| \times 1}$ an all zero vector with only $\mathsf{v}(\mathsf{f}_k)_{\mathsf{i}(\mathsf{f}_k)} = 1$.

For a word f_i, the result of $\mathbf{I} \cdot \mathsf{v}(\mathsf{f}_i) \in \mathbb{R}^{q \times 1}$ is the input continuous representation of f_i. The parameter q governs the size of the word representation. The prediction proceeds by successively applying the recurrent transformation $\mathbf{R}$ to the word representations and predicting the next word at each step. In detail, the computation of each $P(\mathsf{f}_i|\mathsf{f}_{1:i-1})$ proceeds recursively. For $1 < i < m$,

$$h_1 = \sigma(\mathbf{I} \cdot \mathsf{v}(\mathsf{f}_1)) \quad (3a)$$
$$h_{i+1} = \sigma(\mathbf{R} \cdot h_i + \mathbf{I} \cdot \mathsf{v}(\mathsf{f}_{i+1})) \quad (3b)$$
$$o_{i+1} = \mathbf{O} \cdot h_i \quad (3c)$$

and the conditional distribution is given by,

$$P(\mathsf{f}_i = v|\mathsf{f}_{1:i-1}) = \frac{\exp(o_{i,v})}{\sum_{v=1}^{V} \exp(o_{i,v})} \quad (4)$$

In Eq. 3, σ is a nonlinear function such as $\tanh$. Bias values b_h and b_o are included in the computation. An illustration of the RLM is given in Fig. 1.

The RLM is trained by backpropagation through time (Mikolov et al., 2010). The error in the predicted distribution calculated at the output layer is

backpropagated through the recurrent layers and cumulatively added to the errors of the previous predictions for a given number d of steps. The procedure is equivalent to standard backpropagation over a RLM that is unravelled to depth d as in Fig. 1.

RCTMs may be thought of as RLMs, in which the predicted distributions for each word f_i are conditioned on the source sentence e. We next define two conditioning architectures each giving rise to a specific RCTM.

3 Recurrent Continuous Translation Model I

The RCTM I uses a convolutional sentence model (CSM) in the conditioning architecture. The CSM creates a representation for a sentence that is progressively built up from representations of the n-grams in the sentence. The CSM embodies a hierarchical structure. Although it does not make use of an explicit parse tree, the operations that generate the representations act locally on small n-grams in the lower layers of the model and act increasingly more globally on the whole sentence in the upper layers of the model. The lack of the need for a parse tree yields two central advantages over sentence models that require it (Grefenstette et al., 2011; Socher et al., 2012). First, it makes the model robustly applicable to a large number of languages for which accurate parsers are not available. Secondly, the translation probability distribution over the target sentences does not depend on the chosen parse tree.

The RCTM I conditions the probability of each target word f_i on the continuous representation of the source sentence e generated through the CSM. This is accomplished by adding the sentence representation to each hidden layer h_i in the target recurrent language model. We next describe the procedure in more detail, starting with the CSM itself.

3.1 Convolutional Sentence Model

The CSM models the continuous representation of a sentence based on the continuous representations of the words in the sentence. Let $\mathsf{e} = \mathsf{e}_1 ... \mathsf{e}_k$ be a sentence in a language and let $\mathsf{v}(\mathsf{e}_i) \in \mathbb{R}^{q \times 1}$ be the continuous representation of the word e_i. Let $\mathbf{E}^{\mathsf{e}} \in \mathbb{R}^{q \times k}$ be the *sentence matrix* for e defined by,

$$\mathbf{E}^{\mathsf{e}}_{:,i} = \mathsf{v}(\mathsf{e}_i) \quad (5)$$

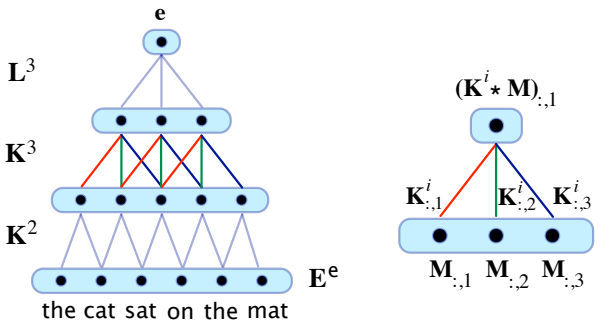


Figure 2: A CSM for a six word source sentence e and the computed sentence representation $\mathbf{e}$. $\mathbf{K}^2, \mathbf{K}^3$ are weight matrices and $\mathbf{L}^3$ is a top weight matrix. To the right, an instance of a one-dimensional convolution between some weight matrix $\mathbf{K}^i$ and a generic matrix $\mathbf{M}$ that could for instance correspond to $\mathbf{E}^{\mathsf{e}}_2$. The color coding of weights indicates weight sharing.

The main component of the architecture of the CSM is a sequence of *weight* matrices $(\mathbf{K}^i)_{2 \leq i \leq r}$ that correspond to the kernels or filters of the convolution and can be thought of as learnt feature detectors. From the sentence matrix $\mathbf{E}^{\mathsf{e}}$ the CSM computes a continuous vector representation $\mathbf{e} \in \mathbb{R}^{q \times 1}$ for the sentence e by applying a sequence of convolutions to $\mathbf{E}^{\mathsf{e}}$ whose weights are given by the weight matrices. The weight matrices and the sequence of convolutions are defined next.

We denote by $(\mathbf{K}^i)_{2 \leq i \leq r}$ a sequence of weight matrices where each $\mathbf{K}^i \in \mathbb{R}^{q \times i}$ is a matrix of i columns and $r = \lceil \sqrt{2N} \rceil$, where N is the length of the longest source sentence in the training set. Each row of $\mathbf{K}^i$ is a vector of i weights that is treated as the kernel or filter of a *one-dimensional* convolution. Given for instance a matrix $\mathbf{M} \in \mathbb{R}^{q \times j}$ where the number of columns $j \geq i$, each row of $\mathbf{K}^i$ can be convolved with the corresponding row in $\mathbf{M}$, resulting in a matrix $\mathbf{K}^i * \mathbf{M}$, where $*$ indicates the convolution operation and $(\mathbf{K}^i * \mathbf{M}) \in \mathbb{R}^{q \times (j-i+1)}$. For $i = 3$, the value $(\mathbf{K}^i * \mathbf{M})_{:,a}$ is computed by:

$$\mathbf{K}^i_{:,1} \odot \mathbf{M}_{:,a} + \mathbf{K}^i_{:,2} \odot \mathbf{M}_{:,a+1} + \mathbf{K}^i_{:,3} \odot \mathbf{M}_{:,a+2} \quad (6)$$

where $\odot$ is component-wise vector product. Applying the convolution kernel $\mathbf{K}^i$ yields a matrix $(\mathbf{K}^i * \mathbf{M})$ that has $i-1$ columns less than the original matrix $\mathbf{M}$.

Given a source sentence of length k, the CSM convolves successively with the sentence matrix $\mathbf{E}^{\mathsf{e}}$

the sequence of weight matrices $(\mathbf{K}^i)_{2\leq i\leq r}$, one after the other starting with $\mathbf{K}^2$ as follows:

$$\mathbf{E}^{\mathsf{e}}_1 = \mathbf{E}^{\mathsf{e}} \tag{7a}$$

$$\mathbf{E}^{\mathsf{e}}_{i+1} = \sigma(\mathbf{K}^{i+1} * \mathbf{E}^{\mathsf{e}}_i) \tag{7b}$$

After a few convolution operations, $\mathbf{E}^{\mathsf{e}}_i$ is either a vector in $\mathbb{R}^{q\times 1}$, in which case we obtained the desired representation, or the number of columns in $\mathbf{E}^{\mathsf{e}}_i$ is smaller than the number $i+1$ of columns in the next weight matrix $\mathbf{K}^{i+1}$. In the latter case, we equally obtain a vector in $\mathbb{R}^{q\times 1}$ by simply applying a top weight matrix $\mathbf{L}^j$ that has the same number of columns as $\mathbf{E}^{\mathsf{e}}_i$. We thus obtain a sentence representation $\mathbf{e} \in \mathbb{R}^{q\times 1}$ for the source sentence e. Note that the convolution operations in Eq. 7b are interleaved with non-linear functions σ. Note also that, given the different levels at which the weight matrices $\mathbf{K}^i$ and $\mathbf{L}^i$ are applied, the top weight matrix $\mathbf{L}^j$ comes from an additional sequence of weight matrices $(\mathbf{L}^i)_{2\leq i\leq r}$ distinct from $(\mathbf{K}^i)_{2\leq i\leq r}$. Fig. 2 depicts an instance of the CSM and of a one-dimensional convolution.[2]

3.2 RCTM I

As defined in Sect. 2, the RCTM I models the conditional probability $P(\mathsf{f}|\mathsf{e})$ of a sentence $\mathsf{f} = \mathsf{f}_1, ..., \mathsf{f}_m$ in a target language F being the translation of a sentence $\mathsf{e} = \mathsf{e}_1, ..., \mathsf{e}_k$ in a source language E. According to Eq. 1, the RCTM I explicitly computes the conditional distributions $P(\mathsf{f}_i|\mathsf{f}_{1:i-1}, \mathsf{e})$. The architecture of the RCTM I comprises a source vocabulary V^{E} and a target vocabulary V^{F}, two sequences of weight matrices $(\mathbf{K}^i)_{2\leq i\leq r}$ and $(\mathbf{L}^i)_{2\leq i\leq r}$ that are part of the constituent CSM, transformations $\mathbf{I} \in \mathbb{R}^{q\times|V^{\mathsf{F}}|}$, $\mathbf{R} \in \mathbb{R}^{q\times q}$ and $\mathbf{O} \in \mathbb{R}^{|V^{\mathsf{F}}|\times q}$ that are part of the constituent RLM and a sentence transformation $\mathbf{S} \in \mathbb{R}^{q\times q}$. We write $\mathbf{e} = \mathsf{csm}(\mathsf{e})$ for the output of the CSM with e as the input sentence.

The computation of the RCTM I is a simple modification to the computation of the RLM described in Eq. 3. It proceeds recursively as follows:

$$\mathbf{s} = \mathbf{S} \cdot \mathsf{csm}(\mathsf{e}) \tag{8a}$$

$$h_1 = \sigma(\mathbf{I} \cdot \mathsf{v}(\mathsf{f}_1) + \mathbf{s}) \tag{8b}$$

$$h_{i+1} = \sigma(\mathbf{R} \cdot h_i + \mathbf{I} \cdot \mathsf{v}(\mathsf{f}_{i+1}) + \mathbf{s}) \tag{8c}$$

$$o_{i+1} = \mathbf{O} \cdot h_i \tag{8d}$$

[2]For a formal treatment of the construction, see (Kalchbrenner and Blunsom, 2013).

and the conditional distributions $P(\mathsf{f}_{i+1}|\mathsf{f}_{1:i}, \mathsf{e})$ are obtained from o_i as in Eq. 4. σ is a nonlinear function and bias values are included throughout the computation. Fig. 3 illustrates an RCTM I.

Two aspects of the RCTM I are to be remarked. First, the length of the target sentence is predicted by the target RLM itself that by its architecture has a bias towards shorter sentences. Secondly, the representation of the source sentence e constraints uniformly all the target words, contrary to the fact that the target words depend more strongly on certain parts of the source sentence and less on other parts. The next model proposes an alternative formulation of these aspects.

4 Recurrent Continuous Translation Model II

The central idea behind the RCTM II is to first estimate the length m of the target sentence independently of the main architecture. Given m and the source sentence e, the model constructs a representation for the n-grams in e, where n is set to 4. Note that each level of the CSM yields n-gram representations of e for a specific value of n. The 4-gram representation of e is thus constructed by truncating the CSM at the level that corresponds to $n = 4$. The procedure is then inverted. From the 4-gram representation of the source sentence e, the model builds a representation of a sentence that has the predicted length m of the target. This is similarly accomplished by truncating the *inverted* CSM for a sentence of length m.

We next describe in detail the Convolutional n-gram Model (CGM). Then we return to specify the RCTM II.

4.1 Convolutional n-gram model

The CGM is obtained by truncating the CSM at the level where n-grams are represented for the chosen value of n. A column $\mathbf{g}$ of a matrix $\mathbf{E}^{\mathsf{e}}_i$ obtained according to Eq. 7 represents an n-gram from the source sentence e. The value of n corresponds to the number of word vectors from which the n-gram representation $\mathbf{g}$ is constructed; equivalently, n is the span of the weights in the CSM underneath $\mathbf{g}$ (see Fig. 2-3). Note that any column in a matrix $\mathbf{E}^{\mathsf{e}}_i$ represents an n-gram with the same span value n. We denote by $\mathsf{gram}(\mathbf{E}^{\mathsf{e}}_i)$ the size of the n-grams

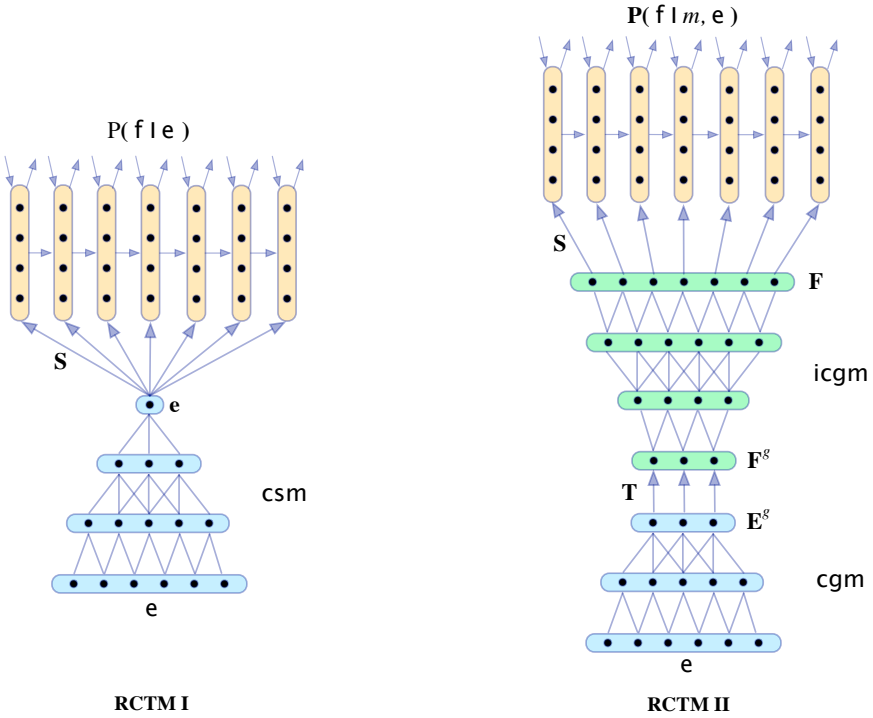


Figure 3: A graphical depiction of the two RCTMs. Arrows represent full matrix transformations while lines are vector transformations corresponding to columns of weight matrices.

represented by $\mathbf{E}^{\mathsf{e}}_i$. For example, for a sufficiently long sentence e, $\mathsf{gram}(\mathbf{E}^{\mathsf{e}}_2) = 2$, $\mathsf{gram}(\mathbf{E}^{\mathsf{e}}_3) = 4$, $\mathsf{gram}(\mathbf{E}^{\mathsf{e}}_4) = 7$. We denote by $\mathsf{cgm}(\mathsf{e}, n)$ that matrix $\mathbf{E}^{\mathsf{e}}_i$ from the CSM that represents the n-grams of the source sentence e.

The CGM can also be inverted to obtain a representation for a sentence from the representation of its n-grams. We denote by icgm the inverse CGM, which depends on the size of the n-gram representation $\mathsf{cgm}(\mathsf{e}, n)$ and on the target sentence length m. The transformation icgm unfolds the n-gram representation onto a representation of a target sentence with m words. The architecture corresponds to an inverted CGM or, equivalently, to an inverted truncated CSM (Fig. 3). Given the transformations cgm and icgm, we now detail the computation of the RCTM II.

4.2 RCTM II

The RCTM II models the conditional probability $P(\mathsf{f}|\mathsf{e})$ by factoring it as follows:

$$P(\mathsf{f}|\mathsf{e}) = P(\mathsf{f}|m, \mathsf{e}) \cdot P(m|\mathsf{e}) \tag{9a}$$

$$= \prod_{i=1}^{m} P(\mathsf{f}_{i+1}|\mathsf{f}_{1:i}, m, \mathsf{e}) \cdot P(m|\mathsf{e}) \tag{9b}$$

and computing the distributions $P(\mathsf{f}_{i+1}|\mathsf{f}_{1:i}, m, \mathsf{e})$ and $P(m|\mathsf{e})$. The architecture of the RCTM II comprises all the elements of the RCTM I together with the following additional elements: a translation transformation $\mathbf{T}^{q\times q}$ and two sequences of weight matrices $(\mathbf{J}^i)_{2\le i\le s}$ and $(\mathbf{H}^i)_{2\le i\le s}$ that are part of the icgm[3].

The computation of the RCTM II proceeds recursively as follows:

$$\mathbf{E}^g = \mathsf{cgm}(\mathsf{e}, 4) \tag{10a}$$

$$\mathbf{F}^g_{:,j} = \sigma(\mathbf{T} \cdot \mathbf{E}^g_{:,j}) \tag{10b}$$

$$\mathbf{F} = \mathsf{icgm}(\mathbf{F}^g, m) \tag{10c}$$

$$h_1 = \sigma(\mathbf{I} \cdot \mathsf{v}(\mathsf{f}_1) + \mathbf{S} \cdot \mathbf{F}_{:,1}) \tag{10d}$$

$$h_{i+1} = \sigma(\mathbf{R} \cdot h_i + \mathbf{I} \cdot \mathsf{v}(\mathsf{f}_{i+1}) + \mathbf{S} \cdot \mathbf{F}_{:,i+1}) \tag{10e}$$

$$o_{i+1} = \mathbf{O} \cdot h_i \tag{10f}$$

and the conditional distributions $P(\mathsf{f}_{i+1}|\mathsf{f}_{1:i}, \mathsf{e})$ are obtained from o_i as in Eq. 4. Note how each reconstructed vector $\mathbf{F}_{:,i}$ is added successively to the corresponding layer h_i that predicts the target word f_i. The RCTM II is illustrated in Fig. 3.

[3]Just like r the value s is small and depends on the length of the source and target sentences in the training set. See Sect. 5.1.2.

For the separate estimation of the length of the translation, we estimate the conditional probability $P(m|\mathsf{e})$ by letting,

$$P(m|\mathsf{e}) = P(m|k) = \mathsf{Poisson}(\lambda_k) \qquad (11)$$

where k is the length of the source sentence e and $\mathsf{Poisson}(\lambda)$ is a Poisson distribution with mean λ. This concludes the description of the RCTM II. We now turn to the experiments.

5 Experiments

We report on four experiments. The first experiment considers the perplexities of the models with respect to reference translations. The second and third experiments test the sensitivity of the RCTM II to the linguistic aspects of the source sentences. The final experiment tests the rescoring performance of the two models.

5.1 Training

Before turning to the experiments, we describe the data sets, hyper parameters and optimisation algorithms used for the training of the RCTMs.

5.1.1 Data sets

The training set used for all the experiments comprises a bilingual corpus of 144953 pairs of sentences less than 80 words in length from the news commentary section of the Eighth Workshop on Machine Translation (WMT) 2013 training data. The source language is English and the target language is French. The English sentences contain about 4.1M words and the French ones about 4.5M words. Words in both the English and French sentences that occur twice or less are substituted with the $\langle unknown \rangle$ token. The resulting vocabularies V^{E} and V^{F} contain, respectively, 25403 English words and 34831 French words.

For the experiments we use four different test sets comprised of the Workshop on Machine Translation News Test (WMT-NT) sets for the years 2009, 2010, 2011 and 2012. They contain, respectively, 2525, 2489, 3003 and 3003 pairs of English-French sentences. For the perplexity experiments unknown words occurring in these data sets are replaced with the $\langle unknown \rangle$ token. The respective 2008 WMT-NT set containing 2051 pairs of English-French sentences is used as the validation set throughout.

5.1.2 Model hyperparameters

The parameter q that defines the size of the English vectors $\mathsf{v}(\mathsf{e}_i)$ for $\mathsf{e}_i \in V^{\mathsf{E}}$, the size of the hidden layer h_i and the size of the French vectors $\mathsf{v}(\mathsf{f}_i)$ for $\mathsf{v}(\mathsf{f}_i) \in V^{\mathsf{F}}$ is set to $q = 256$. This yields a relatively small recurrent matrix and corresponding models. To speed up training, we factorize the target vocabulary V^{F} into 256 classes following the procedure in (Mikolov et al., 2011).

The RCTM II uses a convolutional n-gram model CGM where n is set to 4. For the RCTM I, the number of weight matrices r for the CSM is 15, whereas in the RCTM II the number r of weight matrices for the CGM is 7 and the number s of weight matrices for the inverse CGM is 9. If a test sentence is longer than all training sentences and a larger weight matrix is required by the model, the larger weight matrix is easily factorized into two smaller weight matrices whose weights have been trained. For instance, if a weight matrix of 10 weights is required, but weight matrices have been trained only up to weight 9, then one can factorize the matrix of 10 weights with one of 9 and one of 2. Across all test sets the proportion of sentence pairs that require larger weight matrices to be factorized into smaller ones is $< 0.1\%$.

5.1.3 Objective and optimisation

The objective function is the average of the sum of the cross-entropy errors of the predicted words and the true words in the French sentences. The English sentences are taken as input in the prediction of the French sentences, but they are not themselves ever predicted. An l_2 regularisation term is added to the objective. The training of the model proceeds by back-propagation through time. The cross-entropy error calculated at the output layer at each step is back-propagated through the recurrent structure for a number d of steps; for all models we let $d = 6$. The error accumulated at the hidden layers is then further back-propagated through the transformation $\mathbf{S}$ and the CSM/CGM to the input vectors $\mathsf{v}(\mathsf{e}_i)$ of the English input sentence e. All weights, including the English vectors, are randomly initialised and inferred during training.

The objective is minimised using mini-batch adaptive gradient descent (Adagrad) (Duchi et al., 2011). The training of an RCTM takes about 15 hours on 3 multicore CPUs. While our experiments

WMT-NT	*2009*	*2010*	*2011*	*2012*
KN-5	218	213	222	225
RLM	178	169	178	181
IBM 1	207	200	188	197
FA-IBM 2	153	146	135	144
RCTM I	143	134	140	142
RCTM II	**86**	**77**	**76**	**77**

Table 1: Perplexity results on the WMT-NT sets.

are relatively small, we note that in principle our models should scale similarly to RLMs which have been applied to hundreds of millions of words.

5.2 Perplexity of gold translations

Since the computation of the probability of a translation under one of the RCTMs is efficient, we can compute the perplexities of the RCTMs with respect to the reference translations in the test sets. The perplexity measure is an indication of the quality that a model assigns to a translation. We compare the perplexities of the RCTMs with the perplexity of the IBM Model 1 (Brown et al., 1993) and of the Fast-Aligner (FA-IBM 2) model that is a state-of-the-art variant of IBM Model 2 (Dyer et al., 2013). We add as baselines the unconditional target RLM and a 5-gram target language model with modified Kneser-Nay smoothing (KN-5). The results are reported in Tab. 1. The RCTM II obtains a perplexity that is $> 43\%$ lower than that of the alignment based models and that is 40% lower than the perplexity of the RCTM I. The low perplexity of the RCTMs suggests that continuous representations and the transformations between them make up well for the lack of explicit alignments. Further, the difference in perplexity between the RCTMs themselves demonstrates the importance of the conditioning architecture and suggests that the localised 4-gram conditioning in the RCTM II is superior to the conditioning with the whole source sentence of the RCTM I.

5.3 Sensitivity to source sentence structure

The second experiment aims at showing the sensitivity of the RCTM II to the order and position of words in the English source sentence. To this end, we randomly permute in the training and testing sets

WMT-NT PERM	*2009*	*2010*	*2011*	*2012*
RCTM II	174	168	175	178

Table 2: Perplexity results of the RCTM II on the WMT-NT sets where the words in the English source sentences are randomly permuted.

the words in the English source sentence. The results on the permuted data are reported in Tab. 2. If the RCTM II were roughly comparable to a bag-of-words approach, there would be no difference under the permutation of the words. By contrast, the difference of the results reported in Tab. 2 with those reported in Tab. 1 is very significant, clearly indicating the sensitivity to word order and position of the translation model.

5.3.1 Generating from the RCTM II

To show that the RCTM II is sensitive not only to word order, but also to other syntactic and semantic traits of the sentence, we generate and inspect candidate translations for various English source sentences. The generation proceeds by sampling from the probability distribution of the RCTM II itself and does not depend on any other external resources. Given an English source sentence e, we let m be the length of the gold translation and we search the distribution computed by the RCTM II over all sentences of length m. The number of possible target sentences of length m amounts to $|V|^m = 34831^m$ where $V = V^{\mathsf{F}}$ is the French vocabulary; directly considering all possible translations is intractable. We proceed as follows: we sample with replacement 2000 sentences from the distribution of the RCTM II, each obtained by predicting one word at a time. We start by predicting a distribution for the first target word, restricting that distribution to the top 5 most probable words and sampling the first word of a candidate translation from the restricted distribution of 5 words. We proceed similarly for the remaining words. Each sampled sentence has a well-defined probability assigned by the model and can thus be ranked. Table 3 gives various English source sentences and some candidate French translations generated by the RCTM II together with their ranks.

The results in Tab. 3 show the remarkable syntactic agreements of the candidate translations; the

English source sentence	French gold translation	RCTM II candidate translation	Rank
the patient is sick .	le patient est malade .	le patient est insuffisante .	1
		le patient est mort .	4
		la patient est insuffisante .	23
the patient is dead .	le patient est mort .	le patient est mort .	1
		le patient est dépassé .	4
the patient is ill .	le patient est malade .	le patient est mal .	3
the patients are sick .	les patients sont malades .	les patients sont confrontés .	2
		les patients sont corrompus .	5
the patients are dead .	les patients sont morts .	les patients sont morts .	1
the patients are ill .	les patients sont malades .	les patients sont confrontés .	5
the patient was ill .	le patient était malade .	le patient était mal .	2
the patients are not dead .	les patients ne sont pas morts .	les patients ne sont pas morts .	1
the patients are not sick .	les patients ne sont pas malades .	les patients ne sont pas $\langle unknown \rangle$.	1
		les patients ne sont pas mal .	6
the patients were saved .	les patients ont été sauvés .	les patients ont été sauvées .	6

Table 3: English source sentences, respective translations in French and candidate translations generated from the RCTM II and ranked out of 2000 samples according to their decreasing probability. Note that end of sentence dots $(.)$ are generated as part of the translation.

WMT-NT	*2009*	*2010*	*2011*	*2012*
RCTM I + WP	19.7	21.1	22.5	21.5
RCTM II + WP	19.8	21.1	22.5	21.7
`cdec` (12 features)	19.9	21.2	22.6	21.8

Table 4: Bleu scores on the WMT-NT sets of each RCTM linearly interpolated with a word penalty WP. The `cdec` system includes WP as well as five translation models and two language modelling features, among others.

large majority of the candidate translations are fully well-formed French sentences. Further, subtle syntactic features such as the singular or plural ending of nouns and the present and past tense of verbs are well correlated between the English source and the French candidate targets. Finally, the meaning of the English source is well transferred on the French candidate targets; where a correlation is unlikely or the target word is not in the French vocabulary, a semantically related word or synonym is selected by the model. All of these traits suggest that the RCTM II is able to capture a significant amount of both syntactic and semantic information from the English source sentence and successfully transfer it onto the French translation.

5.4 Rescoring and BLEU Evaluation

The fourth experiment tests the ability of the RCTM I and the RCTM II to choose the best translation among a large number of candidate translations produced by another system. We use the `cdec` system to generate a list of 1000 best candidate translations for each English sentence in the four WMT-NT sets. We compare the rescoring performance of the RCTM I and the RCTM II with that of the `cdec` itself. `cdec` employs 12 engineered features including, among others, 5 translation models, 2 language model features and a word penalty feature (WP). For the RCTMs we simply interpolate the log probability assigned by the models to the candidate translations with the word penalty feature WP, tuned on the validation data. The results of the experiment are reported in Tab. 4.

While there is little variance in the resulting Bleu scores, the performance of the RCTMs shows that their probabilities correlate with translation quality. Combining a monolingual RLM feature with the RCTMs does not improve the scores, while reducing `cdec` to just one core translation probability and language model features drops its score by two to five tenths. These results indicate that the RCTMs have been able to learn both translation and language modelling distributions.

6 Conclusion

We have introduced Recurrent Continuous Translation Models that comprise a class of purely continuous sentence-level translation models. We have shown the translation capabilities of these models and the low perplexities that they obtain with respect to reference translations. We have shown the ability of these models at capturing syntactic and semantic information and at estimating during reranking the quality of candidate translations.

The RCTMs offer great modelling flexibility due to the sensitivity of the continuous representations to conditioning information. The models also suggest a wide range of potential advantages and extensions, from being able to include discourse representations beyond the single sentence and multilingual source representations, to being able to model morphologically rich languages through character-level recurrences.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Peter F. Brown, Vincent J.Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proc. of NAACL*.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2011. Concrete sentence spaces for compositional distributional models of meaning. *CoRR*, abs/1101.0309.

Karl Moritz Hermann and Phil Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August. Association for Computational Linguistics. Forthcoming.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Convolutional Neural Networks for Discourse Compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *HLT-NAACL*, pages 39–48.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *SLT*, pages 234–239.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH*, pages 1045–1048. ISCA.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE.

Holger Schwenk, Daniel Déchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *ACL*.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *COLING (Posters)*, pages 1071–1080.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *ICML*, pages 1017–1024. Omnipress.

Learning Biological Processes with Global Constraints

Aju Thalappillil Scaria[*], **Jonathan Berant**[*], **Mengqiu Wang and Christopher D. Manning**
Stanford University, Stanford

Justin Lewis and Brittany Harding
University of Washington, Seattle

Peter Clark
Allen Institute for Artificial Intelligence, Seattle

Abstract

Biological processes are complex phenomena involving a series of events that are related to one another through various relationships. Systems that can understand and reason over biological processes would dramatically improve the performance of semantic applications involving inference such as question answering (QA) – specifically *"How?"* and *"Why?"* questions. In this paper, we present the task of *process extraction*, in which events within a process and the relations between the events are automatically extracted from text. We represent processes by graphs whose edges describe a set of temporal, causal and co-reference event-event relations, and characterize the structural properties of these graphs (e.g., the graphs are *connected*). Then, we present a method for extracting relations between the events, which exploits these structural properties by performing joint inference over the set of extracted relations. On a novel dataset containing 148 descriptions of biological processes (released with this paper), we show significant improvement comparing to baselines that disregard process structure.

1 Introduction

A *process* is defined as a series of inter-related events that involve multiple entities and lead to an end result. Product manufacturing, economical developments, and various phenomena in life and social sciences can all be viewed as types of processes. Processes are complicated objects; consider for example the biological process of ATP synthesis described in Figure 1. This process involves 12 entities and 8 events. Additionally, it describes relations between events and entities, and the relationship between events (e.g., the second occurrence of the event *'enter'*, causes the event *'changing'*).

Automatically extracting the structure of processes from text is crucial for applications that require reasoning, such as non-factoid QA. For instance, answering a question on ATP synthesis, such as *"How do H+ ions contribute to the production of ATP?"* requires a structure that links *H+ ions* (Figure 1, sentence 1) to *ATP* (Figure 1, sentence 4) through a sequence of intermediate events. Such *"How?"* questions are common on FAQ websites (Surdeanu et al., 2011), which further supports the importance of process extraction.

Process extraction is related to two recent lines of work in Information Extraction – event extraction and timeline construction. Traditional event extraction focuses on identifying a closed set of events within a single sentence. For example, the BioNLP 2009 and 2011 shared tasks (Kim et al., 2009; Kim et al., 2011) consider nine events types related to proteins. In practice, events are currently almost always extracted from a single sentence. Process extraction, on the other hand, is centered around discovering *relations* between events that span *multiple* sentences. The set of possible event types in process extraction is also much larger.

Timeline construction involves identifying temporal relations between events (Do et al., 2012; McClosky and Manning, 2012; D'Souza and Ng, 2013), and is thus related to process extraction as both focus on event-event relations spanning multiple sentences. However, events in processes are tightly coupled in ways that go beyond simple temporal ordering, and these dependencies are central for the process extraction task. Hence, capturing process structure requires modeling a larger set of relations that includes temporal, causal and co-reference relations.

In this paper, we formally define the task of process extraction and present automatic extraction methods. Our approach handles an open set of event types and works over multiple sentences, extracting a rich set of event-event relations. Furthermore,

[*] Both authors equally contributed to the paper

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1710–1720,
Seattle, Washington, USA, 18-21 October 2013.

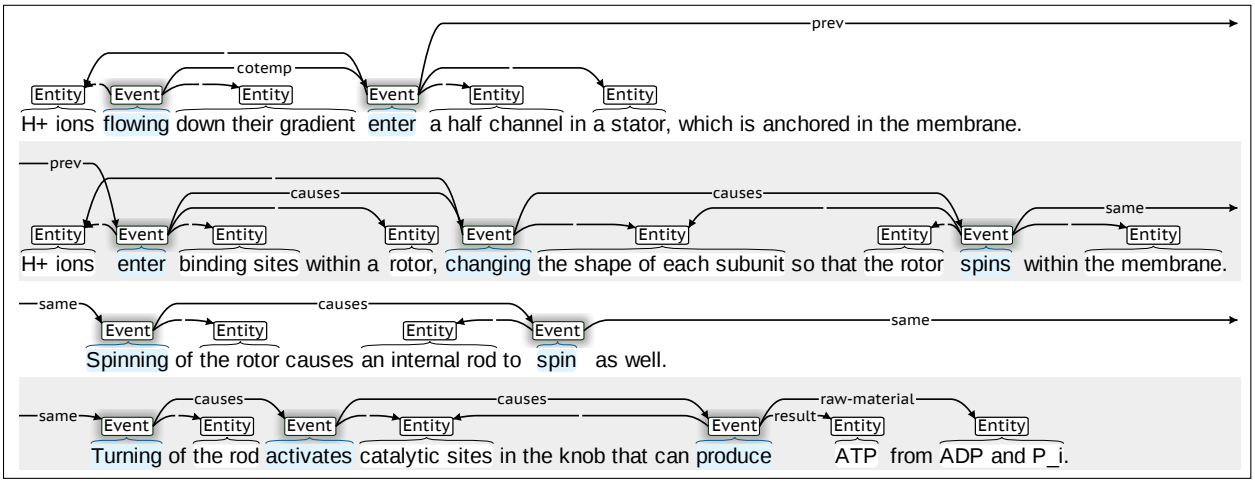


Figure 1: Partial annotation of the ATP synthesis process. Most of the semantic roles have been removed for simplicity.

we characterize a set of global properties of process structure that can be utilized during process extraction. For example, all events in a process are somehow connected to one another. Also, processes usually exhibit a "chain-like" structure reflecting process progression over time. We show that incorporating such global properties into our model and performing joint inference over the extracted relations significantly improves the quality of process structures predicted. We conduct experiments on a novel dataset of process descriptions from the textbook "Biology" (Campbell and Reece, 2005) that were annotated by trained biologists. Our method does not require any domain-specific knowledge and can be easily adapted to non-biology domains.

The main contributions of this paper are:

1. We define process extraction and characterize processes' structural properties.
2. We model global structural properties in processes and demonstrate significant improvement in extraction accuracy.
3. We publicly release a novel data set of 148 fully annotated biological process descriptions along with the source code for our system. The dataset and code can be downloaded from `http://nlp.stanford.edu/software/bioprocess/`.

2 Process Definition and Dataset

We define a process description as a paragraph or sequence of tokens $\mathbf{x} = \{x_1, \ldots x_{|\mathbf{x}|}\}$ that describes a series of events related by temporal and/or causal relations. For example, in ATP synthesis (Figure 1), the event of rotor spinning *causes* the event where an internal rod spins.

We model the events within a process and their relations by a directed graph $\mathcal{P} = (V, E)$, where the nodes $V = \{1, \ldots, |V|\}$ represent event mentions and labeled edges E correspond to event-event relations. An event mention $v \in V$ is defined by a trigger t_v, which is a span of words $x_i, x_{i+1}, \ldots, x_j$; and by a set of argument mentions A_v, where each argument mention $a_v \in A_v$ is also a span of words labeled by a semantic role l taken from a set $\mathcal{L}$. For example, in the last event mention of ATP synthesis, $t_v =$ *produce*, and one of the argument mentions is $a_v =$ *(ATP,* RESULT*)*. A labeled edge (u, v, r) in the graph describes a relation $r \in \mathcal{R}$ between the event mentions u and v. The task of process extraction is to extract the graph $\mathcal{P}$ from the text $\mathbf{x}$.[1]

A natural way to break down process extraction into sub-parts is to first perform semantic role labeling (SRL), that is, identify triggers and predict argument mentions with their semantic role, and then extract event-event relations between pairs of event mentions. In this paper, we focus on the second step, where given a set of event triggers $\mathcal{T}$, we find all event-event relations, where a trigger represents the entire event. For completeness, we now describe the semantic roles $\mathcal{L}$ used in our dataset, and then

[1] Argument mentions are also related by coreference relations, but we neglect that since it is not central in this paper.

present the set of event-event relations $\mathcal{R}$.

The set $\mathcal{L}$ contains standard semantic roles such as AGENT, THEME, ORIGIN, DESTINATION and LOCATION. Two additional semantic roles were employed that are relevant for biological text: RESULT corresponds to an entity that is the result of an event, and RAW-MATERIAL describes an entity that is used or consumed during an event. For example, the last event *'produce'* in Figure 1, has *'ATP'* as the RESULT, and *'ADP'* as the RAW-MATERIAL.

The event-event relation set $\mathcal{R}$ contains the following (assuming a labeled edge (u, v, r)):

1. PREV denotes that u is an event immediately before v. Thus, the edges (u, v, PREV) and (v, w, PREV), preclude the edge (u, w, PREV). For example, in "When a photon *strikes* ...energy is *passed* ...until it *reaches* ...", there is no edge (*strikes*, *reaches*, PREV) due to the intervening event *'passed'*.
2. COTEMP denotes that events u and v overlap in time (e.g., the first two event mentions *flowing* and *enter* in Figure 1).
3. SUPER denotes that event u includes event v. For instance, in "During *DNA replication*, DNA polymerases *proofread* each nucleotide..." there is an edge (*DNA replication*, *proofread*, SUPER).
4. CAUSES denotes that event u causes event v (e.g., the relation between *changing* and *spins* in sentence 2 of Figure 1).
5. ENABLES denotes that event u creates preconditions that allow event v to take place. For example, the description "...cause cancer cells to *lose* attachments to neighboring cells..., allowing them to *spread* into nearby tissues" has the edge (*lose*, *spread*, ENABLES). An intuitive way to think about the difference between *Causes* and *Enables* is the following: if u causes v this means that if u happens, then v happens. If u enables v, then if u does not happen, then v does not happen.
6. SAME denotes that u and v both refer to the same event (*spins* and *Spinning* in Figure 1).

Early work on temporal logic (Allen, 1983) contained more temporal relations than are used in our relation set $\mathcal{R}$. We chose a relation set $\mathcal{R}$ that captures the essential aspects of temporal relations between events in a process, while keeping the annotation as simple as possible. For instance, we include the SUPER relation that appears in temporal annotations such as the Timebank corpus (Pustejovsky et al., 2003) and Allen's work, but in practice was not considered by many temporal ordering systems (Chambers and Jurafsky, 2008; Yoshikawa et al., 2009; Do et al., 2012). Importantly, our relation set also includes the relations CAUSES and ENABLES, which are fundamental to modeling processes and go beyond simple temporal ordering.

	Avg	**Min**	**Max**
# of sentences	3.80	1	15
# of tokens	89.98	19	319
# of events	6.20	2	15
# of non-NONE relations	5.64	1	24

Table 1: Process statistics over 148 process descriptions. NONE is used to indicate no relation.

We also added event coreference (SAME) to $\mathcal{R}$. Do et al. (2012) used event coreference information in a temporal ordering task to modify probabilities provided by pairwise classifiers prior to joint inference. In this paper, we simply treat SAME as another event-event relation, which allows us to easily perform joint inference and employ structural constraints that combine both coreference and temporal relations simultaneously. For example, if u and v are the same event, then there can exist no w, such that u is before w, but v is after w (see Section 3.3)

We annotated 148 process descriptions based on the aforementioned definitions. Further details on annotation and data set statistics are provided in Section 4 and Table 1.

Structural properties of processes Coherent processes exhibit many structural properties. For example, two argument mentions related to the same event cannot overlap – a constraint that has been used in the past in SRL (Toutanova et al., 2008). In this paper we focus on three main structural properties of the graph $\mathcal{P}$. First, in a coherent process, all events mentioned are related to one another, and hence the graph $\mathcal{P}$ must be connected. Second, processes tend to have a "chain-like" structure where one event follows another, and thus we expect

Deg.	Gold	Local	Global
0	0	29	0
1	219	274	224
2	369	337	408
3	46	14	17
≥ 4	22	2	7

Table 2: Node degree distribution for event mentions on the training set. Predictions for the *Local* and *Global* models were obtained using 10-fold cross validation.

nodes' degree to generally be ≤ 2. Indeed, 90% of event mentions have degree ≤ 2, as demonstrated by the *Gold* column of Table 2. Last, if we consider relations between all possible triples of events in a process, clearly some configurations are impossible, while others are common (illustrated in Figure 2). In Section 3.3, we show that modeling these properties using a joint inference framework improves the quality of process extraction significantly.

3 Joint Model for Process Extraction

Given a paragraph $\mathbf{x}$ and a trigger set $\mathcal{T}$, we wish to extract all event-event relations E. Similar to Do et al. (2012), our model consists of a local pairwise classifier and global constraints. We first introduce a classifier that is based on features from previous work. Next, we describe novel features specific for process extraction. Last, we incorporate global constraints into our model using an ILP formulation.

3.1 Local pairwise classifier

The local pairwise classifier predicts relations between all event mention pairs. In order to model the direction of relations, we expand the set $\mathcal{R}$ to include the reverse of four directed relations: PREV-NEXT, SUPER- SUB, CAUSES-CAUSED, ENABLES-ENABLED. After adding NONE to indicate no relation, and including the undirected relations COTEMP and SAME, $\mathcal{R}$ contains 11 relations. The classifier is hence a function $f : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{R}$. As an example, $f(t_i, t_j) = \text{PREV}$ iff $f(t_j, t_i) = \text{NEXT}$. Let n be the number of triggers in a process, and t_i be the i-th trigger in its description. Since $f(t_i, t_j)$ completely determines $f(t_j, t_i)$, it suffices to consider only pairs with $i < j$. Note that the process graph $\mathcal{P}$ is undirected under the new definition of $\mathcal{R}$.

Table 3 describes features from previous work (Chambers and Jurafsky, 2008; Do et al., 2012) extracted for a trigger pair (t_i, t_j). Some features were omitted since they did not yield improvement in performance on a development set (e.g., lemmas and part-of-speech tags of context words surrounding t_i and t_j), or they require gold annotations provided in TimeBank, which we do not have (e.g., *tense* and *aspect* of triggers). To reduce sparseness, we convert nominalizations into their verbal forms when computing word lemmas, using WordNet's (Fellbaum, 1998) derivation links.

Feature	Description
POS	Pair of POS tags
Lemma	Pair of lemmas
Prep*	Preposition lexeme, if in a prepositional phrase
Sent. count	Quantized number of sentences between triggers
Word count	Quantized number of words between triggers
LCA	Least common ancestor on constituency tree, if exists
Dominates*	Whether one trigger dominates other
Share	Whether triggers share a child on dependency tree
Adjacency	Whether two triggers are adjacent
Words btw.	For adjacent triggers, content words between triggers
Temp. btw.	For adjacent triggers, temporal connectives (from a small list) between triggers

Table 3: Features extracted for a trigger pair (t_i, t_j). Asteriks (*) indicate features that are duplicated, once for each trigger.

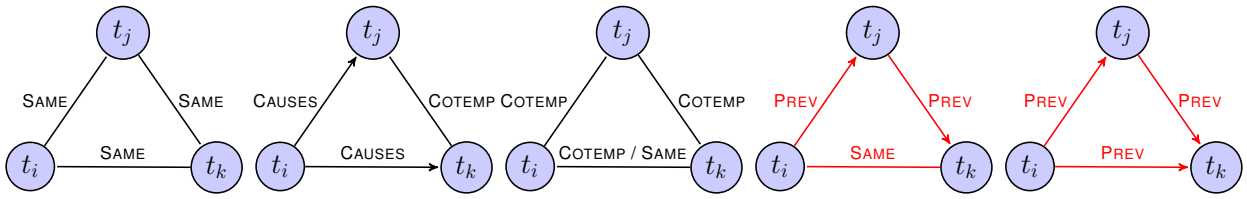
3.2 Classifier extensions

A central source of information to extract event-event relations from text are *connectives* such as *after*, *during*, etc. However, there is variability in the occurrence of these connectives as demonstrated by the following two sentences (connectives in boldface, triggers in italics):

1. **Because** alleles are *exchanged* during *gene flow*, genetic differences are *reduced*.

2. During *gene flow*, alleles are *exchanged*, and genetic differences are **hence** *reduced*.

Even though both sentences express the same relation $(\textit{exchanged}, \textit{reduced}, \text{CAUSES})$, the connectives used and their linear position with respect to the triggers are different. Also, in sentence 1, *gene flow* intervenes between *exchanged* and *reduced*. Since our dataset is small, we wish to identify the triggers related to each connective, and share features between such sentences. We do this using the syntactic structure and by clustering the connectives.



(a) SAME transitivity (b) CAUSE-COTEMP (c) COTEMP transitivity (d) SAME contradiction (e) PREV contradiction

Figure 2: Relation triangles (a)-(c) are common in the gold standard while (d)-(e) are impossible.

Sentence 1 presents a typical case where by walking up the dependency tree from the marker *because*, we can find the triggers related by this marker: *because* $\xleftarrow{mark}$ *exchanged* $\xleftarrow{advcl}$ *reduced*. Whenever a trigger is the head of an adverbial clause and marked by a *mark* dependency label, we walk on the dependency tree and look for a trigger in the main clause that is closest to the root (or the root itself in this example). By utilizing the syntactic structure, we can correctly spot that the trigger *gene flow* is not related to the trigger *exchanged* through the connective *because*, even though they are linearly closer. In order to reduce sparseness of connectives, we created a hand-made clustering of 30 connectives that maps words into clusters[2] (e.g., *because*, *since* and *hence* to a "causality" cluster). After locating the relevant pair of triggers, we use these clusters to fire the same feature for connectives belonging to the same cluster. We perform a similar procedure whenever a trigger is part of a prepositional phrase (imagine sentence 1 starting with *"due to allele exchange during gene flow ..."*) by walking up the constituency tree, but details are omitted for brevity. In sentence 2, the connective *hence* is an adverbial modifier of the trigger *reduced*. We look up the cluster for the connective *hence* and fire the same feature for the adjacent triggers *exchanged* and *reduced*.

We further extend our features to handle the rich relation set necessary for process extraction. The first event of a process is often expressed as a nominalization and includes subsequent events (SUPER relation), e.g., "The *Calvin cycle* begins by *incorporating*...". To capture this, we add a feature that fires when the first event of the process description is a noun. We also add two features targeted at the SAME relation: one indicating if the lemmas of t_i and t_j are the same, and another specifying the determiner of t_j, if it exists. Certain determiners indicate that an event trigger has already been mentioned, e.g., the determiner *this* hints a SAME relation in "The next steps *decompose* citrate back to oxaloacetate. This *regeneration* makes ...". Last, we add as a feature the dependency path between t_i and t_j, if it exists, e.g., in "meiosis produces cells that divide ...", the feature $\xrightarrow{dobj}\xrightarrow{rcmod}$ is fired for the trigger pair *produces* and *divide*. In Section 4.1 we empirically show that our extensions to the local classifier substantially improve performance.

For our pairwise classifier, we train a maximum entropy classifier that computes a probability p_{ijr} for every trigger pair (t_i, t_j) and relation r. Hence, $f(t_i, t_j) = \arg\max_r p_{ijr}$.

3.3 Global Constraints

Naturally, pairwise classifiers are local models that can violate global properties in the process structure. Figure 3 (left) presents an example for predictions made by the pairwise classifier, which result in two triggers (*deleted* and *dupcliated*) that are isolated from the rest of the triggers. In this section, we discuss how we incorporate constraints into our model to generate coherent global process structures.

Let θ_{ijr} be the score for a relation r between the trigger pair (t_i, t_j) (e.g., $\theta_{ijr} = \log p_{ijr}$), and y_{ijr} be the corresponding indicator variable. Our goal is to find an assignment for the indicators $\mathbf{y} = \{y_{ijr} \mid 1 \leq i < j \leq n, r \in \mathcal{R}\}$. With no global constraints this can be formulated as the following ILP:

[2]The full set of connectives and their clustering are provided as part of our publicly released package.

$$\arg\max_{\mathbf{y}} \sum_{ijr} \theta_{ijr} y_{ijr} \quad (1)$$
$$\text{s.t.} \forall_{i,j} \sum_{r} y_{ijr} = 1$$

where the constraint ensures exactly one relation between each event pair. We now describe constraints that result in a coherent global process structure.

Connectivity Our ILP formulation for enforcing connectivity is a minor variation of the one suggested by Martins et al. (2009) for dependency parsing. In our setup, we want $\mathcal{P}$ to be a connected undirected graph, and not a directed tree. However, an undirected graph $\mathcal{P}$ is connected iff there exists a directed tree that is a subgraph of $\mathcal{P}$ when edge directions are ignored. Thus the resulting formulation is almost identical and is based on flow constraints which ensure that there is a path from a designated root in the graph to all other nodes.

Let $\bar{\mathcal{R}}$ be the set $\mathcal{R} \setminus$ NONE. An edge (t_i, t_j) is in E iff there is some non-NONE relation between t_i and t_j, i.e. iff $y_{ij} := \sum_{r \in \bar{\mathcal{R}}} y_{ijr}$ is equal to 1. For each variable y_{ij} we define two auxiliary binary variables z_{ij} and z_{ji} that correspond to edges of the directed tree that is a subgraph of $\mathcal{P}$. We ensure that the edges in the tree exist also in $\mathcal{P}$ by tying each auxiliary variable to its corresponding ILP variable:

$$\forall_{i<j}\ z_{ij} \leq y_{ij}, z_{ji} \leq y_{ij} \quad (2)$$

Next, we add constraints that ensure that the graph structure induced by the auxiliary variables is a tree rooted in an arbitrary node 1 (The choice of root does not affect connectivity). We add for every $i \neq j$ a flow variable ϕ_{ij} which specifies the amount of flow on the directed edge z_{ij}.

$$\sum_{i} z_{i1} = 0, \forall_{j \neq 1} \sum_{i} z_{ij} = 1 \quad (3)$$

$$\sum_{i} \phi_{1i} = n - 1 \quad (4)$$

$$\forall_{j \neq 1} \sum_{i} \phi_{ij} - \sum_{k} \phi_{jk} = 1 \quad (5)$$

$$\forall_{i \neq j}\ \phi_{ij} \leq n \cdot z_{ij} \quad (6)$$

Equation 3 says that all nodes in the graph have exactly one parent, except for the root that has no parents. Equation 4 ensures that the outgoing flow from the root is $n-1$, and Equation 5 states that each of the other $n-1$ nodes consume exactly one unit of flow. Last, Equation 6 ties the auxiliary variables to the flow variables, making sure that flow occurs only on edges. The combination of these constraints guarantees that the graph induced by the variables z_{ij} is a directed tree and consequently the graph induced by the objective variables $\mathbf{y}$ is connected.

Chain structure A chain is a connected graph where the degree of all nodes is ≤ 2. Table 2 presents nodes' degree and demonstrates that indeed process graphs are close to being chains. The following constraint bounds nodes' degree by 2:

$$\forall_j (\sum_{i<j} y_{ij} + \sum_{j<k} y_{jk} \leq 2) \quad (7)$$

Since graph structures are not always chains, we add this as a soft constraint, that is, we penalize the objective for each node with degree > 2. The chain structure is one of the several soft constraints we enforce. Thus, our modified objective function is $\sum_{ijr} \theta_{ijr} y_{ijr} + \sum_{k \in \mathcal{K}} \alpha_k C_k$, where $\mathcal{K}$ is the set of soft constraints, α_k is the penalty (or reward for desirable structures), and C_k indicates whether a constraint is violated (or satisfied). Note that under this formulation our model is simply a constrained conditional model (wei Chang et al., 2012). The parameters α_k are tuned on a development set (see Section 4).

Relation triads A relation triad (or a relation triangle) for any three triggers t_i, t_j and t_k in a process is a 3-tuple of relations $(f(t_i, t_j), f(t_j, t_k), f(t_i, t_k))$. Clearly, some triads are impossible while others are quite common. To find triads that could improve process extraction, the frequency of all possible triads in both the training set and the output of the pairwise classifier were found, and we focused on those for which the classifier and the gold standard disagree. We are interested in triads that never occur in training data but are predicted by the classifier, and vice versa. Figure 2 illustrates some of the triads found and Equa-

tions 8-12 provide the corresponding ILP formulations. Equations 8-10 were formulated as soft constraints (expanding the set $\mathcal{K}$) and were incorporated by defining a reward α_k for each triad type.[3] On the other hand, Equations 11-12 were formulated as hard constraints to prevent certain structures.

1. SAME transitivity (Figure 2a, Eqn. 8): Coreference transitivity has been used in past work (Finkel and Manning, 2008) and we incorporate it by a constraint that encourages triads that respect transitivity.
2. CAUSE-COTEMP (Figure 2b, Eqn. 9): If t_i causes both t_j and t_k, then often t_j and t_k are co-temporal. E.g, in "*genetic drift* has led to a *loss* of genetic variation and an *increase* in the frequency of . . .", a single event causes two subsequent events that occur simultaneously.
3. COTEMP transitivity (Figure 2c, Eqn. 10): If t_i is co-temporal with t_j and t_j is co-temporal with t_k, then usually t_i and t_k are either co-temporal or denote the same event.
4. SAME contradiction (Figure 2d, Eqn. 11): If t_i is the same event as t_k, then their temporal ordering with respect to a third trigger t_j may result in a contradiction, e.g., if t_j is after t_i, but before t_k. We define 5 temporal categories that generate $\binom{5}{2}$ possible contradictions, but for brevity present just one representative hard constraint. This constraint depends on prediction of temporal and co-reference relations jointly.
5. PREV contradiction (Figure 2e, Eqn. 12): As mentioned (Section 3.3), if t_i is immediately before t_j, and t_j is immediately before t_k, then t_i cannot be immediately before t_k.

$$y_{ij\text{SAME}} + y_{jk\text{SAME}} + y_{ik\text{SAME}} \geq 3 \quad (8)$$

$$y_{ij\text{CAUSES}} + y_{ik\text{CAUSES}} + y_{jk\text{COTEMP}} \geq 3 \quad (9)$$

$$y_{ij\text{COTEMP}} + y_{jk\text{COTEMP}} + y_{ik\text{COTEMP}} + y_{ik\text{SAME}} \geq 3 \quad (10)$$

$$y_{ij\text{PREV}} + y_{jk\text{PREV}} + y_{ik\text{SAME}} \leq 2 \quad (11)$$

$$y_{ij\text{PREV}} + y_{jk\text{PREV}} - y_{ik\text{NONE}} \leq 1 \quad (12)$$

[3]We experimented with a reward for certain triads or a penalty for others and empirically found that using rewards results in better performance on the development set.

We used the Gurobi optimization package[4] to find an exact solution for our ILP, which contains $O(n^2|\mathcal{R}|)$ variables and $O(n^3)$ constraints. We also developed an equivalent formulation amenable to dual decomposition (Sontag et al., 2011), which is a faster approximation method. But practically, solving the ILP exactly with Gurobi was quite fast (average/median time per process: 0.294 sec/0.152 sec on a standard laptop).

4 Experimental Evaluation

We extracted 148 process descriptions by going through chapters from the textbook "Biology" and marking any contiguous sequence of sentences that describes a process, i.e., a series of events that lead towards some objective. Then, each process description was annotated by a biologist. The annotator was first presented with annotation guidelines and annotated 20 descriptions. The annotations were then discussed with the authors, after which all process descriptions were annotated. After training a second biologist, we measured inter-annotator agreement $\kappa = 0.69$, on 30 random process descriptions.

Process descriptions were parsed with Stanford constituency and dependency parsers (Klein and Manning, 2003; de Marneffe et al., 2006), and 35 process descriptions were set aside as a test set (number of training set trigger pairs: 1932, number of test set trigger pairs: 906). We performed 10-fold cross validation over the training set for feature selection and tuning of constraint parameters. For each constraint type (connectivity, chain-structure, and five triad constraints) we introduced a parameter and tuned the seven parameters by coordinate-wise ascent, where for hard constraints a binary parameter controls whether the constraint is used, and for soft constraints we attempted 10 different reward/penalty values. For our global model we defined $\theta_{ijr} = \log p_{ijr}$, where p_{ijr} is the probability at edge (t_i, t_j) for label r, given by the pairwise classifier.

We test the following systems: (a) *All-Prev*: Since the most common process structure was chain-like, we simply predict PREV for every two adjacent triggers in text. (b) $Local_{base}$: A pairwise classifier with features from previous work (Section 3.1) (c) *Local*:

[4]www.gurobi.com

	Temporal			Full		
	P	R	F_1	P	R	F_1
All-Prev	58.4	54.8	56.6	34.1	32.0	33.0
$Local_{base}$	61.5	51.8	56.2	52.1	43.9	47.6
Local	63.2	55.7†	59.2	54.7	48.3†	51.3
Chain	**64.5**	60.5†‡	62.4†	56.1	52.6†‡	54.3†
Global	63.9	**61.4**†‡	**62.6**†‡	**56.2**	**54.0**†‡	**55.0**†‡

Table 4: Test set results on all experiments. Best number in each column is bolded. † and ‡ denote statistical significance ($p < 0.01$) against $Local_{base}$ and *Local* baselines, respectively.

A pairwise classifier with all features (Section 3.2) (d) *Chain*: For every two adjacent triggers, choose the non-NONE relation with highest probability according to *Local*. This baseline heuristically combines our structural assumptions with the pairwise classifier. We deterministically choose a connected chain structure, and then use the classifier to label the edges. (e) *Global*: Our full model that uses ILP inference.

To evaluate system performance we compare the set of predictions on all trigger pairs to the gold standard annotations and compute micro-averaged precision, recall and F_1. We perform two types of evaluations: (a) *Full*: evaluation on our full set of 11 relations (b) *Temporal:* Evaluation on temporal relations only, by collapsing PREV, CAUSES, and ENABLES to a single category and similarly for NEXT, CAUSED, and ENABLED (inter-annotator agreement $\kappa = 0.75$). We computed statistical significance of our results with the paired bootstrap resampling method of 2000 iterations (Efron and Tibshirani, 1993), where the units resampled are trigger-trigger-relation triples.

4.1 Results

Table 4 presents performance of all systems. We see that using global constraints improves performance almost invariably on all measures in both full and temporal evaluations. Particularly, in the full evaluation *Global* improves recall by 12% and overall F_1 improves significantly by 3.7 points against *Local* ($p < 0.01$). Recall improvement suggests that modeling connectivity allowed *Global* to add correct relations in cases where some events were not connected to one another.

The *Local* classifier substantially outperforms $Local_{base}$. This indicates that our novel features (Section 3.2) are important for discriminating between process relations. Specifically, in the full evaluation *Local* improves precision more than in the temporal evaluation, suggesting that designing syntactic and semantic features for connectives is useful for distinguishing PREV, CAUSES, and ENABLES when the amount of training data is small.

The *Chain* baseline performs only slightly worse than our global model. This demonstrates the strong tendency of processes to proceed linearly from one event to the other, which is a known property of discourse structure (Schegloff and Sacks, 1973). However, since the structure is deterministically fixed, *Chain* is highly inflexible and does not allow any extensions or incorporation of other structural constraints or domain knowledge. Thus, it can be used as a simple and efficient approximation but is not a good candidate for a real system. Further support for the linear nature of process structure is provided by the *All-Prev* baseline, which performs poorly in the full evaluation, but in temporal evaluation works reasonably well.

Table 2 presents the degree distribution of *Local* and *Global* on the development set comparing to the gold standard. The degree distribution of *Global* is more similar to the gold standard than *Local*. In particular, the connectivity constraint ensures that there are no isolated nodes and shifts mass from nodes with degree 0 and 1 to nodes with degree 2.

Table 5 presents the order in which constraints were introduced into the global model using coordinate ascent on the development set. Connectivity is the first constraint to be introduced, and improves performance considerably. The chain constraint, on the other hand, is included third and the improvement in F_1 score is relatively smaller. This can be explained by the distribution of degrees in Table 2 which shows that the predictions of *Local* does not have many nodes with degree > 2. As for triad constraints, we see that four constraints are important and are included in the model, but one is discarded.

Last, we examined the results of *Global* when macro-averaging over processes, i.e., assigning each process the same weight by computing recall, precision and F_1 for each process and averaging those scores. We found that results are quite similar (with a slight improvement): in the full evalua-

Order	Parameter name	Value (α)	F_1 score
–	*Local model*	–	49.9
1	Connectivity constraint	∞	51.2
2	SAME transitivity	0.5	52.9
3	Chain constraint	-0.5	53.3
4	CAUSE-COTEMP	1.0	53.7
6	PREV contradiction	∞	53.8
7	SAME contradiction	∞	53.9

Table 5: Order by which constraint parameters were set using coordinate ascent on the development set. For each parameter, the value chosen and F_1 score after including the constraint are provided. Negative values correspond to penalties, positive values to rewards, and a value of ∞ indicates a hard constraint.

tion *Global* obtains R/P/F_1 of 56.4/55.0/55.7, and in the temporal evaluation *Global* obtains R/P/F_1 of 63.8/62.3/63.1.

4.2 Qualitative Analysis

Figure 3 shows two examples where global constraints corrected the predictions of *Local*. In Figure 3, left, *Local* failed to predict the causal relations *skipped-deleted* and *used-duplicated*, possibly because they are not in the same sentence and are not adjacent to one another. By enforcing the connectivity constraint, *Global* correctly adds the correct relations and connects *deleted* and *duplicated* to the other triggers in the process.

In Figure 3, right, *Local* predicts a structure that results in a "SAME contradiction" structure. The triggers *bind* and *binds* cannot denote the same event if a third trigger *secrete* is temporally between them. However, *Local* predicts they are the same event, as they share a lemma. *Global* prohibits this structure and correctly predicts the relation as NONE.

To better understand the performance of *Local*, we analyzed the confusion matrix generated based on its predictions. Although this is a challenging 11-class classification task, most of the mass is concentrated on the matrix diagonal, as desired. Error analysis reveals that 17.5% of all errors are confusions between NONE and PREV, 11.1% between PREV and CAUSES, and 8.6% between PREV and COTEMP. This demonstrates that distinguishing the classes PREV, CAUSES and COTEMP is challenging for *Local*. Our current global constraints do not address this type of error, and thus an important direction for future work is to improve the local model.

The global model depends on the predictions of the local classifier, and so enforcing global constraints does not guarantee improvement in performance. For instance, if *Local* produces a graph that is disconnected (e.g., *deleted* in Figure 3, left), then *Global* will add an edge. However, the label of the edge is determined by scores computed based on the local classifier, and if this prediction is wrong, we will now be penalized for both the false negative of the correct class (just as before), and also for the false positive of the predicted class. Despite that we see that *Global* improves overall performance by 3.7 F_1 points on the test set.

5 Related Work

A related line of work is biomedical event extraction in recent BioNLP shared tasks (Kim et al., 2009; Kim et al., 2011). Earlier work employed a pipeline architecture where first events are found, and then their arguments are identified (Miwa et al., 2010; Björne et al., 2011). Subsequent methods predicted events and arguments jointly using Markov logic (Poon and Vanderwende, 2010) and dependency parsing algorithms (McClosky et al., 2011). Riedel and McCallum (2011) further improved performance by capturing correlations between events and enforcing consistency across arguments.

Temporal event-event relations have been extensively studied (Chambers and Jurafsky, 2008; Yoshikawa et al., 2009; Denis and Muller, 2011; Do et al., 2012; McClosky and Manning, 2012; D'Souza and Ng, 2013), and we leverage such techniques in our work (Section 3.1). However, we extend beyond temporal relations alone, and strongly rely on dependencies between process events. Chambers and Jurafsky (2011) learned event templates (or frames), where events that are related to one another and their semantic roles are extracted. Recently, Cheung et al. (2013) proposed an unsupervised generative model for inducing such templates. A major difference in our work is that we do not learn typical event relations from a large and redundant corpus, but are given a paragraph and have a "one-shot" chance to extract the process structure.

We showed in this paper that global structural properties lead to significant improvements in extraction accuracy, and ILP is an effective framework

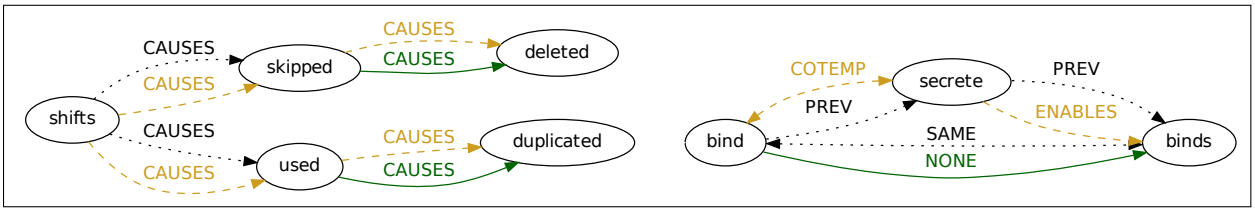


Figure 3: Process graph fragments. Black edges (dotted) are predictions of *Local*, green (solid) are predictions of *Global*, and gold (dashed) are gold standard edges. To reduce clutter, we present the predictions of *Global* only when it disagrees with *Local*. In all other cases, the predictions of *Global* and *Local* are identical. Original text, Left: "... the template *shifts* ..., and a part of the template strand is either *skipped* by the replication machinery or *used* twice as a template. As a result, a segment of DNA is *deleted* or *duplicated*." Right: "Cells of mating type A secrete a signaling molecule, which can *bind* to specific receptor proteins on nearby cells. At the same time, cells *secrete* factor, which *binds* to receptors on A cells."

for modeling global constraints. Similar observations and techniques have been proposed in other information extraction tasks. Reichart and Barzilay (2012) tied information from multiple sequence models that describe the same event by using global higher-order potentials. Berant et al. (2011) proposed a global inference algorithm to identify entailment relations. There is an abundance of examples of enforcing global constraints in other NLP tasks, such as in coreference resolution (Finkel and Manning, 2008), parsing (Rush et al., 2012) and named entity recognition (Wang et al., 2013).

6 Conclusion

Developing systems that understand process descriptions is an important step towards building applications that require deeper reasoning, such as biological process models from text, intelligent tutoring systems, and non-factoid QA systems. In this paper we have presented the task of process extraction, and developed methods for extracting relations between process events. Processes contain events that are tightly coupled through strong dependencies. We have shown that exploiting these structural dependencies and performing joint inference over all event mentions can significantly improve accuracy over several baselines. We have also released a new dataset containing 148 fully annotated descriptions of biological processes. Though the models we built were trained on biological processes, they do not encode domain specific information, and hence should be extensible to other domains.

In this paper we assumed that event triggers are given as input. In future work, we want to perform trigger identification jointly with extraction of event-event relations. As explained in Section 4.2, the performance of our system is confined by the performance of the local classifier, which is trained on relatively small amounts of data. Since data annotation is expensive, it is important to improve the local classifier without increasing the annotation burden. For example, one can use unsupervised methods that learn narrative chains (Chambers and Jurafsky, 2011) to provide some prior on the typical order of events. Alternatively, we can search on the web for redundant descriptions of the same process and use this redundancy to improve classification. Last, we would like to integrate our method into QA systems and allow non-factoid questions that require deeper reasoning to be answered by matching the questions against the learned process structures.

Acknowledgments

The authors would like to thank Roi Reichart for fruitful discussion and the anonymous reviewers for their constructive feedback. This work was partially funded by Vulcan Inc. The second author was sponsored by a Rothschild fellowship.

References

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Learning entailment relations by global graph structure optimization. *Journal of Computational Linguistics*, 38(1).

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2011. Extracting contextualized complex biological events with rich graph-based feature sets. *Computational Intelligence*, 27(4):541–557.

Neil Campbell and Jane Reece. 2005. *Biology*. Benjamin Cummings.

Nathanael Chambers and Daniel Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *ACL*, pages 976–986.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of NAACL-HLT*.

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

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *Proceedings of IJCAI*.

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of EMNLP-CoNLL*.

Jennifer D'Souza and Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge. In *Proceedings of NAACL-HLT*.

Bradley Efron and Robert Tibshirani. 1993. *An introduction to the bootstrap*, volume 57. CRC press.

Christiane Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of ACL*.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Junichi Tsujii. 2009. Overview of BioNLP 09 shared task on event extraction. In *Proceedings of BioNLP*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, and Junichi Tsujii. 2011. Overview of BioNLP shared task 2011. In *Proceedings of BioNLP*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*.

André L. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL/IJCNLP*.

David McClosky and Christopher D. Manning. 2012. Learning constraints for consistent timeline extraction. In *Proceedings of EMNLP-CoNLL*, pages 873–882.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL*, pages 1626–1635.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *J. Bioinformatics and Computational Biology*, 8(1).

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Proceedings of HLT-NAACL*.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003. TimeML: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering*.

Roi Reichart and Regina Barzilay. 2012. Multi-event extraction guided by global constraints. In *Proceedings of HLT-NAACL*.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of EMNLP*.

Alexander M. Rush, Roi Reichert, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *Proceedings of EMNLP*.

Emanuel A Schegloff and Harvey Sacks. 1973. Opening up closings. *Semiotica*, 8(4):289–327.

David Sontag, Amir Globerson, and Tommi Jaakkola. 2011. Introduction to dual decomposition for inference. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*. MIT Press.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2).

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

Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *Proceedings of AAAI*.

Ming wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with Markov logic. In *Proceedings of ACL/IJCNLP*.

Generating Coherent Event Schemas at Scale

Niranjan Balasubramanian, Stephen Soderland, Mausam, Oren Etzioni
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{niranjan,ssoderlan, mausam, etzioni}@cs.washington.edu

Abstract

Chambers and Jurafsky (2009) demonstrated that event schemas can be automatically induced from text corpora. However, our analysis of their schemas identifies several weaknesses, e.g., some schemas lack a common topic and distinct roles are incorrectly mixed into a single actor. It is due in part to their pair-wise representation that treats subject-verb independently from verb-object. This often leads to subject-verb-object triples that are not meaningful in the real-world.

We present a novel approach to inducing open-domain event schemas that overcomes these limitations. Our approach uses co-occurrence statistics of semantically typed relational triples, which we call Rel-grams (relational n-grams). In a human evaluation, our schemas outperform Chambers's schemas by wide margins on several evaluation criteria. Both Rel-grams and event schemas are freely available to the research community.

1 Introduction

Event schemas (also known as templates or frames) have been widely used in information extraction. An event schema is a set of actors (also known as slots) that play different roles in an event, such as the perpetrator, victim, and instrument in a bombing event. They provide essential guidance in extracting information related to events from free text (Patwardhan and Riloff, 2009), and can also aid in other NLP tasks, such as coreference (Irwin et al., 2011), summarization (Owczarzak and Dang, 2010), and inference about temporal ordering and causality.

Actor	Rel	Actor
A1:<person>	failed	A2:test
A1:<person>	was suspended for	A3:<time period>
A1:<person>	used	A4:<substance, drug>
A1:<person>	was suspended for	A5:<game, activity>
A1:<person>	was in	A6:<location>
A1:<person>	was suspended by	A7:<org, person>
Actor Instances:		
A1: {Murray, Morgan, Governor Bush, Martin, Nelson}		
A2: {test}		
A3: {season, year, week, month, night}		
A4: {cocaine, drug, gasoline, vodka, sedative}		
A5: {violation, game, abuse, misfeasance, riding}		
A6: {desert, Simsbury, Albany, Damascus, Akron}		
A7: {Fitch, NBA, Bud Selig, NFL, Gov Jeb Bush}		

Table 1: An event schema produced by our system, represented as a set of $(Actor, Rel, Actor)$ triples, and a set of instances for each actor $A1$, $A2$, etc. For clarity we show unstemmed verbs.

Until recently, all event schemas in use in NLP were hand-engineered, *e.g.*, the MUC templates and ACE event relations (ARPA, 1991; ARPA, 1998; Doddington et al., 2004). This led to technology that could only focus on specific domains of interest and has not been applicable more broadly.

The seminal work of Chambers and Jurafsky (2009) showed that event schemas can also be induced automatically from text corpora. Instead of labeled roles these schemas have a set of relations and actors that serve as arguments.[1] Their system is fully automatic, domain-independent, and scales to large text corpora.

However, we identify several limitations in the schemas produced by their system.[2] Their schemas

[1] In the rest of this paper we use event schemas to refer to these automatically induced schemas with actors and relations.

[2] Available at http://www.usna.edu/Users/cs/nchamber/data/schemas/acl09

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1721–1731,
Seattle, Washington, USA, 18-21 October 2013.

Actor	Rel	Actor
A1	caused	A2
A2	spread	A1
A2	burned	A1
-	extinguished	A1
A1	broke out	-
-	put out	A1
Actor Instances: A1: {fire, aids, infection, disease} A2: {virus, bacteria, disease, urushiol, drug}		

Table 2: An event schema from Chambers' system that mixes the events of fire spreading and disease spreading.

often lack coherence: mixing unrelated events and having actors whose entities do not play the same role in the schema. Table 2 shows an event schema from Chambers that mixes the events of fire spreading and disease spreading.

Much of the incoherence of Chambers' schemas can be traced to their representation that uses *pairs* of elements from an assertion, thus, treating subject-verb and verb-object separately. This often leads to subject-verb-object triples that do not make sense in the real world. For example, the assertions "fire caused virus" and "bacteria burned AIDS" are implicit in Table 2.

Another limitation in schemas Chambers released is that they restrict schemas to two actors, which can result in combining different actors. Table 4 shows an example of combining perpeterators and victims into a single actor.

1.1 Contributions

We present an event schema induction algorithm that overcomes these weaknesses. Our basic representation is *triples* of the form (Arg1, Relation, Arg2), extracted from a text corpus using Open Information Extraction (Mausam et al., 2012). The use of triples aids in agreement between subject and object of a relation. The use of Open IE leads to more expressive relation phrases (*e.g.*, with prepositions). We also assign semantic types to arguments, both to alleviate data sparsity and to produce coherent actors for our schemas.

Table 1 shows an event schema generated by our system. It has six relations and seven actors. The schema makes several related assertions about a person using a drug, failing a test, and getting suspended. The main actors in the schema include the person who failed the test, the drug used, and the agent that suspended the person.

Our first step in creating event schemas is to tabulate co-occurrence of tuples in a database that we call Rel-grams (relational n-grams) (Sections 3, 5.1). We then perform analysis on a graph induced from the Rel-grams database and use this to create event schemas (Section 4).

We compared our event schemas with those of Chambers on several metrics including whether the schema pertains to a coherent topic or event and whether the actors play a coherent role in that event (Section 5.2). Amazon Mechanical Turk workers judged that our schemas have significantly better coherence – 92% versus 82% have coherent topic and 81% versus 59% have coherent actors.

We release our open domain event schemas and the Rel-grams database[3] for further use by the NLP community.

2 System Overview

Our approach to schema generation is based on the idea that frequently co-occurring relations in text capture relatedness of assertions about real-world events. We begin by extracting a set of relational tuples from a large text corpus and tabulate occurrence of pairs of tuples in a database.

We then construct a graph from this database and identify high-connectivity nodes (relational tuples) in this graph as a starting point for constructing event schemas. We use graph analysis to rank the tuples and merge arguments to form the actors in the schema.

3 Modeling Relational Co-occurrence

In order to tabulate pairwise occurences of relational tuples we need a suitable relation-based representation. We now describe the extraction and representation of relations, a database for storing co-occurrence information, and our probabilistic model for the co-occurrence. We call this model Rel-grams, as it can be seen as a relational analog to the n-grams language model.

3.1 Relations Extraction and Representation

We extract relational triples from each sentence in a large corpus using the OLLIE Open IE system

[3] Available at http://relgrams.cs.washington.edu

Tuples Table

Id	Arg1	Rel	Arg2	Count
...	...	...	...	...
13	bomb	explode in	<loc>	547
14	bomb	explode in	Baghdad	22
15	bomb	explode in	market	7
...	...	...	...	...
87	bomb	kill	<per>	173
...	...	...	...	...
92	<loc>	be suburb of	<loc>	1023
...	...	...	...	...

BigramCounts Table

T1	T2	Dist.	Count	E11	E12	E21	E22
...	...	...	...	...	...	...	...
13	87	1	27	25	0	0	0
13	87	2	35	33	0	0	0
...	...	...	...	...	...	...	...
13	87	10	62	59	0	0	0
87	13	1	6	0	0	0	0
...	...	...	...	...	...	...	...
92	13	1	12	0	0	12	0
...	...	...	...	...	...	...	...

Figure 1: Tables in the Rel-grams Database: *Tuples* maps tuples to unique identifiers, *BigramCounts* provides the co-occurrence counts (Count) within various distances (Dist.), and four types of argument equality counts (E11-E22). E11 is the number of times when T1.Arg1 = T2.Arg1, E12 is when T1.Arg1 = T2.Arg2 and so on.

(Mausam et al., 2012).[4] This provides relational tuples in the format (Arg1, Relation, Arg2) where each tuple element is a phrase from the sentence. The sentence "He cited a new study that was released by UCLA in 2008." produces three tuples:

1. (He, cited, a new study)
2. (a new study, was released by, UCLA)
3. (a new study, was released in, 2008)

Relational triples provide a more specific representation which is less ambiguous when compared to (subj, verb) or (verb, obj) pairs. However, using relational triples also increases sparsity. To reduce sparsity and to improve generalization, we represent the relation phrase by its stemmed head verb plus any prepositions. The relation phrase may include embedded nouns, in which case these are stemmed as well. Moreover, tuple arguments are represented as stemmed head nouns, and we also record semantic types of the arguments.

We selected 29 semantic types from WordNet, examining the set of instances on a small development set to ensure that the types are useful, but not overly specific. The set of types are: person, organization, location, time_unit, number, amount, group, business, executive, leader, effect, activity, game, sport, device, equipment, structure, building, substance, nutrient, drug, illness, organ, animal, bird, fish, art, book, and publication.

To assign types to arguments, we apply Stanford Named Entity Recognizer (Finkel et al., 2005)[5], and also look up the argument in WordNet 2.1 and record the first three senses if they map to our target semantic types. We use regular expressions to recognize dates and numeric expressions, and map personal pronouns to <person>. We associate all types found by this mechanism with each argument. The tuples in the example above are normalized to the following:

1. (He, cite, study)
2. (He, cite, <activity>)
3. (<person>, cite, study)
4. (<person>, cite, <activity>)
5. (study, be release by, UCLA)
6. (study, be release by, <organization>)
7. (study, be release in, 2008)
8. (study, be release in, <time_unit>)
9. (<activity>, be release by, UCLA)

...

In our preliminary experiments, we found that using normalized relation strings and semantic classes for arguments results in a ten-fold increase in the number of Rel-grams with a minimum support.

3.2 Co-occurrence Tabulation

We construct a database to hold co-occurrence statistics for pairs of tuples found in each document. Figure 1 shows examples for the types of statistics contained in the database. The database consists of two tables: 1) *Tuples* – Maps each tuple to a unique identifier and tabulates tuple counts. 2) *BigramCounts* – Stores the directional co-occurrence frequency, a count for tuple T followed by T' at a distance of k, and tabulates the number of times the same argument was present in the pair of tuples.

Equality Constraints: Along with the co-occurrence counts, we record the equality of arguments in Rel-grams pairs. We assert an argument

[4]Available at: http://knowitall.github.io/ollie/

[5]We used the system downloaded from: http://nlp.stanford.edu/software/CRF-NER.shtml and used the seven class CRF model distributed with it.

Table 3: Given a source tuple, the Rel-grams language model estimates the probability of encountering other relational tuples in a document. For clarity, we show the unstemmed version.

Top tuples related to (<person>, convicted of, murder)
1. (<person>, convicted in, <time_unit>)
2. (<person>, sentenced to, death)
3. (<person>, sentenced to, year)
4. (<person>, convicted in, <location>)
5. (<person>, sentenced to, life)
6. (<person>, convicted in, <person>)
7. (<person>, convicted after, trial)
8. (<person>, sent to, prison)

pair is equal if they are from the same token sequence in the source sentence or one argument is a co-referent mention of the other. We use the Stanford Co-reference system (Lee et al., 2013)[6] to detect co-referring mentions. There are four possible equalities depending on the specific pair of arguments in the tuples are the same, shown as E11, E12, E21 and E22 in Figure 1. For example, the E21 column has counts for the number of times the Arg2 of T1 was determined to be the same as the Arg1 of T2.

Implementation and Query Language: We populated the Rel-grams database using OLLIE extractions from a set of 1.8 Million New York Times articles drawn from the Gigaword corpus. The database consisted of approximately 320K tuples that have frequency ≥ 3 and 1.1M entries in the bigram table.

The Rel-grams database allows for powerful querying using SQL. For example, Table 3 shows the most frequent rel-grams associated with the query tuple (<person>, convicted of, murder).

3.3 Rel-grams Language Model

From the tabulated co-occurrence statistics, we estimate bi-gram conditional probabilities of tuples that occur within a window of k tuples from each other. Formally, we use $P_k(T'|T)$ to denote the conditional probability that T' follows T within a window of k tuples. To discount estimates from low-frequency tuples, we use a δ-smoothed estimate:

$$P_k(T'|T) = \frac{\#(T, T', k) + \delta}{\sum_{T'' \in V} \#(T, T'', k) + \delta \cdot |V|} \quad (1)$$

where, $\#(T, T', k)$ is the number of times T' follows T within a window of k tuples. $k = 1$ indicates adjacent tuples in the document. $|V|$ is the number of unique tuples in the corpus. For experiments in this paper, we set δ to 0.05.

Co-occurrence within a small window is usually more reliable but is also sparse, whereas co-occurrence within larger windows addresses sparsity but may lead to topic drift. To leverage the benefits of different window sizes, we also define a metric with a weighted average of window sizes from 1 to 10, where the weight decays as window size increases. For example, with α set to 0.5 in equation 2, a window of $k+1$ has half the weight of a window of k.

$$P(T'|T) = \frac{\sum_{k=1}^{10} \alpha^k P_k(T'|T)}{\sum_{k=1}^{10} \alpha^k} \quad (2)$$

We believe that Rel-grams is a valuable source of common-sense knowledge and may be useful for several downstream tasks such as improving information extractors, inference of implicit information, etc. We assess its usefulness in the context of generating event schemas.

4 Schema Generation

We now use Rel-grams to identify relations and actors pertaining to a particular event. Our schema generation consists of three steps. First, we build a relation graph of tuples (G) using connections identified by Rel-grams. Second, we identify a set of seed tuples as starting points for schemas. We use graph analysis to find the tuples most related to each seed. Finally, we merge the arguments in these tuples to create actors and output the final schema. Next we describe each of these steps in detail.

4.1 Rel-graph construction

We define a Rel-graph as an undirected weighted graph $G = (V, E)$, whose vertices (V) are relation tuples with edges (E), where an edge between vertices T and T' is weighted by the symmetric conditional probability $SCP(T, T')$ defined as

[6] Available for download at: http://nlp.stanford.edu/software/dcoref.shtml

$$SCP(T, T') = P(T|T') \times P(T'|T) \quad (3)$$

Both conditional probabilities are computed in Equation 2. Figure 2 shows a portion of a Rel-graph where the thickness of the edge indicates symmetric conditional probability.

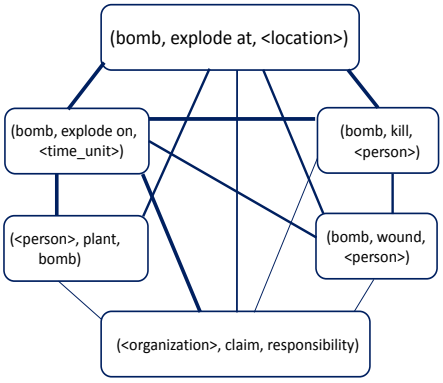


Figure 2: Part of a Rel-graph showing tuples strongly associated with *(bomb, explode at, <location>)*. Undirected edges are weighted by symmetric conditional probability with line thickness indicating weight.

4.2 Finding Related Tuples

Our goal is to find closely related tuples that pertain to an event or topic. First, we locate high-connectivity nodes in the Rel-graph to use as seeds. We sort nodes by the sum of their top 25 edge weights[7] and take the top portion of this list after filtering out redundant views of the same relation.

For each seed (Q), we find related tuples by extracting the sub-graph (G_Q) from Q's neighbors (within two hops from Q) in the Rel-graph. Graph analysis can detect the strongly connected nodes within this sub-graph, representing tuples that frequently co-occur in the context of the seed tuple.

Page rank is a well-known graph analysis algorithm that uses graph connectivity to identify important nodes within a graph (Brin and Page, 1998). We are interested in connectivity within a subgraph with respect to a designated query node (the seed). Connection to a query node can help minimize concept drift and ensure that the selected tuples are closely related to the main topic of the sub-graph.

[7]Limiting to the top 25 edges avoids overly general tuples that occur in many topics, which tend to have a large number of weak edges.

In this work, we adapt the Personalized PageRank algorithm (Haveliwala, 2002). The personalized version of PageRank returns ranks of various nodes with respect to a given query node and hence is more appropriate for our task than the basic PageRank algorithm. Within the subgraph G_Q for a given seed Q, we compute a solution to the following set of PageRank Equations:

$$\begin{aligned} & PR_Q(T) \\ &= (1-d) + d\sum_{T'} SCP(T,T')PR_Q(T') \quad \text{if } T = Q \\ &= d\sum_{T'} SCP(T,T')PR_Q(T') \quad \text{otherwise} \end{aligned}$$

Here $PR_Q(T)$ denotes the page rank of a tuple T personalized for the query tuple Q. It is a sum of all its neighbors' page ranks, weighted by the edge weights; d is the damping probability, which we set to be 0.85 in our implementation.

The solution is computed iteratively by initializing the page rank of Q to 1 and all others to 0, then recomputing page rank values until they converge to within a small ϵ. This computation remains scalable, since we restrict it to subgraphs a small number of hops away from the query node. This is a standard practice to handle large graphs (Agirre and Soroa, 2009; Mausam et al., 2010).

4.3 Creating Actors and Relations

We take the top n tuples from G_Q according to their Page rank scores. From each tuple $T : (Arg1, Rel, Arg2)$ in G_Q, we record two actors (A_1, A_2) corresponding to $Arg1$ and $Arg2$, and add Rel to the list of relations that they participate in.

Then, we merge actors in two steps. First, we collect the equality constraints for the tuples in G_Q. If the arguments corresponding to any pair of actors have a non-zero equality constraint then we merge them. Second, we merge actors that perform similar actions. A_1 and A_2 are merged if they are connected to the same actor A_3 through the same relation. For example, A1 and A2 in (A1:lawsuit, file by, A3:company) and (A2:suit, file by, A3:company), will be merged into a single actor. To avoid merging distinct actors, we use a small list of rules that specify the semantic type pairs that cannot be merged (*e.g.*, location-date). Also, we do not merge two actors, if it can result in a relation where the same actor

System	A_1	Rel	A_2
Relgrams	{bomb, missile, grenade, device}	explode in	{city, Bubqua, neighborhood}
	{bomb, missile, grenade, device}	explode kill	{people, civilian, lawmaker, owner, soldier}
	{bomb, missile, grenade, device}	explode on	{Feb., Fri., Tues., Sun., Sept.}
	{bomb, missile, grenade, device}	explode wound	{civilian, person, people, soldier, officer}
	{bomb, missile, grenade, device}	explode in	{Feb., Beirut Monday, Sept., Aug.}
	{bomb, missile, grenade, device}	explode injure	{woman, people, immigrant, policeman}
Chambers	{bomb, explosion, blast, **bomber**, mine}	explode	{soldier, child, civilian, bomber, palestinian}
	{soldier, **child**, **civilian**, bomber, palestinian}	set off	{bomb, explosion, blast, **bomber**, mine, bombing}
	{bomb, explosion, blast, bomber, mine}	kill	{soldier, child, civilian, bomber, palestinian}
	{soldier, **child**, **civilian**, bomber, palestinian}	detonate	{bomb, explosion, blast, **bomber**, mine, bombing}
	{bomb, explosion, blast, bomber, mine}	injure	{soldier, child, civilian, bomber, palestinian}
	{soldier, **child**, **civilian**, bomber, palestinian}	plant	{bomb, explosion, blast, **bomber**, mine, **bombing**}
Relgrams	{Carey, John Anthony Volpe, Chavez, She }	veto	{legislation, bill, law, measure, version}
	{legislation, bill, law, measure, version}	be sign by	{Carey, John Anthony Volpe, Chavez, She }
	{legislation, bill, law, measure, version}	be pass by	{State Senate, State Assembly, House, Senate, Parliament}
	{Carey, John Anthony Volpe, Chavez, She }	sign into	{law}
	{Carey, John Anthony Volpe, Chavez, She }	to sign	{bill}
	{Carey, John Anthony Volpe, Chavez, She }	be governor of	{Massachusetts, state, South Carolina, Texas, California}
Chambers	{clinton, bush, **bill**, president, house}	oppose	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, **house**}	sign	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, house}	approve	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, **house**	veto	{bill, measure, legislation, plan, law}
	{clinton, bush, **bill**, president, house}	support	{bill, measure, legislation, plan, law}
	{**clinton**, **bush**, **bill**, **president**, house}	pass	{bill, measure, legislation, plan, law}

Table 4: "Bombing" and "legislation" schema examples from Rel-grams and Chambers represented as a set of (A_1, Rel, A_2) tuples, where the schema provides a set of instances for each actor A_1 and A_2. Relations and arguments are in the stemmed form, e.g., 'explode kill' refers to 'exploded killing'. Instances in bold produce tuples that are not valid in the real world.

is both the Arg1 and Arg2.

Finally, we generate an ordered list of tuples using the final set of actors and their relations. The output tuples are sorted by the average page rank of the original tuples, thereby reflecting their importance within the sub-graph G_Q.

5 Evaluation

We present experiments to explore two main questions: How well do Rel-grams capture real world knowledge, and what is the quality of event schemas built using Rel-grams.

5.1 Evaluating Rel-grams

What sort of common-sense knowledge is encapsulated in Rel-grams? How often does it indicate an implication between a pair of statements, and how often does it indicate a common real-world event or topic? To answer these questions, we conducted an experiment to identify a subset of our Rel-grams database with high precision for two forms of common-sense knowledge:

- **Implication:** The Rel-grams express an implication from T to T' or from T' to T, a bi-directional form of the Recognizing Textual Entailment (RTE) guidelines (Dagan et al., 2005).
- **Common Topic:** Is there an underlying common topic or event to which both T and T' are relevant?

We also evaluated whether both T and T' are **valid tuples** that are well-formed and make sense in the real world, a necessary pre-condition for either implication or common topic.

We are particularly interested in the highest precision portion of our Rel-grams database. The database has 1.1M entries with support of at least three instances for each tuple. To find the highest precision subset of these, we identified tuples that have at least 25 Rel-grams, giving us 12,600 seed tuples with a total of over 280K Rel-grams. Finally, we sorted this subset by the total symmetrical conditional probability of the top 25 Rel-grams for each seed tuple.

We tagged a sample of this 280K set of Rel-grams for valid tuples, implication between T and T', and common topic. We found that in the top 10% of this set, 87% of the seed tuples were valid and 74% of the Rel-grams had both tuples valid. Of the Rel-grams

System	Id	A_1	Rel	A_2
Relgrams	R1	bomb	explode in	city
		bomb	explode kill	people
		bomb	explode on	Fri.
		...	...	...
Chambers	C1	blast	explode	child
		child	detonate	blast
		child	plant	bomb
		...	...	...

Table 5: A grounded instantiation of the schemas from Table 4, where each actor is represented as a randomly selected instance.

System	Id	A_1	Rel	A_2
Relgrams	R11	bomb	explode in	city
		missile	explode in	city
		grenade	explode in	city
		...	...	...
Relgrams	R21	missile	explode in	city
		missile	explode in	neighborhood
		missile	explode in	front
		...	...	...

Table 6: A schema instantiation used to test for actor coherence. Each of the top instances for A_1 or A_2 is presented, holding the relation and the other actor fixed.

with both tuples valid, 83% expressed an implication between the tuples, and 90% had a common topic.

There were several reasons for invalid tuples – parsing errors; binary projections of inherently n-ary relations, for example (<person>, put, <person>); head-noun only representation omitting essential information; and incorrect semantic types, primarily due to NER tagging errors.

While the Rel-grams suffer from noise in the tuple validity, there is clearly strong signal in the data about common topic and implication between tuples in the Rel-grams. As we demonstrate in the following section, an end task can use graph analysis techniques to amplify this strong signal, producing high-quality relational schemas.

5.2 Schemas Evaluation

In our schema evaluation, we are interested in assessing how well the schemas correspond to common-sense knowledge about real world events. To this end, we focus on three measures, *topical coherence*, *tuple validity*, and *actor coherence*.

A good schema must be topically coherent, i.e., the relations and actors should relate to some real world topic or event. The tuples that comprise a schema should be valid assertions that make sense in the real world. Finally, each actor in the schema should belong to a cohesive set that plays a consistent role in the relations. Since there are no good automated ways to make such judgments, we perform a human evaluation using workers from Amazon's Mechanical Turk (AMT).

We compare Rel-grams schemas against the state-of-the-art narrative schemas released by Chambers (Chambers and Jurafsky, 2009).[8] Chambers' schemas are less expressive than ours – they do not associate types with actors and each schema has a constant pre-specified number of relations. For a fair comparison we use a similarly expressive version of our schemas that strips off argument types and has the same number of relations per schema (six) as their highest quality output set.

5.2.1 Evaluation Design

We created two tasks for AMT annotators. The first task tests the coherence and validity of relations in a schema and the second does the same for the schema actors. In order to make the tasks understandable to unskilled AMT workers, we followed the accepted practice of presenting them with *grounded* instances of the schemas (Wang et al., 2013), e.g., instantiating a schema with a specific argument instead of showing the various possibilities for an actor.

First, we collect the information in schemas as a set of tuples: $S = \{T_1, T_2, \cdots, T_n\}$, where each tuple is of the form $T : (X, Rel, Y)$, which conveys a relationship *Rel* between actors X and Y. Each actor is represented by its highest frequency examples (instances). Table 4 shows examples of schemas from Chambers and Rel-grams represented in this format. Then, we create grounded tuples by randomly sampling from top instances for each actor.

Task I: Topical Coherence To test whether the relations in a schema form a coherent topic or event, we presented the AMT annotators with a schema as a set of grounded tuples, showing each relation in the schema, but randomly selecting one of the top 5 instances from each actor. We generated five such

[8] Available at http://www.usna.edu/Users/cs/nchamber/data/schemas/acl09

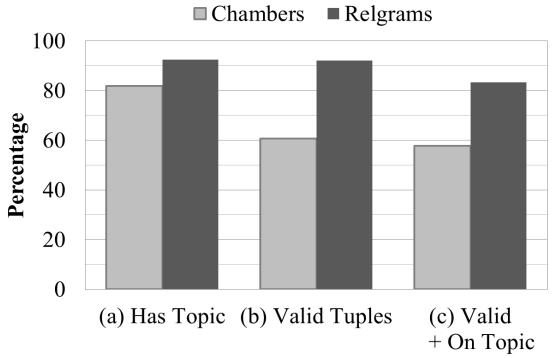


Figure 3: (a) Has Topic: Percentage of schema instantiations with a coherent topic. (b) Valid Tuples: Percentage of grounded statements that assert valid real-world relations. (c) Valid + On Topic: Percentage of grounded statements where 1) the instantiation has a coherent topic, 2) the tuple is valid and 3) the relation belongs to the common topic. All differences are statistically significant with a *p-value* < 0.01.

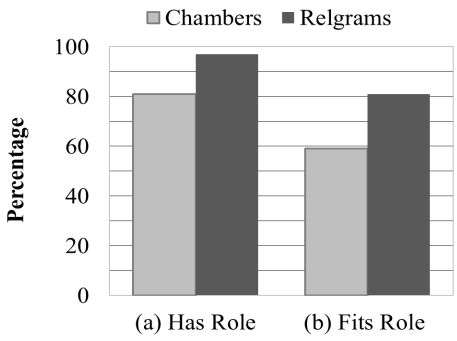


Figure 4: Actor Coherence: *Has Role* bars compare the percentage of tuples where the tested actors have a coherent role. *Fits Role* compares the percentage of top instances that fit the specified role for the tested actors. All differences are statistically significant with a *p-value* < 0.01.

instantiations for each schema. An example instantiation is shown in Table 5.

We ask three kinds of questions on each grounded schema: (1) is each of the grounded tuples valid (i.e. meaningful in the real world); (2) do the majority of relations form a coherent topic; and (3) does each tuple belong to the common topic. Similar to previous AMT studies we get judgments from multiple (five) annotators on each task and use the majority labels (Snow et al., 2008).

Our instructions specified that the annotators should ignore grammar and focus on whether a tuple may be interpreted as a real world statement. For example, the first tuple in R1 in Table 5 is a valid statement – "a bomb exploded in a city", but the tuples in C1 "a blast exploded a child", "a child detonated a blast", and "a child planted a blast" don't make sense.

Task II: Actor Coherence To test whether the instances of an actor form a coherent set, we held the relation and one actor fixed and presented the AMT annotators with the top 5 instances for the other actor. The first example R11 in Table 6 holds the relation "explode in" fixed, and A2 is grounded to the randomly selected instance "city". We present grounded tuples by varying A1 and ask annotators to judge whether these instances form a coherent topic and whether each instance belongs to that common topic. As with Task I, we create five random instantiations for each schema.

5.2.2 Results

We obtained a test set of 100 schemas per system by randomly sampling from the top 500 schemas from each system. We evaluate this test set using Task I and II as described above. For both tasks we obtained ratings from five turkers and use the majority labels as the final annotation.

Does the schema belong to a single topic? The *Has Topic* bars in Figure 3 show results for schema coherence. Rel-grams has a higher proportion of schemas with a coherent topic, 91% compared to 82% for Chambers'. This is a 53% reduction in incoherent schemas.

Do tuples assert valid real-world relations? The *Valid Tuples* bars in Figure 3 compare the percentage of valid grounded tuples in the schema instantiations. A tuple was labeled *valid* if a majority of the annotators labeled it to be meaningful in the real world. Here we see a dramatic difference – Rel-grams have 92% valid tuples, compared with Chambers' 61%.

What proportion of tuples belong? The *Valid + On Topic* bars in Figure 3 compare the percentage of tuples that are both *valid* and *on topic*, i.e., fits the main topic of the schema. Tuples from schema instantiations that did not have a coherent topic were labeled incorrect.

Rel-grams have a higher proportion of valid tuples belonging to a common topic, 82% compared to

58% for Chambers' schemas, a 56% error reduction. This is the strictest of the experiments described thus far – 1) the schema must have a topic, 2) the tuple must be valid, and 3) the tuple must belong to the topic.

Do actors represent a coherent set of arguments? We evaluated schema actors from the top 25 schemas in Chambers' and Rel-grams schemas, using grounded instances such as those in Table 6. Figure 4 compares the percentage of tuples where the actors play a coherent role (Has Role), and the percentage of instances that fit that role for the actor (Fits Role). Rel-grams has much higher actor coherence than Chambers', with 97% judged to have a topic compared to 81%, and 81% of instances fitting the common role compared with Chambers' 59%.

5.2.3 Error Analysis

The errors in both our schemas and those of Chambers are primarily due to mismatched actors and from extraction errors, although Chambers' schemas have a larger number of actor mismatch errors and the cause of the errors is different for each system.

Examining the data published by Chambers, the main source of invalid tuples are mismatch of subject and object for a given relation, which accounts for 80% of the invalid tuples. We hypothesize that this is due to the pair-wise representation that treats subject-verb and verb-object separately, causing inconsistent s-v-o tuples. An example is (boiler, light, candle) where (boiler, light) and (light, candle) are well-formed, yet the entire tuple is not. In addition, 43% of the invalid tuples seem to be from errors by the dependency parser.

Our schemas also suffer from mismatched actors, despite the semantic typing of the actors – we found a mismatch in 56% of the invalid tuples (5% of all tuples). A general type such as <person> or <organization> may still have an instance that does not play the same role as other instances. For example a relation (A1, graduated from, A2) has A2 that is mostly school names, but also includes "church" which leads to an invalid tuple.

Extraction errors account for 47% of the invalid tuples in our schemas, primarily errors that truncate an n-ary relation as a binary tuple. For example, the sentence "Mr. Diehl spends more time ... than the commissioner" is misanalysed by the Open IE extractor as (Mr. Diehl, spend than, commissioner).

6 Related Work

Prior work by Chambers and Jurafsky (2008; 2009; 2010) showed that event sequences (narrative chains) mined from text can be used to induce event schemas in a domain-independent fashion. However, our manual evaluation of their output showed key limitations which may limit applicability.

As pointed out earlier, a major weakness in Chambers' approach is the pair-wise representation of subject-verb and verb-object. Also, their released a set of schemas are limited to two actors, although this number can be increased by setting a chain splitting parameter.

Chambers and Jurafsky (2011) extended schema generation to learn domain-specific event templates and associated extractors. In work parallel to ours, Cheung et al. (2013), developed a probabilistic solution for template generation. However, their approach requires performing joint probability estimation using EM, which can limit scaling to large corpora.

In this work we developed an Open IE based solution to generate schemas. Following prior work (Balasubramanian et al., 2012), we use Open IE triples for modeling relation co-occurrence. We extend the triple representation with semantic types for arguments to alleviate sparisty and to improve coherence. We developed a page rank based schema induction algorithm which results in more coherent schemas with several actors. Unlike Chambers' approach this method does not require explicit parameter tuning for controlling the number of actors.

While our event schemas are close to being templates (because of associated types, and actor clustering), they do not have associated extractors. Our future work will focus on building extractors for these. It will also be interesting to compare with Cheung's system on smaller focused corpora.

Defining representations for events is a topic of active interest (Fokkens et al., 2013). In this work, we use a simpler representation, defining event schemas as a set of actors with associated types and a set of roles they play.

7 Conclusions

We present a system for inducing event schemas from text corpora based on Rel-grams, a language model derived from co-occurrence statistics of relational triples (Arg1, Relation, Arg2) extracted by a state-of-the-art Open IE system. By using triples rather than a pair-wise representation of subject-verb and verb-object, we achieve more coherent schemas than Chambers and Jurafsky (2009). In particular, our schemas have higher topic coherence (92% compared to Chambers' 82%; make a higher percentage of valid assertions (94% compared with 61%); and have greater actor coherence (81% compared with 59%).

Our schemas are also more expressive than those published by Chambers – we have semantic typing for the actors, we are not limited to two actors per schema, and our relation phrases include prepositions and are thus more precise and have higher coverage of actors involved in the event.

Our future plans are to build upon our event schemas to create an open-domain event extractor. This will extend each induced schema to have associated extractors. These extractors will operate on a document and instantiate an instance of the schema.

We have created a Rel-grams database with 1.1M entries and a set of over 2K event schemas from a corpus of 1.8M New York Times articles. Both are freely available to the research community[9] and may prove useful for a wide range of NLP applications.

Acknowledgments

We thank the anonymous reviewers, Tony Fader, and Janara Christensen for their valuable feedback. This paper was supported by Office of Naval Research (ONR) grant number N00014-11-1-0294, Army Research Office (ARO) grant number W911NF-13-1-0246, Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058, and Defense Advanced Research Projects Agency (DARPA) via AFRL contract number AFRL FA8750-13-2-0019. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ONR, ARO, IARPA, AFRL, or the U.S. Government.

[9] available at *http://relgrams.cs.washington.edu*

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–41.

ARPA. 1991. *Proc. 3rd Message Understanding Conf.* Morgan Kaufmann.

ARPA. 1998. *Proc. 7th Message Understanding Conf.* Morgan Kaufmann.

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. 2012. Rel-grams: a probabilistic model of relations in text. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 101–105. Association for Computational Linguistics.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117.

N. Chambers and D. Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*.

N. Chambers and D. Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL*.

N. Chambers and D. Jurafsky. 2010. A database of narrative schemas. In *Proceedings of LREC*.

N. Chambers and D. Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of ACL*.

J. Cheung, H. Poon, and L. Vandervende. 2013. Probabilistic frame induction. In *Proceedings of NAACL HLT*.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, , and R. Weischedel. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *Procs. of LREC*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Antske Fokkens, Marieke van Erp, Piek Vossen, Sara Tonelli, Willem Robert van Hage, BV SynerScope, Luciano Serafini, Rachele Sprugnoli, and Jesper Hoeksema. 2013. Gaf: A grounded annotation framework for events. *NAACL HLT 2013*, page 11.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *WWW*, pages 517–526.

Joseph Irwin, Mamoru Komachi, and Yuji Matsumoto. 2011. Narrative schema as world knowledge for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 86–92. Association for Computational Linguistics.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, (Just Accepted):1–54.

Mausam, Stephen Soderland, Oren Etzioni, Daniel Weld, Kobi Reiter, Michael Skinner, Marcus Sammer, and Jeff Bilmes. 2010. Panlingual lexical translation via probabilistic inference. *Artificial Intelligence Journal (AIJ)*.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of EMNLP*.

K. Owczarzak and H.T. Dang. 2010. Overview of the tac 2010 summarization track.

S. Patwardhan and E. Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of EMNLP 2009*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

A. Wang, C.D.V. Hoang, and M-Y. Kan. 2013. Perspectives on crowdsourcing annotations for Natural Language Processing. *Language Resources and Evaluation*, 47:9–31.

Orthonormal Explicit Topic Analysis for Cross-lingual Document Matching

John Philip McCrae
University Bielefeld
Inspiration 1
Bielefeld, Germany

Philipp Cimiano
University Bielefeld
Inspiration 1
Bielefeld, Germany

Roman Klinger
University Bielefeld
Inspiration 1
Bielefeld, Germany

{jmccrae,cimiano,rklinger}@cit-ec.uni-bielefeld.de

Abstract

Cross-lingual topic modelling has applications in machine translation, word sense disambiguation and terminology alignment. Multilingual extensions of approaches based on latent (LSI), generative (LDA, PLSI) as well as explicit (ESA) topic modelling can induce an interlingual topic space allowing documents in different languages to be mapped into the same space and thus to be compared across languages. In this paper, we present a novel approach that combines latent and explicit topic modelling approaches in the sense that it builds on a set of explicitly defined topics, but then computes latent relations between these. Thus, the method combines the benefits of both explicit and latent topic modelling approaches. We show that on a cross-lingual mate retrieval task, our model significantly outperforms LDA, LSI, and ESA, as well as a baseline that translates every word in a document into the target language.

1 Introduction

Cross-lingual document matching is the task of, given a *query* document in some source language, estimating the similarity to a document in some target language. This task has important applications in machine translation (Palmer et al., 1998; Tam et al., 2007), word sense disambiguation (Li et al., 2010) and ontology alignment (Spiliopoulos et al., 2007). An approach that has become quite popular in recent years for cross-lingual document matching is Explicit Semantics Analysis (ESA, Gabrilovich and Markovitch (2007)) and its cross-lingual extension CL-ESA (Sorg and Cimiano, 2008). ESA indexes documents by mapping them into a topic space defined by their similarity to predefined explicit topics – generally articles from an encyclopaedia – in such a way that there is a one-to-one correspondence between topics and encyclopedic entries. CL-ESA extends this to the multilingual case by exploiting a background document collection that is aligned across languages, such as Wikipedia. A feature of ESA and its extension CL-ESA is that, in contrast to latent (e.g. LSI, Deerwester et al. (1990)) or generative topic models (such as LDA, Blei et al. (2003)), it requires no training and, nevertheless, has been demonstrated to outperform LSI and LDA on cross-lingual retrieval tasks (Cimiano et al., 2009).

A key choice in Explicit Semantic Analysis is the document space that will act as the topic space. The standard choice is to regard all articles from a background document collection – Wikipedia articles are a typical choice – as the topic space. However, it is crucial to ensure that these topics cover the semantic space evenly and completely. In this paper, we present an alternative approach where we remap the semantic space defined by the topics in such a manner that it is orthonormal. In this way, each document is mapped to a topic that is distinct from all other topics. Such a mapping can be considered as equivalent to a variant of Latent Semantic Indexing (LSI) with the main difference that our model exploits the matrix that maps topic vectors back into document space, which is normally discarded in LSI-based approaches. We dub our model *ONETA* (OrthoNormal Explicit Topic Analysis) and empirically show that on a cross-lingual retrieval

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1732–1740,
Seattle, Washington, USA, 18-21 October 2013.

task it outperforms ESA, LSI, and Latent Dirichlet Allocation (LDA) as well as a baseline consisting of translating each word into the target language, thus reducing the task to a standard monolingual matching task. In particular, we quantify the effect of different approximation techniques for computing the orthonormal basis and investigate the effect of various methods for the normalization of frequency vectors.

The structure of the paper is as follows: we situate our work in the general context of related work on topic models for cross-lingual document matching in Section 2. We present our model in Section 3 and present our experimental results and discuss these results in Section 4.

2 Related Work

The idea of applying topic models that map documents into an interlingual topic space seems a quite natural and principled approach to tackle several tasks including the cross-lingual document retrieval problem.

Topic modelling is the process of finding a representation of a document d in a lower dimensional space $\mathbb{R}^K$ where each dimension corresponds to one *topic* that abstracts from specific words and thus allows us to detect deeper semantic similarities between documents beyond the computation of the pure overlap in terms of words.

Three main variants of document models have been mainly considered for cross-lingual document matching:

Latent methods such as Latent Semantic Indexing (LSI, Deerwester et al. (1990)) induce a decomposition of the term-document matrix in a way that reduces the dimensionality of the documents, while minimizing the error in reconstructing the training data. For example, in Latent Semantic Indexing, a term-document matrix is approximated by a partial singular value decomposition, or in Non-Negative Matrix Factorization (NMF, Lee and Seung (1999)) by two smaller non-negative matrices. If we append comparable or equivalent documents in multiple languages together before computing the decomposition as proposed by Dumais et al. (1997) then the topic model is essentially cross-lingual allowing to compare documents in different languages once they have been mapped into the topic space.

Probabilistic or generative methods instead attempt to induce a (topic) model that has the highest likelihood of generating the documents actually observed during training. As with latent methods, these topics are thus interlingual and can generate words/terms in different languages. Prominent representatives of this type of method are Probabilistic Latent Semantic Indexing (PLSI, Hofmann (1999)) or Latent Dirichlet Allocation (LDA, Blei et al. (2003)), both of which can be straightforwardly extended to the cross-lingual case (Mimno et al., 2009).

Explicit topic models make the assumption that topics are explicitly given instead of being induced from training data. Typically, a background document collection is assumed to be given whereby each document in this corpus corresponds to one topic. A mapping from document to topic space is calculated by computing the similarity of the document to every document in the topic space. A prominent example for this kind of topic modelling approach is Explicit Semantic Analysis (ESA, Gabrilovich and Markovitch (2007)).

Both latent and generative topic models attempt to find topics from the data and it has been found that in some cases they are equivalent (Ding et al., 2006). However, this approach suffers from the problem that the topics might be artifacts of the training data rather than coherent semantic topics. In contrast, explicit topic methods can use a set of topics that are chosen to be well-suited to the domain. The principle drawback of this is that the method for choosing such explicit topics by selecting documents is comparatively crude. In general, these topics may be overlapping and poorly distributed over the semantic topic space. By comparison, our method takes the advantage of the pre-specified topics of explicit topic models, but incorporates a training step to learn latent relations between these topics.

3 Orthonormal explicit topic analysis

Our approach follows Explicit Semantic Analysis in the sense that it assumes the availability of a background document collection $B = \{b_1, b_2, ..., b_N\}$ consisting of textual representations. The mapping into the explicit topic space is defined by a language-specific function Φ that maps documents into $\mathbb{R}^N$ such that the j^{th} value in the vector is given by some *association measure* $\phi_j(d)$ for each background document b_j. Typical choices for this association measure ϕ are the sum of the TF-IDF scores or an information retrieval relevance scoring function such as BM-25 (Sorg and Cimiano, 2010).

For the case of TF-IDF, the value of the j-th element of the topic vector is given by:

$$\phi_j(d) = \overrightarrow{\text{tf-idf}}(b_j)^{\text{T}} \ \overrightarrow{\text{tf-idf}}(d)$$

Thus, the mapping function can be represented as the product of a TF-IDF vector of document d multiplied by an $W \times N$ matrix, $\mathbf{X}$, each element of which contains the TF-IDF value of word i in document b_j:

$$\Phi(d) = \begin{pmatrix} \overrightarrow{\text{tf-idf}}(b_1)^{\text{T}} \\ \vdots \\ \overrightarrow{\text{tf-idf}}(b_N)^{\text{T}} \end{pmatrix} \overrightarrow{\text{tf-idf}}(d) = \mathbf{X}^{\text{T}} \cdot \overrightarrow{\text{tf-idf}}(d)$$

For simplicity, we shall assume from this point on that all vectors are already converted to a TF-IDF or similar numeric vector form. In order to compute the similarity between two documents d_i and d_j, typically the cosine-function (or the normalized dot product) between the vectors $\Phi(d_i)$ and $\Phi(d_j)$ is computed as follows:

$$\text{sim}(d_i, d_j) = \cos(\Phi(d_i), \Phi(d_j)) = \frac{\Phi(d_i)^{\text{T}}\Phi(d_j)}{||\Phi(d_i)||||\Phi(d_j)||}$$

If we represent the above using our above defined $W \times N$ matrix $\mathbf{X}$ then we get:

$$\text{sim}(d_i, d_j) = \cos(\mathbf{X}^{\text{T}}d_i, \mathbf{X}^{\text{T}}d_j) = \frac{d_i^{\text{T}}\mathbf{X}\mathbf{X}^{\text{T}}d_j}{||\mathbf{X}^{\text{T}}d_i||||\mathbf{X}^{\text{T}}d_j||}$$

The key challenge with ESA is choosing a good background document collection $B = \{b_1, ..., b_N\}$. A simple minimal criterion for a good background document collection is that each document in this collection should be maximally similar to itself and less similar to any other document:

$$\forall i \neq j \ \ 1 = \text{sim}(b_j, b_j) > \text{sim}(b_i, b_j) \geq 0$$

While this criterion is trivially satisfied if we have no duplicate documents in our collection, our intuition is that we should choose a background collection that maximizes the *slack margin* of this inequality, i.e. $|\text{sim}(b_j, b_j) - \text{sim}(b_i, b_j)|$. We can see that maximizing this margin for all i,j is the same as minimizing the *semantic overlap* of the background documents, which is given as follows:

$$\text{overlap}(B) = \sum_{\substack{i=1,\ldots,N \\ j=1,\ldots,N \\ i \neq j}} \text{sim}(b_i, b_j)$$

We first note that we can, without loss of generality, normalize our background documents such that $||Xb_j|| = 1$ for all j, and in this case we can redefine the semantic overlap as the following matrix expression[1]

$$\text{overlap}(\mathbf{X}) = ||\mathbf{X}^{\text{T}}\mathbf{X}\mathbf{X}^{\text{T}}\mathbf{X} - \mathbf{I}||_1$$

It is trivial to verify that this equation has a minimum when $\mathbf{X}^{\text{T}}\mathbf{X}\mathbf{X}^{\text{T}}\mathbf{X} = \mathbf{I}$. This is the case when the topics are *orthonormal*:

$$(\mathbf{X}^{\text{T}}b_i)^{\text{T}}(\mathbf{X}^{\text{T}}b_j) = 0 \quad \text{if } i \neq j$$

$$(\mathbf{X}^{\text{T}}b_i)^{\text{T}}(\mathbf{X}^{\text{T}}b_i) = 1$$

Unfortunately, this is not typically the case as the documents have significant word overlap as well as semantic overlap. Our goal is thus to apply a suitable transformation to $\mathbf{X}$ with the goal of ensuring that the orthogonality property holds.

Assuming that this transformation of $\mathbf{X}$ is done by multiplication with some other matrix $\mathbf{A}$, we can define the learning problem as finding that matrix $\mathbf{A}$ such that:

$$(\mathbf{A}\mathbf{X}^{\text{T}}\mathbf{X})^{\text{T}}(\mathbf{A}\mathbf{X}^{\text{T}}\mathbf{X}) = I$$

[1] $||\mathbf{A}||_p = \sum_{i,j} |a_{ij}|^p$ is the p-norm. $||\mathbf{A}||_{\mathcal{F}} = \sqrt{||\mathbf{A}||_2}$ is the Frobenius norm.

If we have the case that $W \geq N$ and that the rank of $\mathbf{X}$ is N, then $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is invertible and thus $\mathbf{A} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ is the solution to this problem.[2]

We define the projection function of a document d, represented as a normalized term frequency vector, as follows:

$$\Phi_{\mathrm{ONETA}}(d) = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}d$$

For the cross-lingual case we assume that we have two sets of background documents of equal size, $B^1 = \{b_1^1, \ldots, b_N^1\}$, $B^2 = \{b_1^2, \ldots, b_N^2\}$ in languages l_1 and l_2, respectively and that these documents are aligned such that for every index i, b_i^1 and b_i^2 are documents on the same topic in each language. Using this we can construct a projection function for each language which maps into the same topic space. Thus, as in CL-ESA, we obtain the cross-lingual similarity between a document d_i in language l_1 and a document d_j in language l_2 as follows:

$$\mathrm{sim}(d_i, d_j) = \cos(\Phi^{l_1}_{\mathrm{ONETA}}(d_i), \Phi^{l_2}_{\mathrm{ONETA}}(d_j))$$

We note here that we assume that Φ could be represented as a symmetric inner product of two vectors. However, for many common choices of association measures, including BM25, this is not the case. In this case the expression $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ can be replaced with a kernel matrix specifying the association of each background document to each other background document.

3.1 Relationship to Latent Semantic Indexing

In this section we briefly clarify the relationship between our method ONETA and Latent Semantic Indexing. Latent Semantic Indexing defines a mapping from a document represented as a term frequency vector to a vector in $\mathbb{R}^K$. This transformation is defined by means of calculating the singular value decomposition (SVD) of the matrix $\mathbf{X}$ as above, namely

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$$

Where $\mathbf{\Sigma}$ is diagonal and $\mathbf{U}$ $\mathbf{V}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{X}$., respectively. Let $\mathbf{\Sigma}_K$ denote the $K \times K$ submatrix containing the largest eigenvalues, and $\mathbf{U}_K$,$\mathbf{V}_K$ denote the corresponding eigenvectors. Thus LSI can be defined as:

$$\Phi_{\mathrm{LSI}}(d) = \mathbf{\Sigma}_K^{-1}\mathbf{U}_K d$$

With regards to orthonormalized topics, we see that using the SVD, we can simply derive the following:

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}} = \mathbf{V}\mathbf{\Sigma}^{-\mathbf{1}}\mathbf{U}^{\mathrm{T}}$$

When we set $K = N$ and thus choose the maximum number of topics, ONETA is equivalent to LSI modulo the fact that it multiplies the resulting topic vector by $\mathbf{V}$, thus projecting back into document space, i.e. into explicit topics.

In practice, both methods differ significantly in that the approximations they make are quite different. Furthermore, in the case that $W \gg N$ and $\mathbf{X}$ has n non-zeroes, the calculation of the SVD is of complexity $\mathcal{O}(nN + WN^2)$ and requires $\mathcal{O}(WN)$ bytes of memory. In contrast, ONETA requires computation time of $\mathcal{O}(N^a)$ for $a > 2$, which is the complexity of the matrix inversion algorithm[3], and only $\mathcal{O}(n + N^2)$ bytes of memory.

3.2 Approximations

The computation of the inverse has a complexity that, using current practical algorithms, is approximately cubic and as such the time spent calculating the inverse can grow very quickly. There are several methods for obtaining an approximate inverse. The most commonly used are based on the SVD or eigendecomposition of the matrix. As $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is symmetric positive definite, it holds that:

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$$

Where $\mathbf{U}$ are the eigenvectors of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{\Sigma}$ is a diagonal matrix of the eigenvalues. With $\mathbf{U}_K$,$\mathbf{\Sigma}_K$

[2] In the case that the matrix is not invertible we can instead solve $||\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{A} - \mathbf{I}||_{\mathcal{F}}$, which has a minimum at $\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^{\mathrm{T}}$ where $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ is the singular value decomposition of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

As usual we do not in fact compute the inverse for our experiments, but instead the LU Decomposition and solve by Gaussian elimination at test time.

[3] Algorithms with $a = 2.3727$ are known but practical algorithms have $a = 2.807$ or $a = 3$ (Coppersmith and Winograd, 1990)

as the first K eigenvalues and eigenvectors, respectively, we have:

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} \simeq \mathbf{U}_K \mathbf{\Sigma}_K^{-1} \mathbf{U}_K^{\mathrm{T}} \quad (1)$$

We call this the *orthonormal eigenapproximation* or *ON-Eigen*. The complexity of calculating $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$ from this is $\mathcal{O}(N^2K + Nn)$, where n is the number of non-zeros in $\mathbf{X}$.

Similarly, using the formula derived in the previous section we can derive an approximation of the full model as follows:

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}} \simeq \mathbf{U}_K \mathbf{\Sigma}_K^{-1} \mathbf{V}_K^{\mathrm{T}} \quad (2)$$

We call this approximation *Explicit LSI* as it first maps into the latent topic space and then into the explicit topic space.

We can consider another approximation by noticing that $\mathbf{X}$ is typically very sparse and moreover some rows of $\mathbf{X}$ have significantly fewer non-zeroes than others (these rows are for terms with low frequency). Thus, if we take the first N_1 columns (documents) in $\mathbf{X}$, it is possible to rearrange the rows of $\mathbf{X}$ with the result that there is some W_1 such that rows with index greater than W_1 have only zeroes in the columns up to N_1. In other words, we take a subset of N_1 documents and enumerate the words in such a way that the terms occurring in the first N_1 documents are enumerated $1, \ldots, W_1$. Let $N_2 = N - N_1$, $W_2 = W - W_1$. The result of this row permutation does not affect the value of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and we can write the matrix $\mathbf{X}$ as:

$$\mathbf{X} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{C} \end{pmatrix}$$

where $\mathbf{A}$ is a $W_1 \times N_1$ matrix representing term frequencies in the first N_1 documents, $\mathbf{B}$ is a $W_1 \times N_2$ matrix containing term frequencies in the remaining documents for terms that are also found in the first N_1 documents, and $\mathbf{C}$ is a $W_2 \times N_2$ containing the frequency of all terms not found in the first N_1 documents.

Application of the well-known divide-and-conquer formula (Bernstein, 2005, p. 159) for matrix inversion yields the following easily verifiable matrix identity, given that we can find $\mathbf{C}'$ such that $\mathbf{C}'\mathbf{C} = \mathbf{I}$.

$$\begin{pmatrix} (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}} & -(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{B}\mathbf{C}' \\ \mathbf{0} & \mathbf{C}' \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{C} \end{pmatrix} = \mathbf{I} \quad (3)$$

We denote the above equation using a matrix $\mathbf{L}$ as $\mathbf{L}^{\mathrm{T}}\mathbf{X} = \mathbf{I}$. We note that $\mathbf{L} \neq (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}$, but for any document vector d that is representable as a linear combination of the background document set (i.e., columns of $\mathbf{X}$) we have that $\mathbf{L}d = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}d$ and in this sense $\mathbf{L} \simeq (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$.

We further relax the assumption so that we only need to find a $\mathbf{C}'$ such that $\mathbf{C}'\mathbf{C} \simeq \mathbf{I}$. For this, we first observe that $\mathbf{C}$ is very sparse as it contains only terms not contained in the first N_1 documents and we notice that very sparse matrices tend to be approximately orthogonal, hence suggesting that it should be very easy to find a left-inverse of $\mathbf{C}$. The following lemma formalizes this intuition:

Lemma: If $\mathbf{C}$ is a $W \times N$ matrix with M non-zeros, distributed randomly and uniformly across the matrix, and all the non-zeros are 1, then $\mathbf{D}\mathbf{C}^{\mathrm{T}}\mathbf{C}$ has an expected value on each non-diagonal value of $\frac{M}{N^2}$ and a diagonal value of 1 if $\mathbf{D}$ is the diagonal matrix whose values are given by $||c_i||^{-2}$, the square of the norm of the corresponding column of $\mathbf{C}$.

Proof: We simply observe that if $\mathbf{D}' = \mathbf{D}\mathbf{C}^{\mathrm{T}}\mathbf{C}$, then the $(i,j)^{\text{th}}$ element of $\mathbf{D}'$ is given by

$$d_{ij} = \frac{c_i^{\mathrm{T}} c_j}{||c_i||^2}$$

If $i \neq j$ then the $c_i^{\mathrm{T}} c_j$ is the number of non-zeroes overlapping in the i^{th} and j^{th} column of $\mathbf{C}$ and under a uniform distribution we expect this to be $\frac{M^2}{N^3}$. Similarly, we expect the column norm to be $\frac{M}{N}$ such that the overall expectation is $\frac{M}{N^2}$. The diagonal value is clearly equal to 1.■

As long as $\mathbf{C}$ is very sparse, we can use the following approximation, which can be calculated in $\mathcal{O}(M)$ operations, where M is the number of non-zeroes.

$$\mathbf{C}' \simeq \begin{pmatrix} ||c_1||^{-2} & & 0 \\ & \ddots & \\ 0 & & ||c_{N_2}||^{-2} \end{pmatrix} \mathbf{C}^{\mathrm{T}}$$

We call this method *L-Solve*. The complexity of calculating a left-inverse by this method is of

Frequency Normalization	Document Normalization: No	Document Normalization: Yes
TF	0.31	0.78
Relative	0.23	0.42
TFIDF	0.21	0.63
SQRT	0.28	0.66

Table 1: Effect of Term Frequency and Document Normalization on Top-1 Precision

order $\mathcal{O}(N_1^a)$, being much more efficient than the eigenvalue methods. However, it is potentially more error-prone as it requires that a left-inverse of $\mathbf{C}$ exists. On real data this might be violated if we do not have linear independence of the rows of $\mathbf{C}$, for example if $W_2 < N_2$ or if we have even one document which has only words that are also contained in the first N_1 documents and hence there is a row in $\mathbf{C}$ that consists of zeros only. This can be solved by removing documents from the collection until $\mathbf{C}$ is row-wise linear independent.[4]

3.3 Normalization

A key factor in the effectiveness of topic-based methods is the appropriate normalization of the elements of the document matrix $\mathbf{X}$. This is even more relevant for orthonormal topics as the matrix inversion procedure can be very sensitive to small changes in the matrix. In this context, we consider two forms of normalization, term and document normalization, which can also be considered as row/column normalizations of $\mathbf{X}$.

A straightforward approach to normalization is to normalize each column of $\mathbf{X}$ to obtain a matrix as follows:

$$\mathbf{X}' = \left(\frac{x_1}{||x_1||} \cdots \frac{x_N}{||x_N||} \right)$$

If we calculate ${\mathbf{X}'}^{\mathrm{T}}\mathbf{X}' = \mathbf{Y}$ then we get that the (i, j)-th element of $\mathbf{Y}$ is:

$$y_{ij} = \frac{x_i^{\mathrm{T}} x_j}{||x_i||||x_j||}$$

[4]In the experiments in the next section we discarded 4.2% of documents at $N_1 = 1000$ and 47% of documents at $N_1 = 5000$

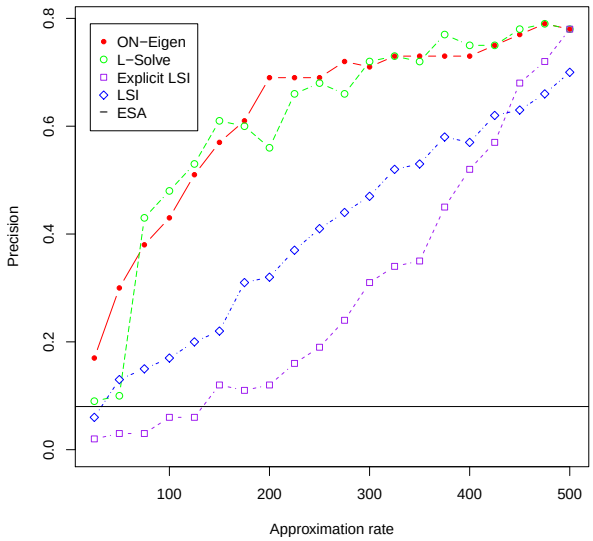


Figure 1: Effect on Top-1 Precision by various approximation method

Thus, the diagonal of $\mathbf{Y}$ consists of ones only and due to the Cauchy-Schwarz inequality we have that $|y_{ij}| \leq 1$, with the result that the matrix $\mathbf{Y}$ is already close to $\mathbf{I}$. Formally, we can use this to state a bound on $||{\mathbf{X}'}^{\mathrm{T}}\mathbf{X}' - \mathbf{I}||_{\mathcal{F}}$, but in practice it means that the orthonormalizing matrix has more small or zero values.

A further option for normalization is to consider some form of term frequency normalization. For term frequency normalization, we use *TF* (tf_{wn}), *Relative* ($\frac{\mathrm{tf}_{wn}}{F_w}$), *TFIDF* ($\mathrm{tf}_{wn} \log(\frac{N}{\mathrm{df}_w})$), and *SQRT* ($\frac{\mathrm{tf}_{wn}}{\sqrt{F_w}}$). Here, tf_{wn} is the term frequency of word w in document n, F_w is the total frequency of word w in the corpus, and df_w is the number of documents containing the words w. The first three of these normalizations have been chosen as they are widely used in the literature. The SQRT normalization has been shown to be effective for explicit topic methods in previous experiments not reported here.

4 Experiments and Results

For evaluation, we consider a cross-lingual mate retrieval task from English/Spanish on the basis of Wikipedia as aligned corpus. The goal is to, for each document of a test set, retrieve the aligned document or *mate*. For each test document, on the basis of

Method	Top-1 Prec.	Top-5 Prec.	Top-10 Prec.	MRR	Time	Memory
ONETA L-Solve ($N_1 = 1000$)	0.290	0.501	0.596	0.390	73s	354MB
ONETA L-Solve ($N_1 = 2000$)	0.328	0.531	0.600	0.423	2m18s	508MB
ONETA L-Solve ($N_1 = 3000$)	0.462	0.662	0.716	0.551	4m12s	718MB
ONETA L-Solve ($N_1 = 4000$)	0.599	0.755	0.781	0.667	7m44s	996MB
ONETA L-Solve ($N_1 = 5000$)	0.695	0.817	0.843	0.750	12m28s	1.30GB
ONETA L-Solve ($N_1 = 6000$)	0.773	0.883	0.905	0.824	18m40s	1.69GB
ONETA L-Solve ($N_1 = 7000$)	0.841	0.928	0.937	0.881	26m31s	2.14GB
ONETA L-Solve ($N_1 = 8000$)	0.896	0.961	0.968	0.927	37m39s	2.65GB
ONETA L-Solve ($N_1 = 9000$)	0.924	0.981	0.987	0.950	52m52s	3.22GB
ONETA (No Approximation)	**0.929**	**0.987**	**0.990**	**0.956**	57m10s	3.42GB
Word Translation	0.751	0.884	0.916	0.812	n/a	n/a
ESA (SQRT Normalization)	0.498	0.769	0.835	0.621	72s	284MB
LDA (K=1000)	0.287	0.568	0.659	0.417	4h12m	8.4GB
LSI (K=4000)	0.615	0.756	0.783	0.676	13h51m	19.7GB
ONETA + Word Translation	**0.932**	**0.987**	**0.993**	**0.958**	n/a	n/a

Table 2: Result on large-scale mate-finding studies for English to Spanish matching

the similarity of the query document to all indexed documents, we compute the value rank_i indicating at which position the mate of the i^{th} document occurs. We use two metrics: *Top-k Precision*, defined as the percentage of documents for which the mate is retrieved among the first k elements, and *Minimum Reciprocal Rank*, defined as

$$\mathrm{MRR} = \sum_{i \in \mathrm{test}} \frac{1}{\mathrm{rank}_i}$$

For our experiments, we first extracted a subset of documents (every 20th) from Wikipedia, filtering this set down to only those that have aligned pages in both English and Spanish with a minimum length of 100 words. This gives us 10,369 aligned documents in total, which form the background document collection B. We split this data into a training and test set of 9,332 and 1,037 documents, respectively. We then removed all words whose total frequencies were below 50. This resulted in corpus of 6.7 millions words in English and 4.2 million words in Spanish.

Normalization Methods: In order to investigate the impact of different normalization methods, we ran small-scale experiments using the first 500 documents from our dataset to train ONETA and then evaluate the resulting models on the mate-finding task on 100 unseen documents. The results are presented in Table 1, which shows the Top-1 Precision for the different normalization methods. We see that the effect of applying document normalization in all cases improves the quality of the overall result. Surprisingly, we do not see the same result for frequency normalization yielding the best result for the case where we do no normalization at all[5] . In the remaining experiments we thus employ document normalization and no term frequency normalization.

Approximation Methods: In order to evaluate the different approximation methods, we experimentally compare 4 different approximation methods: standard LSI, ON-Eigen (Equation 1), Explicit LSI (Equation 2), L-Solve (Equation 3) on the same small-scale corpus. For convenience we plot an *approximation rate* which is either K or N_1 depending on method; at $K = 500$ and $N_1 = 500$, these approximations become exact. This is shown in Figure 1. We also observe the effects of approximation and see that the performance increases steadily as we increase the computational factor. We see that the orthonormal eigenvector (Equation 1) method and the L-solve (Equation 3) method are clearly similar in approximation quality. We see that the explicit LSI method (Equation 2) and the LSI method both perform significantly worse for most of the approxi-

[5] A likely explanation for this is that low frequency terms are less evenly distributed and the effect of calculating the matrix inverse magnifies the noise from the low frequency terms

mation amounts. Explicit LSI is worse than the other approximations as it first maps the test documents into a K-dimensional LSI topic space, before mapping back into the N-dimensional explicit space. As expected this performs worse than standard LSI for all but high values of K as there is significant error in both mappings. We also see that the (CL-)ESA baseline, which is very low due to the small number of documents, is improved upon by even the least approximation of orthonormalization. In the remaining of this section, we report results using the L-Solve method as it has a very good performance and is computationally less expensive than ON-Eigen.

Evaluation and Comparison: We compare ONETA using the L-Solve method with N_1 values from 1000 to 9000 topics with (CL-)ESA (using SQRT normalization), LDA (using 1000 topics) and LSI (using 4000 topics). We choose the largest topic count for LSI and LDA we could to provide the best possible comparison. For LSI, the choice of K was determined on the basis of operating system memory limits, while for LDA we experimented with higher values for K without any performance improvement, likely due to overfitting. We also stress that for L-Solve ONETA, N_1 is not the topic count but an approximation rate of the mapping. In all settings we use N topics as with standard ESA, and so should not be considered directly comparable to the K values of these methods.

We also compare to a baseline system that relies on word-by-word translation, where we use the most likely single translation of a word as given by a phrase table generated by the Moses system (Koehn et al., 2007) on the EuroParl corpus (Koehn, 2005). Top 1, Top 5 and Top 10 Precision as well as Mean Reciprocal Rank are reported in Table 2.

Interestingly, even for a small number of documents (e.g., $N_1 = 6000$) our results improve both the word-translation baseline as well as all other topic models, ESA, LDA and LSI in particular. We note that at this level the method is still efficiently computable and calculating the inverse in practice takes less time than training the Moses system. The significance for results ($N_1 \geq 7000$) have been tested by means of a bootstrap resampling significance test, finding out that our results significantly improve on the translation base line at a 99% level.

Further, we consider a straightforward combination of our method with the translation system consisting of appending the topic vectors and the translation frequency vectors, weighted by the relative average norms of the vectors. We see that in this case the translations continue to improve the performance of the system (albeit not significantly), suggesting a clear potential for this system to help in improving machine translation results. While we have presented results for English and Spanish here, similar results were obtained for the German and French case but are not presented here due to space limitations.

In Table 2 we also include the user time and peak resident memory of each of these processes, measured on an 8 Core Intel Xeon 2.50 GHz server. We do not include the results for Word Translation as many hours were spent learning a phrase table, which includes translations for many phrases not in the test set. We see that the ONETA method significantly outperforms LSI and LDA in terms of speed and memory consumption. This is in line with the theoretical calculations presented earlier where we argued that inverting the $N \times N$ dense matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ when $W \gg N$ is computationally lighter than finding an eigendecomposition of the $W \times W$ sparse matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$. In addition, as we do not multiply $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ and $\mathbf{X}^{\mathrm{T}}$, we do not need to allocate a large $W \times K$ matrix in memory as with LSI and LDA.

The implementations of ESA, ONETA, LSI and LDA used as well as the data for the experiments are available at `http://github.com/jmccrae/oneta`.

5 Conclusion

We have presented a novel method for cross-lingual topic modelling, which combines the strengths of explicit and latent topic models and have demonstrated its application to cross-lingual document matching. We have in particular shown that the method outperforms widely used topic models such as Explicit Semantic Analysis (ESA), Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA). Further, we have shown that it outperforms a simple baseline relying on word-by-word translation of the query document into the target language,

while the induction of the model takes less time than training the machine translation system from a parallel corpus. We have also presented an effective approximation method, i.e. L-Solve, which significantly reduces the computational cost associated with computing the topic models.

Acknowledgements

This work was funded by the Monnet Project and the Portdial Project under the EC Seventh Framework Programme, Grants No. 248458 and 296170. Roman Klinger has been funded by the "Its OWL" project ("Intelligent Technical Systems Ostwestfalen-Lippe", `http://www.its-owl.de/`), a leading-edge cluster of the German Ministry of Education and Research.

References

Dennis S Bernstein. 2005. *Matrix mathematics, 2nd Edition*. Princeton University Press Princeton.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Philipp Cimiano, Antje Schultz, Sergej Sizov, Philipp Sorg, and Steffen Staab. 2009. Explicit versus latent concept models for cross-language information retrieval. In *IJCAI*, volume 9, pages 1513–1518.

Don Coppersmith and Shmuel Winograd. 1990. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Chris Ding, Tao Li, and Wei Peng. 2006. NMF and PLSI: equivalence and a hybrid algorithm. In *Proceedings of the 29th annual international ACM SIGIR*, pages 641–642. ACM.

Susan T Dumais, Todd A Letsche, Michael L Littman, and Thomas K Landauer. 1997. Automatic cross-language retrieval using latent semantic indexing. In *AAAI spring symposium on cross-language text and speech retrieval*, volume 15, page 21.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, volume 6, page 12.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference*, pages 50–57. ACM.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1138–1147. Association for Computational Linguistics.

David Mimno, Hanna M Wallach, Jason Naradowsky, David A Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889. Association for Computational Linguistics.

Martha Palmer, Owen Rambow, and Alexis Nasr. 1998. Rapid prototyping of domain-specific machine translation systems. In *Machine Translation and the Information Soup*, pages 95–102. Springer.

Philipp Sorg and Philipp Cimiano. 2008. Cross-lingual information retrieval with explicit semantic analysis. In *Proceedings of the Cross-language Evaluation Forum 2008*.

Philipp Sorg and Philipp Cimiano. 2010. An experimental comparison of explicit semantic analysis implementations for cross-language retrieval. In *Natural Language Processing and Information Systems*, pages 36–48. Springer.

Vassilis Spiliopoulos, George A Vouros, and Vangelis Karkaletsis. 2007. Mapping ontologies elements using features in a latent space. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 457–460. IEEE.

Yik-Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual LSA-based adaptation for statistical machine translation. *Machine Translation*, 21(4):187–207.

Automated Essay Scoring by Maximizing Human-machine Agreement

Hongbo Chen and Ben He
School of Computer and Control Engineering
University of Chinese Academy of Sciences
100190 Beijing, China
chenhongbo11@mails.ucas.ac.cn, benhe@ucas.ac.cn

Abstract

Previous approaches for automated essay scoring (AES) learn a rating model by minimizing either the classification, regression, or pairwise classification loss, depending on the learning algorithm used. In this paper, we argue that the current AES systems can be further improved by taking into account the agreement between human and machine raters. To this end, we propose a rank-based approach that utilizes listwise learning to rank algorithms for learning a rating model, where the agreement between the human and machine raters is directly incorporated into the loss function. Various linguistic and statistical features are utilized to facilitate the learning algorithms. Experiments on the publicly available English essay dataset, Automated Student Assessment Prize (ASAP), show that our proposed approach outperforms the state-of-the-art algorithms, and achieves performance comparable to professional human raters, which suggests the effectiveness of our proposed method for automated essay scoring.

1 Introduction

Automated essay scoring utilizes the NLP techniques to automatically rate essays written for given prompts, namely, essay topics, in an educational setting (Dikli, 2006). Nowadays, AES systems have been put into practical use in large-scale English tests and play the role of one human rater. For example, before AES systems enter the picture, essays in the writing assessment of Graduate Record Examination (GRE) are rated by two human raters. A third human rater is needed when the difference of the scores given by the two human raters is larger than one in the 6-point scale. Currently, GRE essays are rated by one human rater and one AES system. A second human rater is required only when there exists a non-negligible disagreement between the first human rater and the machine rater. With the help of an AES system that highly agrees with human raters, the human workload can be reduced by half at most. Therefore, the agreement between the AES system and the human rater is an important indicator of an AES system's effectiveness.

There have been efforts in developing AES methods since the 1960s. Various kinds of algorithms and models based on NLP and machine learning techniques have been proposed to implement AES systems. Existing approaches consider essay rating as a classification (Larkey, 1998), regression (Attali and Burstein, 2006) or preference ranking problem (Yannakoudakis et al., 2011), where the loss function is the regression loss, classification loss and pairwise classification loss, respectively. In this paper, we argue that the purpose of AES is to predict the essay's rating that human raters would give. If an AES system frequently disagrees with the first human rater, a second human rater will be needed in most cases. Thus, the introduction of the AES system does not bring much benefit in reducing the human workload. It is therefore desirable to minimize the disagreement between the machine and human raters. However, this disagreement is not explicitly, if any, addressed in the current AES methods.

To this end, we propose a rank-based approach in this paper that utilizes a listwise learning to rank

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1741–1752,
Seattle, Washington, USA, 18-21 October 2013.

algorithm to address automated essay scoring in the view of directly optimizing the agreement between human raters and the AES system. Different from the preference ranking-based approach (Yannakoudakis et al., 2011) that maximizes the pairwise classification precision (Liu, 2009), our rank-based approach follows the listwise learning paradigm and the agreement between the machine and human raters is directly integrated into the loss function that is optimized by gradient boost regression trees.

To the best of our knowledge, this work is the first to apply listwise learning to rank approach for AES, which aims at the optimization of the agreement between the human and machine raters. Experimental results on the publicly available dataset ASAP indicate that our proposed method achieves high agreement with human raters, that is about 0.80, measured by quadratic weighted Kappa (Brenner and Kliebsch, 1996). Our proposed method also outperforms the previous classification, regression and preference ranking based approaches. As it is widely accepted that the agreement between human raters, measured by either quadratic weighted Kappa or Pearson's correlation coefficient, ranges from 0.70 to 0.80 (Powers et al., 2000) (Williamson, 2009), our proposed approach therefore performs as well as human raters.

The rest of this paper is organized as follows. In section 2, we introduce the research background of automated essay scoring and give a brief introduction to learning to rank. In section 3, a detailed description of our listwise learning to rank approach for automated essay scoring is presented. Section 4 explains the experimental setup and section 5 presents the experimental results. Finally, in section 6 we conclude this research.

2 Related Work and Background

Firstly, we give a brief description of existing approaches for AES in section 2.1. Then, an introduction to learning to rank is presented in section 2.2.

2.1 Existing AES Methods

In general, existing solutions consider AES as a learning problem. Based on a large number of predefined objectively measurable features, various learning techniques, including classification, regression and preference ranking, are applied (Larkey, 1998) (Yannakoudakis et al., 2011).

Regression-based approach treats feature values and essay score as independent variables and dependent variable, respectively, and then learns a regression equation by classical regression algorithms, such as support vector regression (Vapnik et al., 1996). In 1966, the first AES system, Project Essay Grader, was developed by Ellis Page upon the request of the American College Board. The PEG system defines a large set of surface text features from essays, e.g. fourth root of essay length, and uses regression-based approach to predict the score that human raters will give. E-rater, developed by Educational Testing Services (ETS) in America, in late 1990s, is a commercial AES system which has been put into practical use in the Graduate Record Examination (GRE) and the Test of English as a Foreign Language (TOEFL). The E-rater system uses natural language processing techniques to extract various kinds of linguistic features of essays, such as lexical, syntactic, grammar, etc.. Then it predicts the final score by the stepwise regression method (Attali and Burstein, 2006).

The classification-based approach sees essay scores as in-discriminative class labels and uses classical classification algorithms, e.g. the K-nearest neighbor (KNN) and the naive Bayesian model, to predict to which class an essay belongs, where a class is associated to a numeric rating. Intelligent Essay Assessor (IEA) (Foltz et al., 1999), developed also in late 1990s, evaluates essay by measuring semantic features. Each ungraded essay, represented by a semantic vector generated by Latent Semantic Analysis (LSA) (Dumais, 2005), is rated according to the similarity degree with semantic vectors of graded essays. Bayesian Essay Test Scoring sYstem, developed by Larkey in 2003, is based on naive Bayesian model. It is the only open-source AES system, but has not been put into practical use yet.

Besides classification and regression-based approaches, (Yannakoudakis et al., 2011) proposed a preference ranking based approach for learning a rating model, where a ranking function or model is learned to construct a global ordering of essays based on writing quality. It is also the first study of rank-based approach in automated essay scoring. Although "learning to rank" is not mentioned in

their paper, the algorithm they used, Ranking SVM (svm-light package with "-z p" option), is actually a pairwise approach. We will give a brief introduction to learning to rank in section 2.2.

The AES systems can be deployed in two different manners, namely prompt-specific and generic. A prompt-specific rating model is built for a specific prompt and designed to be the best rating model for the particular prompt (Williamson, 2009). For different prompts, the features used, their weights, and scoring criteria, may be different. It usually requires several hundreds of graded essays for training, which is time-consuming and usually impractical in a classroom environment. Generic rating model is trained from essays across a group of prompts and designed to be the best fit for predicting human scores for all prompts. It usually does not consider prompt-specific features and just takes writing quality into account. Generic rating model evaluates essays across all prompts with the same scoring criteria, which is more consistent with the human rubric that is usually the same for all prompts, and therefore has validity-related advantages (Attali et al., 2010).

2.2 Learning to Rank

Learning to rank, also called machine-learned ranking, was originally proposed to settle the ranking problem in information retrieval (IR) (Liu, 2009). It is a type of supervised or semi-supervised machine learning algorithm that automatically construct a ranking model or function from training data.

Current learning to rank algorithms fall into three categories, that is, the pointwise, pairwise, listwise approaches. Pointwise approach takes individual documents as training examples for learning a scoring function. In fact, both multiple linear regression and support vector regression (Vapnik et al., 1996), which have been widely used in automated essay scoring (Shermis and Burstein, 2002), can be seen as pointwise approaches. Pairwise approaches process a pair of documents each time and usually model ranking as a pairwise classification problem. Thus, the loss function is always a classification loss. Representative algorithms are ranking SVM (Joachims, 2006), RankNet (Li et al., 2007), etc.. (Yannakoudakis et al., 2011) apply pairwise approach, ranking SVM, to automated essay scoring and achieve better performance than support vector regression. In listwise approaches, ranking algorithms process a list of documents each time and the loss function aims at measuring the accordance between predicted ranking list and the ground truth label. Representative algorithms are LambdaMart (Wu et al., 2008), RankCosine (Qin et al., 2008), etc.. Listwise approach has not yet been used in automated essay scoring.

3 Automated Essay Scoring by Maximizing Human-machine Agreement

The main work-flow of our proposed approach is as follows. Firstly, a set of essays rated by professional human raters are gathered for the training. A listwise learning to rank algorithm learns a ranking model or function using this set of human rated essays represented by vectors of the pre-defined features. Then the learned ranking model or function outputs a model score for each essay, including both rated and unrated essays, from which a global ordering of essays is constructed. Finally, the model score is mapped to a predefined scale of valid ratings, such as an integer from 1 to 6 in a 6-point scale

In this section, we give a detailed description of our listwise learning to rank approach for AES in section 3.1. And features used in our approach are presented in section 3.2.

3.1 Listwise Learning to Rank for AES

Our choice of the listwise learning to rank algorithm is due to the fact that it takes the entire set of labeled essays associated to a given prompt, instead of the individual essays or essay pairs as in (Yannakoudakis et al., 2011), as training examples. This brings us the convenience of easily embedding the inter-rater agreement into the loss function for the learning.

In this paper, we deploy LambdaMART (Wu et al., 2008), a listwise learning to rank algorithm and then use Random Forests (RF) (Breiman, 2001) for the bagging of LambdaMART learners. Having been widely used in information retrieval applications, LambdaMART is one of the most effective learning to rank algorithms. For instance, it achieves the top results in the 2010 Yahoo! Learning to Rank challenge (Burges, 2010). Random Forests is an

ensemble learning method for classification and regression.

Previously, the loss function of LambdaMART is defined as the gradient loss of the retrieval effectiveness, measured by IR evaluation criteria such as Normalized Discounted Cumulative Gain (nDCG) (Wu et al., 2008). More specifically, it is a heuristic method that directly defines λ, the gradient of nDCG with respect to the model score of each document, and has been shown to work empirically for particular loss functions NDCG (Yue and Burges, 2007). Then, Multiple Additive Regression Trees (MART) (Friedman, 2000), also called Gradient Boosting Decision Tree (GBDT)[1], are used to "learn" these gradients iteratively. MART is a class of boosting algorithms that performs gradient descent in function space, using regression trees. Its output $F(x)$ can be written as $F(x) = \sum_i \alpha_i f_i(x), i = 1, 2, ...N$. Each $f_i(x)$ is a function modeled by a single regression tree and the α_i is the corresponding weight. Given that *n* trees have been trained, the *(n+1)*th regression tree, $f_{i+1}(x)$, models the derivative of the cost with respect to the current model score at each training point. Thus, what remains is to compute the derivative.

As for the automated essay scoring, LambdaMART is not readily available since its loss function is defined as a function of the gradient of IR evaluation measures. While such measures focus on the top-ranked documents that are of great importance to the IR applications, they are not suitable to our study. This is because for AES, the rating prediction of all essays equally matters, no matter what ratings they receive.

It is therefore necessary to re-define the λ. Specifically, we need to define the gradient of the evaluation criteria in AES, e.g. quadratic weighted Kappa (Brenner and Kliebsch, 1996) and Pearson's correlation coefficient, with respect to the model score of each essay. In this paper, we use quadratic weighted Kappa as the evaluation metric. Kappa (Cohen and others, 1960) is a statistical metric which is used to measure inter-rater agreement. Quadratic weighted Kappa takes the degree of disagreement between raters into account. This measuring method is widely accepted as a primary evaluation metric for the AES tasks. For instance, it is the official evaluation metric in the Automated Student Assessment Prize sponsored by Hewlett Foundation[2]. We denote our modified LambdaMART as K-LambdaMART in which K stands for the Kappa-based gradient function. Specific steps include the following:

To begin with, we re-define the $\lambda_{i,j}$ for each pair of essays. For a pair of essays, essay i and essay j, $\lambda_{i,j}$ is defined as the derivative of RankNet (Li et al., 2007) loss function multiplied by the Quadratic weighted Kappa gain after exchanging the two essays' ratings.

$$\lambda_{i,j} = \frac{-\delta}{1 + e^{\delta(s_i - s_j)}} |\Delta_{Kappa}| \tag{1}$$

s_i and s_j are the model scores for essay i and essay j, respectively. δ is a parameter which determines the shape of the sigmoid. Quadratic weighted Kappa are calculated as follows:

$$\kappa = 1 - \frac{\sum_{i,j} \omega_{i,j} O_{i,j}}{\sum_{i,j} \omega_{i,j} E_{i,j}} \tag{2}$$

In matrix O, $O_{i,j}$ corresponds to the number of essays that received a score i by human rater and a score j by the AES system. In matrix ω, $\omega_{i,j}$ is the difference between raters scores $\frac{(i-j)^2}{(N-1)^2}$, where N is the number of possible ratings. Matrix E is calculated as the outer product between the two raters vectors of scores, normalized such that E and O have the same sum.

It is necessary to define the quadratic weighted Kappa gain, namely Δ_{Kappa}, in an explicit manner. In each iteration, every essay is ranked by its model score and then rated according to its ranking position. For example, for five essays e_1, e_2, e_3, e_4, e_5 with actual ratings $5, 4, 3, 2, 1$, if the ranking (by model score) is e_3, e_4, e_1, e_5, e_2, we assume that e_3, e_4, e_1, e_5, e_2 will get ratings of $5, 4, 3, 2, 1$, over which quadratic weighted kappa gain can be calculated.

After the definition of $\lambda_{i,j}$ for each pair of essays, it is time to re-define the λ, the gradient for each essay. Let I denote the set of pairs of indices $\langle i, j \rangle$, in which essay i receive a higher rating than essay j.

[1]For space reason, we refer the readers to (Friedman, 2000), (Breiman, 2001) for details of MART, GBDT and Random Forests.

[2]http://www.kaggle.com/c/asap-sas

Set I must include each pair just once. Then, the λ gradient for each essay, e.g. essay i, is defined as,

$$\lambda_i = \sum_{j:\langle i,j\rangle \in I} \lambda_{i,j} - \sum_{j:\langle j,i\rangle \in I} \lambda_{i,j}; \qquad (3)$$

The rational behind the above formulae is as follows. For each of the essays in the whole essay collection associated with the same prompt, e.g. essay i, the gradient λ_i is incremented by a positive value $\lambda_{i,j}$ when coming across another essay j that has a lower rating. The value of $\lambda_{i,j}$ is weighted by the quadratic weighted Kappa gain after exchanging the two essays' ratings. On the contrary, the gradient λ_i will be incremented by a negative value $-\lambda_{i,j}$ when the another essay has a higher rating. As a result, after each iteration of MART, essays with higher rating tend to receive a higher model score while essays with lower rating tend to get a lower model score.

After the training process, the ranking model outputs an unscaled model score for each ungraded essay. To determine the final rating of each given unrated essay, we have to map this unscaled model score to the predefined scale, such as an integer from 1 to 6 in a 6 point scale. The mapping process is as follows. To begin with, the learned ranking model also computes an unscaled model score for each essay in the training set. As the model is trained by learning to rank algorithms, essays with higher model scores tend to get higher actual ratings. In other words, essays with close model scores tend to get the same rating. Therefore, we select the *k* essays whose model scores are closest to the given essay. We then remove the essays with the very highest and lowest model scores within the *k*. The final rating is the mean of the remaining $k-2$ essays' ratings. In this paper, *k* is empirically set to 5, obtained in our preliminary experiments on the ASAP validation set.

Finally, the Random Forests algorithm is used to bag K-LambdaMART learners. During the training process, both features and samples are randomly selected for each K-LambdaMART learner. In the testing phase, it outputs a score for each testing sample that is the mode of the scores output by each K-LambdaMART learner.

3.2 Pre-defined Features

We pre-define four types of features that indicate the essay quality, including lexical, syntactical, grammar and fluency, content and prompt-specific features. A brief description of these four classes of features is given below.

Lexical features: We define 4 subsets of lexical features. Each subset of features consists of one or several sub features.

– *Statistics of word length*: The number of words with length in characters larger than 4, 6, 8, 10, 12, respectively. The mean and variance of word length in characters.

– *word level*: All words in Webster dictionary [3] are divided into 8 levels according to the College Board Vocabulary Study (Breland et al., 1994). The higher level a word belongs to, the more sophisticated vocabulary usage it indicates. For example, words like thoroughfare, percolate are in level 8, while words with the same meanings, street, filter, belong to level 1. We count the number of words that belong to each level and calculate the mean word level of a given essay.

– *Unique words*: The number of unique words appeared in each essay, normalized by the essay length in words.

– *Spelling errors*: The number of spelling errors detected by the spelling check API provided by Google [4].

Syntactical features: There are 4 subsets of syntactical features.

– *Statistics of sentence length*: The number of sentences with length in words larger than 10, 18, 25, respectively. The mean and variance of sentence length in words.

– *Subclauses*: The mean number of subclauses in each sentence, normalized by sentence length in words. The mean subclause length in words. Subclauses are labeled as "SBAR" in the parser tree generated by a commonly used NLP tool, Stanford Core NLP (Klein and Manning, 2003), which is an integrated suite of natural language processing tools for English in Java[5], including part-of-speech tagging, parsing, co-reference, etc..

– *Sentence level*: The sum of the depth of all nodes in a parser tree generated by Stanford Core NLP. The height of the parser tree is also incorpo-

rated into the feature set.

– *Mode, preposition, comma*: The number of modes, prepositions and commas in each sentence respectively, normalized by sentence length in words. Part of speech (POS) is detected by Stanford Core NLP (Toutanova et al., 2003). The POS tags of modal verb and preposition are "MD" and "IN", respectively.

Grammar and fluency features: There are two subsets of grammar and fluency features.

– *Word bigram and trigram*: We evaluate the grammar and fluency of an essay by calculating mean tf/TF of word bigrams and trigrams (Briscoe et al., 2010) (tf is the term frequency in a single essay and TF is the term frequency in the whole essay collection). We assume a bigram or trigram with high tf/TF as a grammar error because high tf/TF means that this kind of bigram or trigram is not commonly used in the whole essay collection but appears in the specific essay.

– *POS bigram and trigram*: Mean tf/TF of POS bigrams and trigrams. The reason is the same with word bigrams and trigrams.

Content and prompt-specific features: We define four subsets of content and prompt-specific features.

– *Essay length*: Essay length in characters and words, respectively. The fourth root of essay length in words is proved to be highly correlated with the essay score (Shermis and Burstein, 2002).

– *Word vector similarity*: Mean cosine similarity of word vectors, in which the element is the term frequency multiplied by inverse document frequency (tf-idf) (Salton, 1971) of each word. It is calculated as the weighted mean of all cosine similarities and the weight is set as the corresponding essay score.

– *Semantic vector similarity*: Semantic vectors are generated by Latent Semantic Analysis (Dumais, 2005). The calculation of mean cosine similarity of semantic vectors is the same with word vector similarity.

– *Text coherence*: Coherence in writing means that all the ideas in a paragraph flow smoothly from one sentence to the next. We only consider nouns and pronouns in each sentence as they convey more information. The relevance degree between one sentence and its next in the same paragraph is calculated as the sum of the similarity degrees between nouns and pronouns appeared in the two sentences, normalized by the sum of the two sentences' length in words. The similarity degree between words is set to 1 if coreference exists, indicated by Stanford Core NLP (Lee et al., 2013). Otherwise, it is measured by WordNet similarity package (Pedersen et al., 2004). Finally, text coherence is computed as the average relevance degree of all pairs of neighbored sentences.

The rating model is learned off-line using a set of training essays. For a given target essay, it is the feature extraction that mainly accounts for the overhead. In our experiments, it usually costs in average no more than 10 seconds on a desktop PC with an Intel i5-2410M CPU running at 2.3GHZ to extract the pre-defined features and predict a rating for a given essay, which is affordable, compared to the cost of a human rater.

4 Experimental Setup

This section presents our the experimental design, including the test dataset used, configuration of testing algorithms, feature selection and the evaluation methodology.

4.1 Test Dataset

The dataset used in our experiments comes from the Automated Student Assessment Prize (ASAP)[1], which is sponsored by the William and Flora Hewlett Foundation. Dataset in this competition[6] consists of eight essay sets. Each essay set was generated from a single prompt. The number of essays associated with each prompt ranges from 900 to 1800 and the average length of essays in word in each essay set ranges from 150 to 650. All essays were written by students in different grades and received a resolved score, namely the actual rating, from professional human raters. Moreover, ASAP comes with a validation set that can be used for parameter training. There is no overlap between this validation set and the test set used in our evaluation.

In AES, the agreement between human-machine rater is the most important measurement of success. We use quadratic weighted Kappa to evaluate the agreement between the ratings given by the AES algorithm and the actual ratings. It is widely accepted

[3] http://www.merriam-webster.com/

[4] http://code.google.com/p/google-api-spelling-java/

[5] http://nlp.stanford.edu/software/corenlp.shtml

as a reasonable evaluation measure for AES systems (Williamson, 2009), and is also the official evaluation measure in the ASAP AES competition. It is calculated on all essay topics. If there are essays that come from n essay topics, we calculate the agreement degree on each essay topic first and then compute the overall agreement degree in the z-space. In addition, analysis of variance (ANOVA) (Scheffe, 1999) is conducted to test whether significant difference exists between the two groups of scores given by human and machine raters.

4.2 Configuration of Testing Algorithms

Random Forests bagging K-LambdaMart We denote our proposed method K-LambdaMART where K stands for the Kappa-based gradient. Our implementation of RF bagging K-LambdaMART is based on the open-source RankLib toolkit[7], a library of learning to rank algorithms, in which many popular learning to rank algorithms have been implemented, e.g. LambdaMART and RankNet (Li et al., 2007). Empirical settings of parameters obtained by preliminary experiments on the ASAP validation set are as follows. For bagging: the number of bags is set to 300, subsampling rate is 0.80 and feature sampling rate is 0.50. For LambdaMART in each bag: the number of trees is set to 1, the number of tree leaves is 100 and other parameters are set to default.

Baseline algorithms We use classical machine learning algorithms, support vector machine (SVM) for classification, regression (Vapnik et al., 1996) and preference ranking (Joachims, 2006), respectively, as the baselines. These three algorithms have been used for AES in the literature (Briscoe et al., 2010) (Yannakoudakis et al., 2011). Especially, the state-of-the-art AES approach proposed by (Yannakoudakis et al., 2011) utilizes the SVM for preference ranking, a pairwise learning to rank algorithm, for training a rating model. The linear kernel is used in the experiments. The parameter C, which controls the trade-off between empirical loss and regularizer, is set by grid search on the ASAP validation set.

The original LambdaMART is not included in the baseline algorithms as it has been shown that the performance of LambdaMART is inferior to ranking SVM on the same dataset (Chen et al., 2012).

4.3 Feature Selection

Although machine learning approaches usually use the all features available for training, we try to obtain an carefully selected feature set that can withstand the scrutiny of construct validity in assessment development (Chen and Zechner, 2011). Specific steps of feature selection conducted on individual features are as follows:

To begin with, the importance of the features is determined by computing each features Pearson correlation coefficient with human raters scores based on the training set (Chen and Zechner, 2011). Features whose absolute Pearson correlation coefficient with human scores are lower than 0.20 are removed from the feature set.

Next, we calculate the inter-correlation degrees between these features. For each pair of features whose Pearson correlation coefficient larger than 0.90, one of them should be removed. The criteria for feature removing is as follows. Firstly, at least one feature in each subset of features should be remained. Satisfying the first prerequisite condition, the removed one should be linguistically less meaningful than the remaining one.

For prompt-specific rating model, feature selection is conducted on the essays associated with the same prompt. For generic rating model, the final feature set used for training is the intersection of the 8 feature sets for prompt-specific rating model.

For space reason, here we briefly summarize the feature selection results. Among the lexical features, word length in characters larger than 8 and 10, number of words in each of the levels from 3 to 6, number of unique words, and number of spelling errors are mostly selected. As for the syntactical features, sentence length in words larger than 18 and 25, number of commas, mean clause length and the mean depth of parser tree are usually selected. Among the grammar and fluency features, mean tf/TF of word bigrams and mean tf/TF of POS trigrams are always selected. For content and prompt-specific features, essay length in words, word vector and semantic vector similarity with high rated essays, text coherence are usually selected for training a prompt-

[6] http://www.kaggle.com/c/asap-sas/data

[7] http://people.cs.umass.edu/ vdang/ranklib.html

Table 1: Cross-validation on ASAP dataset measured by quadratic weighted Kappa.

Algorithm	Prompt-specific	ANOVA		Generic	ANOVA
SVMc (baseline)	0.7302(9.75%)	Significant		0.6319(23.93%)	Significant
SVMr (baseline)	0.7861(1.95%)	Significant		0.7022(11.52%)	Significant
SVMp (baseline)	0.7876(1.75%)	Significant		0.7669(2.11%)	Not significant
RF bagging K-LambdaMART	**0.8014**	**Not significant**		**0.7831**	**Not significant**

specific rating model. When it comes to the generic rating model, the prompt-specific features like word vector similarity and semantic vector similarity, are removed.

4.4 Evaluation Methodology

We conduct three sets of experiments to evaluate the effectiveness of our listwise learning to rank approach for automated essay scoring.

The first set of experiments evaluates our proposed approach under a prompt-specific setting. We conduct 5-fold cross-validation, where the essays of each prompt are randomly partitioned into 5 subsets. In each fold, 4 subsets are used for training, and one is used for testing. To avoid bias introduced by the random partition, we repeat the 5-fold cross-validation for 5 times on 5 different random partitions. The overall quadratic weighted Kappa is averaged on all 25 test subsets.

It should be noticed that in random partition of the whole dataset, the overlap between any two partitions should be kept below $1.5*1/(\#folds)*100\%$. For example, in five-fold cross validation, the overlap should be kept below 30%. This is because: according to the Dirichlet principle (Courant, 2005), each subset in one partition overlaps more than 20% with at least one subset in another partition in five-fold cross-validation. The tolerance boundary parameter is then set to 1.5.

The objective of the second set of experiments is to test the performance of our listwise learning to rank approach for generic rating models. We also conduct 5 times 5-fold cross-validation like the first experiment. In 5-fold cross-validation, essays associated with the same prompt are randomly partitioned into 5 subsets. In this way, each fold consists of essays across all prompts. The overall performance is averaged on all 25 test subsets.

In the third set of experiments, we evaluate the quality of the features used in our rating model by feature ablation test and feature unique test. In ablation test, we evaluate our essay rating model's performance before and after the removal of a subset of features from the whole feature set. The performance difference indicates the removed features' contribution to the rating model's overall performance. In unique test, only a subset of features are used in the rating model construction and all other features are removed. The learned rating model's performance indicates to which extent the features are correlated with the actual essay ratings.

5 Experimental Results

5.1 Evaluation Results

Table 1 presents the first set of experimental results obtained on the ASAP dataset, measured by quadratic weighted Kappa. In Table 1, RF stands for random forests. SVMc, SVMr, SVMp are SVM for classification, regression and preference ranking, respectively. ANOVA stands for variance analysis, which aims to test whether significant difference exists between the scores given by human and machine raters. The improvement of our RF bagging K-LambdaMART over each baseline in percentage is also given.

For prompt-specific rating model, all of these algorithms achieve good performance comparable to human raters as literatures have revealed that the agreement between two professional human raters (measured by statistics for correlation analysis, e.g. quadratic weighted Kappa) is around 0.70 to 0.80 (Williamson, 2009) (Williamson, 2009). It is clear that our listwise learning to rank approach, Random Forests bagging K-LambdaMART, gives the best performance on the ASAP dataset. The variance analysis result on the six groups of scores (scores given by five times of five-fold cross-validation and the scores provided by human rater), no significant difference, suggests the robustness of our proposed approach. On the contrary, although pref-

erence ranking based approach, SVM for ranking, and regression based approach, SVM for regression, give very good result in human-machine agreement, their variance analysis results indicate that there exists significant difference between the scores given by human and machine raters. The result of the first set of experiments suggests the effectiveness and robustness of our listwise learning to rank approach in the building of prompt-specific rating model.

For generic rating model, one can conclude from Table 1 that RF bagging LambdaMART performs better than SVM for classification, regression and preference ranking on the ASAP dataset. The dataset used in our experiment consists of essays generated by 8 prompts and each prompt has its own features. With such a training set, both classification and regression based approaches produce not good results, as it is commonly accepted that rating model whose performance measured by inter-rater agreement lower than 0.70 is not applicable (Williamson, 2009). And the variance analysis results also reveal that there exists statistically significant difference between the scores given by human and machine raters, indicating a low robustness of these two baselines. The performance comparison of the generic rating models suggest that the rank based approaches, SVMp and RF bagging K-LambdaMART, are more effective than the classification based SVMc and the regression based SVMr, while our proposed RF bagging K-LambdaMART outperforms the state-of-the-art SVMp. Moreover, we find that there is no obvious performance difference when our proposed method is applied to prompt-specific and generic rating models. Considering the advantages generic rating models have, the result of the second set of experiments suggests the feasibility of building a rating model which is generalizable across different prompts while performs slightly inferior to the prompt-specific rating model.

5.2 Feature Analysis

Table 2 gives the results of feature ablation and unique test. In the table, "All features" stands for the use of all the features available, apart from the prompt-specific features that are not applicable to learning a generic model. In other rows, the feature subset name stands for the feature subset to be ablated in ablation test and the feature subset to be used in unique test. Note that we ablate (as in the ablation test) or use (as in the unique test) a subset of features such as the different statistics of word length as a whole since features belonging to the same subset are usually highly correlated.

Among the lexical features, the two feature subsets, word level and statistics of word length, are highly correlated with essay score in both prompt-specific and generic rating models. This observation was expected since word usage is an important notion of writing quality, regardless of essay topics.

In the syntactical features, the feature subset, sentence level, measured by the height and depth of the parser tree, correlates the most with essay score. One can infer that long sentences with nested subclauses tend to improve the final ratings.

All grammar and fluency features achieve performance around 0.60 in feature unique test for prompt-specific rating model. What is more, during feature selection, we find that the Pearson's correlation coefficient between the feature values and the final ratings in each essay prompt ranges from -0.20 to -0.60, which suggests that our method to estimate the number of grammar errors is applicable because it is widely accepted that in the evaluation of student essays, essays with more grammar errors tend to receive lower ratings.

Among the content and prompt-specific features, essay length and word vector similarity features give good results in feature unique test. The fourth root of essay length in words has been proved to be a highly correlated feature by many works on AES (Shermis and Burstein, 2002). Word vector similarity feature measures prompt-specific vocabulary usage, which is also important to essay evaluation.

In ablation test, there is no significant performance decrease no matter what feature subset is removed. It seems that each feature subset contributes little to the overall performance and therefore can be removed. However, the result of feature unique test suggests that most features used in our rating model are in fact highly correlated with the writing quality.

6 Conclusions and Future Work

We have proposed a listwise learning to rank approach to automated essay scoring (AES) by directly incorporating the human-machine agreement

Table 2: Results of feature ablation and unique test

Feature subset	Prompt-specific			Generic	
All Features	0.8014			0.7831	
	Ablation	Unique		Ablation	Unique
Lexical features					
Statistics of word length	0.7763	0.7512		0.7801	0.7350
Word level	0.7834	0.7582		0.7779	0.7306
Unique words	0.7766	0.6737		0.7692	0.6786
Spelling errors	0.7724	0.6863		0.7730	0.6742
Syntactical features					
Statistics of sentence length	0.7856	0.6410		0.7684	0.7025
Subclauses	0.7862	0.5473		0.7813	0.5050
Sentence level	0.7749	0.7046		0.7796	0.6955
Mode, preposition, comma	0.7847	0.5860		0.7807	0.5606
Grammar and fluency features					
Word bigrams and trigrams	0.7813	0.6017		0.7824	0.4395
POS bigrams and trigrams	0.7844	0.6410		0.7786	0.6022
Content and prompt-specific features					
Essay length	0.7930	0.7502		0.7736	0.7390
Word vector similarity	0.7658	0.7001		–	–
Semantic vector similarity	0.7924	0.5683		–	–
Text coherence	0.7863	0.6947		0.7798	0.6367

into the loss function. Experiments on the public English dataset ASAP show that our approach outperforms the state-of-the-art algorithms in both prompt-specific and generic rating settings. Moreover, it is widely accepted that the agreement between professional human raters ranges from 0.70 to 0.80, measured by quadratic weighted Kappa or Pearson's correlation (Powers et al., 2000) (Williamson, 2009). In the experiments, our approach achieves a quadratic weighted Kappa around 0.80 for prompt-specific rating and around 0.78 for generic rating, suggesting its potential in automated essay scoring.

Most existing research on AES focus on training a prompt-specific rating model. While such approaches have the advantage of providing a satisfactory rating accuracy for essays written for a specific topic, they also suffer from validity and feasibility problem as a significant amount of training data, namely essays with human ratings, are required for every given essay topic (Attali et al., 2010). It is therefore appealing to develop an approach that learns a generic model with acceptable rating accuracy, since it has both validity-related and logistical advantages. In our future work, we plan to continue the research on generic rating model. Because of the diversification of writing features of essays associated with different prompts, a viable approach is to explore more generic writing features that can well reflect the writing quality.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (61103131/F020511), the President Fund of UCAS (Y15101FY00/Y25102HN00), and the National Key Technology R&D Program of China (2012BAH23B03).

References

Y. Attali and J. Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Yigal Attali, Brent Bridgeman, and Catherine Trapani. 2010. Performance of a generic approach in automated essay scoring. *The Journal of Technology, Learning and Assessment*, 10(3).

L. Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

H.M. Breland, R.J. Jones, and L. Jenkins. 1994. *The college board vocabulary study*. College Entrance Examination Board.

Hermann Brenner and Ulrike Kliebsch. 1996. Dependence of weighted kappa coefficients on the number of categories. *Epidemiology*, pages 199–202.

T. Briscoe, B. Medlock, and Ø. Andersen. 2010. Automated assessment of esol free text examinations. Technical report, University of Cambridge Computer Laboratory Technical Reports, UCAM-CL-TR-790.

C. Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581.

Miao Chen and Klaus Zechner. 2011. Computing and evaluating syntactic complexity features for automated scoring of spontaneous non-native speech. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 722–731.

Hongbo Chen, Ben He, Tiejian Luo, and Baobin Li. 2012. A ranked-based learning approach to automated essay scoring. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 448–455. IEEE.

Jacob Cohen et al. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Richard Courant. 2005. *Dirichlet's principle, conformal mapping, and minimal surfaces*. Courier Dover Publications.

S. Dikli. 2006. An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment*, 5(1).

S.T. Dumais. 2005. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230.

Peter W Foltz, Darrell Laham, and Thomas K Landauer. 1999. Automated essay scoring: Applications to educational technology. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, volume 1999, pages 939–944.

Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

T. Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

L.S. Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 90–95. ACM.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

P. Li, C. Burges, and Q. Wu. 2007. Learning to rank using classification and gradient boosting. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*.

T.Y. Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pages 38–41, Boston, Massachusetts, 2-7 May. Association for Computational Linguistics, Stroudsburg, PA, USA.

Donald E Powers, Jill C Burstein, Martin Chodorow, Mary E Fowles, Karen Kukich, and Graduate Record Examinations Board. 2000. Comparing the validity of automated and human essay scoring. *RESEARCH REPORT-EDUCATIONAL TESTING SERVICE PRINCETON RR*, (10).

Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. 2008. Query-level loss functions for information retrieval. *Inf. Process. Manage.*, 44(2):838–855, mar.

G. Salton. 1971. *The SMART Retrieval System-Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Henry Scheffe. 1999. *The analysis of variance*, volume 72. Wiley. com.

M.D. Shermis and J.C. Burstein. 2002. *Automated essay scoring: A cross-disciplinary perspective*. Lawrence Erlbaum.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Vladimir Vapnik, Steven E. Golowich, and Alex Smola. 1996. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems 9*, pages 281–287. MIT Press.

D.M. Williamson. 2009. A framework for implementing automated scoring. In *Annual Meeting of the American Educational Research Association and the National Council on Measurement in Education, San Diego, CA*.

Q. Wu, C.J.C. Burges, K.M. Svore, and J. Gao. 2008. Ranking, boosting, and model adaptation. Technical report.

H. Yannakoudakis, T. Briscoe, and B. Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 180–189.

Yisong Yue and C Burges. 2007. On using simultaneous perturbation stochastic approximation for learning to rank, and the empirical optimality of lambdarank. Technical report, Technical Report MSR-TR-2007-115, Microsoft Research.

Success with Style: Using Writing Style to Predict the Success of Novels

Vikas Ganjigunte Ashok **Song Feng** **Yejin Choi**
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
vganjiguntea, songfeng, ychoi@cs.stonybrook.edu

Abstract

Predicting the success of literary works is a curious question among publishers and aspiring writers alike. We examine the quantitative connection, if any, between *writing style* and successful literature. Based on novels over several different genres, we probe the predictive power of statistical stylometry in discriminating successful literary works, and identify characteristic stylistic elements that are more prominent in successful writings. Our study reports for the first time that statistical stylometry can be surprisingly effective in discriminating highly successful literature from less successful counterpart, achieving accuracy up to 84%. Closer analyses lead to several new insights into characteristics of the writing style in successful literature, including findings that are contrary to the conventional wisdom with respect to good writing style and readability.

1 Introduction

Predicting the success of novels is a curious question among publishers, professional book reviewers, aspiring and even expert writers alike. There are potentially many influencing factors, some of which concern the intrinsic content and quality of the book, such as interestingness, novelty, style of writing, and engaging storyline, but external factors such as social context and even luck can play a role. As a result, recognizing successful literary work is a hard task even for experts working in the publication industries. Indeed, even some of the best sellers and award winners can go through several rejections before they are picked up by a publisher.[1]

Perhaps due to its obvious complexity of the problem, there has been little previous work that attempts to build statistical models that predict the success of literary works based on their intrinsic content and quality. Some previous studies do touch on the notion of stylistic aspects in successful literature, e.g., extensive studies in Literature discuss literary styles of significant authors (e.g., Ellegård (1962), McGann (1998)), while others consider content characteristics such as plots, characteristics of characters, action, emotion, genre, cast, of the best-selling novels and blockbuster movies (e.g., Harvey (1953), Hall (2012), Yun (2011)).

All these studies however, are qualitative in nature, as they rely on the knowledge and insights of human experts on literature. To our knowledge, no prior work has undertaken a systematic quantitative investigation on the overarching characterization of the writing style in successful literature. In consideration of widely different styles of authorship (e.g., Escalante et al. (2011), Peng et al. (2003), Argamon et al. (2003)), it is not even readily clear whether there might be common stylistic elements that help discriminating highly successful ones from less successful counterpart.

In this work, we present the first study that investigates this unstudied and unexpected connection between stylistic elements and the literary success. The key findings of our research reveal that there exists distinct linguistic patterns shared among suc-

[1] E.g., Paul Harding's "Tinkers" that won 2010 Pulitzer Prize for Fiction and J. K. Rowling's "Harry Potter and the Philosopher's Stone" that sold over 450 million copies.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1753–1764,
Seattle, Washington, USA, 18-21 October 2013.

cessful literature, at least within the same genre, making it possible to build a model with surprisingly high accuracy (up to 84%) in predicting the success of a novel. This result is surprising for two reasons. First, we tackle the hard task of predicting the success of novels written by *previously unseen* authors, avoiding incidental learning of authorship signature, since previous research demonstrated that one can achieve very high accuracy in authorship attribution (as high as 96% in some experimental setup) (e.g., Raghavan et al. (2010), Feng et al. (2012)). Second, we aim to discriminate highly successful novels from less successful, but nonetheless published books written by professional writers, which are undoubtedly of higher quality than average writings. It is important to note that the task we tackle here is much harder than discriminating highly successful works from those that have not even passed the scrutinizing eyes of publishers.

In order to quantify the success of literary works, and to obtain corresponding gold standard labels, one needs to first define *"success"*. For practical convenience, we largely rely on the download counts available at Project Gutenberg as a surrogate to quantify the success of novels. For a small number of novels however, we also consider award recipients (e.g., Pulitzer, Nobel), and Amazon's sales records to define a novel's success. We also extend our empirical study to movie scripts, where we quantify the success of movies based on the average review scores at `imdb.com`. We leave analysis based on other measures of literary success as future research.

In this study, we do not attempt to separate out success based on literary quality (award winners) from success based on popularity (commercial hit, often in spite of bad literary quality), mainly because it is not practically easy to determine whether the high download counts are due to only one reason or the other. We expect that in many cases, the two different aspects of success are likely to coincide, however. In the case of the corpus obtained from Project Gutenberg, where most of our experiments are conducted, we expect that the download counts are more indicative of success based on the literary quality (which then may have resulted in popularity) rather than popularity without quality.

We examine several genres in fiction and movie scripts, e.g., adventure stories, mystery, fiction, historical fiction, sci-fi, short stories, as well as poetry, and present systematic analyses based on lexical and syntactic features which have been known to be effective in a variety of NLP tasks ranging from authorship attribution (e.g., Raghavan et al. (2010)), genre detection (e.g., Rayson et al. (2001), Douglas and Broussard (2000)), gender identification (e.g., Sarawgi et al. (2011)) and native language detection (e.g., Wong and Dras (2011)).

GENRE	#BOOKS	τ^-	τ^+
Adventure	409	17	100
Detective / Mystery	374	25	90
Fiction	1148	7	125
Historical Fiction	374	25	115
Love Stories	342	16	85
Poetry	580	9	70
Science Fiction	902	30	100
Short Stories	1117	9	224

Table 1: # of books available per genre at Gutenberg with download thresholds used to define more successful ($\geq \tau^+$) and less successful ($\leq \tau^-$) classes.

Our empirical results demonstrate that (1) statistical stylometry can be surprisingly effective in discriminating successful literature, achieving accuracy up to 84%, (2) some elements of successful styles are genre-dependent while others are more universal. In addition, this research results in (3) findings that are somewhat contrary to the conventional wisdom with respect to the connection between successful writing styles and readability, (4) interesting correlations between sentiment / connotation and the literary success, and finally, (5) comparative insights between fiction and nonfiction with respect to the successful writing style.

2 Dataset Construction

For our experiments, we procure novels from project Gutenberg[2]. Project Gutenberg houses over $40,000$ books available for free download in electronic format and provides a catalog containing brief descriptions (title, author, genre, language, download count, etc.) of these books. We experiment with genres in Table 1, which have sufficient number of books allowing us to construct reasonably sized datasets.

We use the download counts in Gutenberg-catalog

[2] `http://www.gutenberg.org/`

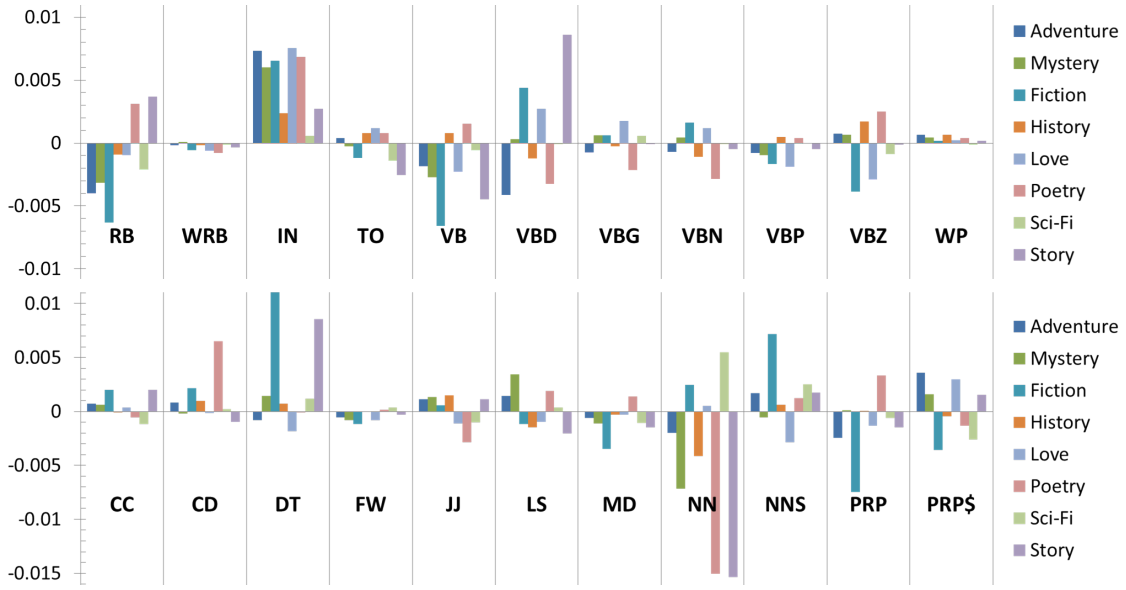


Figure 1: Differences in POS tag distribution between more successful and less successful books across different genres. Negative (positive) value indicates higher percentage in less (more) successful class.

as a surrogate to measure the degree of success of novels. For each genre, we determine a lower bound (τ_+) and an upper bound (τ_-) of download counts as shown in Table 1 to categorize the available books as more successful and less successful respectively. These thresholds are set to obtain at least 50 books for each class, and for each genre. To balance the data, for each genre, we construct a dataset of 100 novels (50 per class).

We make sure that no single author has more than 2 books in the resulting dataset, and in the majority of the cases, only one book has been taken from each author.[3] Furthermore, we make sure that the books from the same author do not show up in both training and test data. These constraints make sure that we learn general linguistic patterns of successful novels, rather than a particular writing style of a few successful authors.

3 Methodology

In what follows, we describe five different aspects of linguistic styles we measure quantitatively. The first three correspond to the features that have been frequently utilized in previous studies in related tasks, e.g., genre detection (e.g., Kessler et al. (1997)) and authorship attribution (e.g., Stamatatos (2009)), while the last two are newly explored in this work.

I. Lexical Choices: unigrams and bigrams.

II. Distribution of Word Categories: Many previous studies have shown that the distribution of part-of-speech (POS) tags alone can reveal surprising insights on genre and authorship (e.g., Koppel and Schler (2003)), hence we examine their distributions with respect to the success of literary works.

III. Distribution of Grammar Rules: Recent studies reported that features based on CFG rules are helpful in authorship attribution (e.g., Raghavan et al. (2010), Feng et al. (2012)). We experiment with four different encodings of production rules:

- Γ: lexicalized production rules (all production rules, including those with terminals)
- Γ^G: lexicalized production rules prepended with the grandparent node.
- γ: unlexicalized production rules (all production rules except those with terminals).
- γ^G: unlexicalized production rules prepended with the grandparent node.

[3]The complete list of novels used for each genre in our dataset is available at `http://www.cs.stonybrook.edu/~ychoi/successwithstyle/`

Feature	Genre								Avg	Avg w/o History
	Adven	Myster	Fiction	Histor	Love	Poetr	Sci-fi	Short		
POS	74.0	63.9	72.0	47.0	65.9	63.0	63.0	67.0	64.5	66.9
Unigram	**84.0**	73.0	**75.0**	60.0	**82.0**	71.0	61.0	57.0	70.3	71.8
Bigram	81.0	73.0	**75.0**	51.0	72.0	70.0	59.0	57.0	67.2	69.5
Γ	73.0	71.0	**75.0**	54.0	78.0	**74.0**	71.0	**77.0**	71.6	74.1
Γ^G	75.0	74.0	**75.0**	58.0	81.0	72.0	**76.0**	77.0	**73.5**	**75.7**
γ	72.0	70.0	65.0	53.0	70.0	66.0	64.0	71	66.3	68.2
γ^G	72.0	69.0	74.0	55.0	75.0	69.0	67.0	73.0	69.2	71.2
Γ+Unigram	79.0	73.0	73.0	59.0	80.0	73.0	71.0	73.0	72.6	74.5
Γ^G+Unigram	80.0	74.0	74.0	56.0	**82.0**	72.0	73.0	72.0	72.8	75.2
γ+Unigram	82.0	72.0	73.0	56.0	81.0	69.0	62.0	59.0	69.2	71.1
γ^G+Unigram	80.0	73.0	74.0	58.0	**82.0**	70.0	65.0	58.0	70	71.7
PHR	74.0	65.0	65.0	56.0	64.0	62.0	69.0	71.0	65.7	67.1
PHR+CLS	75.0	69.0	64.0	**61.0**	59.0	62.0	69.0	67.0	65.7	66.4
PHR+Unigram	80.0	74.0	71.0	56.0	79.0	73.0	67.0	66.0	70.7	72.8
PHR+CLS+Unigram	80.0	**75.0**	71.0	56.0	79.0	73.0	66.0	66.0	70.7	72.8

Table 2: Classification results in accuracy (%).

IV. Distribution of Constituents: PCFG grammar rules are overly specific to draw a big picture on the distribution of large, recursive syntactic units. We hypothesize that the distribution of constituents can serve this purpose, and that it will reveal interesting and more interpretable insights into writing styles in highly successful literature. Despite its relative simplicity, we are not aware of previous work that looks at the distribution of constituents directly. In particular, we are interested in examining the distribution of phrasal and/or clausal tags as follows: (i) Phrasal tag percent (PHR) - percentage distribution of phrasal tags.[4] (ii) Clausal tag percent (CLS) - percentage distribution of clausal tags.

V. Distribution of Sentiment and Connotation: Finally, we examine whether the distribution of sentiment and connotation words, and their polarity, has any correlation with respect to the success of literary works. We are not aware of any previous work that looks into this connection.

4 Prediction Performance

We use LibLinear SVM (Fan et al., 2008) with L2 tuned over training data, and all performance is based on 5-fold cross validation. We take 1000 sentences from the beginning of each book. POS features are encoded as unit normalized frequency and all other features are encoded as tf-idf.[5]

Prediction Results Table 2 shows the classification results. The best performance reaches as high as 84% in accuracy. In fact, in all genres except for history, the best performance is at least 74%, if not higher. Another notable observation is that even in the poetry genre, which is not prose, the accuracy gets as high as 74%. This level of performance is not entirely anticipated, given that (1) the test data consists of books written only by previously unseen authors, and (2) each author has widely different writing style, and (3) we do not have training data at scale, and (4) we aim to tackle the hard task of discriminating highly successful ones from less successful, but nonetheless successful ones, as all of them were, after all, good enough to be published.[6]

Prediction with Varying Thresholds of Download Counts Before we proceed to comprehensive analysis of writing style that are prominent in more successful literature (§5), in Table 3, we present how the prediction accuracy varies as we adjust the definition of more-successful and less-successful literature, by gradually increasing (decreasing) the threshold τ^- (τ^+). As we reduce the gap between τ^- and τ^+, the performance decreases, which shows that indeed there are notable statistical differences in linguistic patterns between novels with high and low download counts, and the stylistic difference monotonically increases (thereby higher prediction accuracy) as we increase the gap between two classes.

[4]The percentage of any phrasal tag is the count of occurrence of that tag over the sum of counts of all phrasal tags.

[5]POS tags are obtained using the Stanford POS tagger (Toutanova and Manning, 2000), and parse trees are based on the Stanford parser (Klein and Manning, 2003).

[6]In our pilot study, we also experimented with the binary classification task of discriminating highly successful ones from those that are not even published (unpublished online novels), and it was a much easier task as expected.

τ^-	τ^+	ACCURACY
17	100	84.0
25	90	78.4
35	80	77.6
45	70	76.4
55	60	73.5

Table 3: Accuracy (%) with varying thresholds of download counts for ADVENTURE with unigram features.

This is particularly interesting as the size of training data set is actually monotonically decreasing (making it harder to achieve high accuracy) while we increase the separation between τ^- and τ^+.

5 Analysis of Successful Writing Styles

5.1 Insights Based on Lexical Choices

It is apparent from Table 2 that unigram features yield curiously high performance in many genres. We therefore examine discriminative unigrams for ADVENTURE, shown in Table 4. Interestingly, less successful books rely on verbs that are explicitly descriptive of actions and emotions (e.g., "wanted", "took", "promised", "cried", "cheered", etc.), while more successful books favor verbs that describe thought-processing (e.g., "recognized", "remembered"), and verbs that serve the purpose of quotes and reports (e.g,. "say"). Also, more successful books use discourse connectives and prepositions more frequently, while less successful books rely more on topical words that could be almost cliché, e.g., "love", typical locations, and involve more extreme (e.g., "breathless") and negative words (e.g., "risk").

5.2 Distribution of Sentiment & Connotation

We also determine the distribution of sentiment and connotation words separately for each class (Table 5) to check if there exists a connection with respect to successful writing styles.[7] We first compare distribution of sentiment and connotation for the entire words. As can be seen in Table 5 – **Top**, there are not notable differences. However, when we compare distribution only with respect to discriminative unigrams only (i.e., features with non-zero weights), as shown in Table 5 – **Bottom**, we find substantial differences in all genres. In particular, discriminative unigrams that characterize less successful novels involve significantly more sentiment-laden words.

[7]We use MPQA subjectivity lexicon (Wilson et al., 2005) and connotation lexicon (Feng et al., 2013) for determining sentiment and connotation of words respectively.

Less Successful	
CATEGORY	UNIGRAMS
Negative	never, risk, worse, slaves, hard, murdered, bruised, heavy, prison,
Body Parts	face, arm, body, skins
Location	room, beach, bay, hills, avenue, boat, door
Emotional / Action Verbs	want, went, took, promise, cry, shout, jump, glare, urge
Extreme Words	never, very, breathless, sacred slightest, absolutely, perfectly
Love Related	desires, affairs
More Successful	
CATEGORY	UNIGRAMS
Negation	not
Report / Quote	said, words, says
Self Reference	I, me , my
Connectives	and, which, though, that, as, after, but, where, what, whom, since, whenever
Prepositions	up, into, out, after, in, within
Thinking Verbs	recognized, remembered

Table 4: Discriminative unigrams for ADVENTURE.

5.3 Distribution of Word Categories

Summarized analysis of POS distribution across all genres is reported in Table 6. It can be seen that prepositions, nouns, pronouns, determiners and adjectives are predictive of highly successful books whereas less successful books are characterized by higher percentage of verbs, adverbs, and foreign words. Per genre distributions of POS tags are visualized in Figure 1. Interestingly, some POS tags show almost universal patterns (e.g., prepositions (IN), NNP, WP, VB), while others are more genre-specific.

In Relation to Journalism Style The work of Douglas and Broussard (2000) reveals that informative writing (journalism) involves increased use of nouns, prepositions, determiners and coordinating conjunctions whereas imaginative writing (novels) involves more use of verbs and adverbs, as has been also confirmed by Rayson et al. (2001). Comparing their findings with Table 6, we find that highly

	Adven		Myster		Fiction		Histor		Love		Poetr		Sci-fi		Short	
	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
+ve S	4.7	4.9	4.8	4.6	5.6	4.9	5.0	5.1	5.5	5.1	6.3	5.7	4.1	3.7	4.7	4.8
-ve S	4.0	4.0	4.0	4.0	4.3	4.2	4.2	4.2	4.1	4.2	4.3	4.3	2.9	2.9	3.8	4.0
Tot S	8.7	8.9	8.9	8.7	9.9	9.0	9.2	9.3	9.6	9.3	10.6	9.9	7.0	6.7	8.5	8.9
+ve C	22.3	22.5	22.3	22.5	23.7	23.0	23.0	23.2	23.34	23.3	23.8	22.9	21.2	20.6	22.6	22.7
-ve C	19.4	19.6	19.8	19.8	20.3	19.5	19.2	19.4	20.2	20.4	17.7	17.4	16.6	16.7	18.3	18.9
Total C	41.7	42.1	42.1	42.3	44.0	42.5	42.3	42.6	43.5	43.7	41.5	40.3	37.9	37.3	41.0	41.6
+ve S	3.5	1.8	4.1	2.0	3.7	1.4	3.0	1.0	3.4	1.3	3.9	2.0	7.3	5.9	5.1	2.7
-ve S	5.5	3.4	6.3	3.6	5.5	2.9	4.7	1.9	5.1	2.6	5.8	3.3	9.0	8.0	7.3	4.8
Total S	9.1	5.2	10.4	5.6	9.2	4.3	7.7	3.0	8.5	3.9	9.7	5.2	16.3	13.9	12.4	7.5
+ve C	12.9	8.9	14.3	9.8	12.9	8.5	11.5	6.2	12.0	7.7	14.0	9.6	19.6	19.2	16.5	11.9
-ve C	14.1	9.8	15.2	10.9	13.7	9.9	12.4	7.0	12.9	8.5	14.3	10.3	20.0	19.7	17.0	13.3
Total C	27.0	18.7	29.5	20.7	26.6	18.4	23.9	13.2	24.87	16.1	28.3	19.8	39.7	38.9	33.5	25.2

Table 5: **Top:** Distribution of sentiment (connotation) among entire unigrams. **Bottom:** distribution of sentiment (connotation) among discriminative unigrams. 'S' and 'C' stand for sentiment and connotation respectively.

More Successful		
CATEGORY	SUB-CATEGORY	DIFF
Prepositions	General	0.00592
Determiners	General	0.00226
Nouns	Plural	0.00189
	Proper (Singular)	0.00016
Coord. conj.	General	0.00118
Numbers	General	0.00102
Pronouns	Posesseive	0.00081
	General WH	0.00042
	Possessive WH	5.4E-05
Adjectives	General	0.00102
	Superlative	0.00011
Less Successful		
CATEGORY	SUB-CATEGORY	DIFF
Adverbs	General	-0.00272
	General WH	-0.00028
Verbs	Base	-0.00239
	Non-3rd sing. present	-0.00084
	Past tense	-0.00041
	Past participle	-0.00039
	3rd person sing. present	-0.00036
	Modal	-0.00091
Foreign	General	-0.00067
Symbols	General	-0.00018
Interjections	General	-0.00016

Table 6: Top discriminative POS tags.

successful books tend to bear closer resemblance to informative articles.

5.4 Distribution of Constituents

It can be seen in Table 2 that deep syntactic features expressed in terms of different encodings of production rules consistently yield good performance across almost all genres. Production rules are overly specific to gain more generalized, interpretable, high-level insights however (Feng et al., 2012). Therefore, similarly as word categories (POS), we consider the categories of nonterminal nodes of the parse trees, in particular, phrasal and clausal tags, as they represent the gist of constituent structure that goes beyond shallow syntactic information represented by POS.

Table 8 shows how the distribution of phrasal and clausal tags differ for successful books when computed over all genres. Positive (negative) DIFF values indicate that the corresponding tags are favored in more successful (less successful) books when counted across all genres. We also report the number of genres (#Genres) in which the individual difference is positive / negative.

In terms of phrasal tags, we find that more successful books are composed of higher percentage of PP, NP and wh-noun phrases (WHNP), whereas less successful books are composed of higher percentage of VP, adverb phrases (ADVP), interjections (INTJ) and fragments (FRAG). Notice that this observation is inline with our earlier findings with respect to the distribution of POS.

In regard to clausal tags, more successful books involve more clausal tags that are necessary for complex sentence structure and inverted sentence structure (SBAR, SBARQ and SQ) whereas less successful books rely more on simple sentence structure (S). Figure 2 shows the visualization of the distribution of these phrasal and clausal tags.

It is also worth to mention that phrasal and clausal

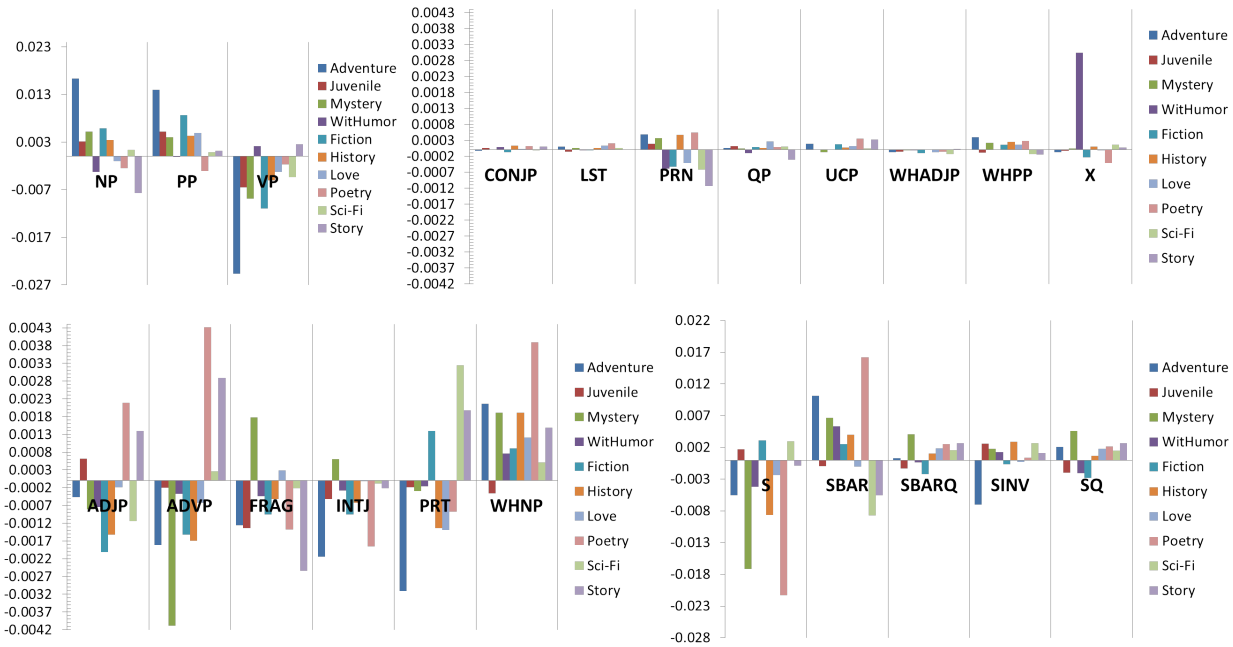


Figure 2: Difference between phrasal and clausal tag percentage distributions of more successful and less successful books across different genres. Specifically, we plot D^-–D^+, where D^+ is the phrasal tag distribution (in %) of more successful books and D^- is the phrasal tag distribution (in %) of less successful books.

READABILITY INDICES	More Succ.	Less Succ.
FOG index	9.88	**9.80**
Flesch index	87.48	**87.64**

Table 7: Readability: Lower FOG and higher Flesch indicate higher readability (numbers in Boldface).

tags alone can yield classification performance that are generally better than that of POS tags, in spite of the very small feature set (26 tags in total). In fact, constituent tags deliver the best performance in case of historical fiction genre (Table 2).

Connection to Readability Pitler and Nenkova (2008) provide comprehensive insights into assessment of readability. In their work, among the most discriminating features characterizing text with better readability is increased use of verb phrases (VP). Interestingly, contrary to the conventional wisdom – that readability is of desirable quality of good writings – our findings in Table 2 suggest that the increased use of VP correlates strongly with the writing style of the opposite spectrum of highly successful novels.

As a secondary way of probing the connection between readability and the writing style of successful literature, we also compute two different readability measures that have been used widely in prior literature (e.g., Sierra et al. (1992), Blumenstock (2008), Ali et al. (2010)): (i) Flesch reading ease score (Flesch, 1948), (ii) Gunning FOG index (Gunning, 1968). The overall weighted average readability scores are reported in Table 7. Again, we find that less successful novels have higher readability compared to more successful ones.

The work of Sawyer et al. (2008) provides yet another interesting contrasting point, where the authors found that award winning academic papers in marketing journals correlate strongly with increased readability, characterized by higher percentage of simple sentences. We conjecture that this opposite trend is likely to be due to difference between fiction and nonfiction, leaving further investigation as future research.

In sum, our analysis reveals an intriguing and unexpected observation on the connection between *readability and the literary success — that they correlate into the opposite directions*. Surely our findings only demonstrate correlation, not to be con-

Phrasal	+	−	DIFF	$\#^{+}_{Gen}/\#^{-}_{Gen}$
ADJP	0.030	0.031	-6E-4	5/3
ADJP	0.030	0.031	-6E-4	5/3
ADVP	0.052	0.054	-0.002	2/6
CONJP	3E-4	3E-4	2E-5	5/3
FRAG	0.008	0.008	-1E-4	2/6
LST	2E-4	1E-4	5E-5	6/2
NAC	9E-6	6E-6	3E-6	5/3
NP	0.459	0.453	0.005	6/2
NX	1E-4	1E-4	-4E-7	3/5
PP	0.122	0.117	0.005	7/1
PRN	0.005	0.004	2E-4	4/4
PRT	0.010	0.010	-5E-4	3/5
QP	0.001	0.001	7E-5	6/2
RRC	8E-5	8E-5	6E-6	6/2
UCP	8E-4	7E-4	1E-4	8/0
VP	0.292	0.300	-0.008	1/7
WHADJP	2E-4	2E-4	-5E-5	1/7
WHAVP	0	0	0	-
WHNP	0.013	0.012	0.001	8/0
WHPP	0.001	9E-4	1E-4	6/2
X	0.001	0.001	-4E-5	4/4
Clausal	+	−	DIFF	$\#^{+}_{Gen}/\#^{-}_{Gen}$
SBAR	0.166	0.164	0.002	4/4
SQ	0.020	0.018	0.002	7/1
SBARQ	0.014	0.013	0.001	7/1
SINV	0.018	0.018	-6E-5	5/3
S	0.781	0.785	-0.004	3/5

Table 8: Overall Phrasal / Clausal Tag Distribution and analysis. All values are rounded to [3-5] decimal places.

MORE SUCCESSFUL				
BOOK (Q)	$P_{D_{KL}}$	$UP_{D_{KL}}$	S_u	S_{Γ^*}
"Don Quixote" – Miguel De Cervantes	0.139	0.152	+	+
"Other Voices, Other Rooms" – Truman Capote	**0.014**	0.010	+	+
"The Fixer" – Bernard Malamud	0.013	0.015	+	+
"Robinson Crusoe" – Daniel Defoe	0.042	0.051	+	+
"The old man and the sea" – Ernest Hemingway	**0.065**	0.060	+	+
"A Tale of Two Cities" – Charles Dickens	0.027	0.030	+	+
"Independence Day" – Richard Ford	0.031	0.026	+	+
"Rabbit At Rest" – John Updike	0.047	0.048	+	+
"American Pastoral" – Philip Roth	0.039	0.043	+	+
"Dr Jackel and Mr. Hyde" – Robert Stevenson	0.036	0.037	+	+
LESS SUCCESSFUL				
"The lost symbol" – Dan Brown	0.046	0.042	-	-
"The magic barrel" – Bernard Malamud	0.0288	0.0284	+	-
"Two Soldiers" – William Faulkner	0.130	0.117	-	+
"My life as a man" – Philip Roth	0.046	**0.052**	-	+

Table 9: Prediction on books beyond Gutenberg. Shaded entries indicate incorrect predictions.

fused as causation, between readability and literary success. We conjecture that the conceptual complexity of highly successful literary work might require syntactic complexity that goes against readability.

6 Literature beyond Project Gutenberg

One might wonder how the prediction algorithms trained on the dataset based on Project Gutenberg might perform on books not included at Gutenberg. This section attempts to address such a question. Due to the limited availability of electronically available books that are free of charge however, we could not procure more than a handful of books.[8]

6.1 Highly Successful Books

First, we apply the classifiers trained on the Project Gutenberg dataset (all genres merged) on a few extremely successful novels (Pulitzer prize, National Award recipients, etc). Table 9 shows the results of two classification options: (1) KL-divergence based, and (2) unigram-feature based.

Although KL-divergence based prediction was not part of the classifiers that we explored in the previous sections, we include it here mainly to provide better insights as to which well-known books share closer structural similarity to either more or less successful writing style. As a probability model, we use the distributions of phrasal tags, as those can give us insights on deep syntactic structure while suppressing potential noises due to topical variances. Table 9 shows symmetrised KL-divergence between each of the previously unseen novels and the collection of books from Gutenberg corresponding to more successful (less successful) labels. For prediction, the label with smaller KL is chosen.

Based only on the distribution of 26 phrasal tags, the KL divergence classifier is able to make correct

[8]We report our prediction results on *all* books beyond Project Gutenberg of which we managed to get electronic copies, i.e., the results in Table 9 are not cherry-picked.

predictions on 7 out of 10 books, a surprisingly high performance based on mere 26 features. Of course, considering only the distribution of phrasal tags is significantly less informed than considering numerous other features that have shown substantially better performance, e.g., unigrams and CFG rewrite rules. Therefore, we also present the SVM classifier trained on unigram features. It turns out unigram features are powerful enough to make correct predictions for all ten books in Table 9.

Hemingway and Minimalism It is good to think about where and why KL-divergence-based approach fails. In fact, when we included Hemingway's *The Old Man and the Sea* into the test set, we were expecting some level of confusions when relying only on high-level syntactic structure, as Hemingway's signature style is minimalism, with 70% of his sentences corresponding to simple sentences. Not surprisingly, more adequately informed classifiers, e.g., SVM with unigram features, are still able to recognize Hemingway's writings as those of highly successful ones.

6.2 Less Successful Books

In order to obtain less successful books, we consider the *Amazon seller's rank* included in the product details of a book. The less successful books considered in Table 9 had an Amazon seller's rank beyond $200k$ (higher rank indicating less commercial success) except Dan Brown's *The lost symbol*, which we included mainly because of negative critiques it had attracted from media despite its commercial success. As shown in Table 9, all three classifiers make (arguably) correct predictions on Dan Brown's book.[9] This result also supports our earlier assumption on the nature of novels available at Project Gutenberg — that they would be more representative of literary success than general popularity (with or without literary quality).

7 Predicting Success of Movie Scripts

We have seen successful results in the novel domain, but can stylometry-based prediction work on very different domains, such as screenplays? Unlike novels, movie scripts are mostly in dialogues, which are likely to be more informal. Also, what to keep in mind is that much of the success of movies depends on factors beyond the quality of writing of the scripts, such as the quality of acting, the popularity of actors, budgets, artistic taste of directors and producers, editing and so forth.

[9]Most notable pattern based on phrasal tag analysis is a significantly increased use of fragments (FRAG), which associates strongly with less successful books in our dataset.

FEATURE	Adven	Fanta	Roman	Thril
POS	62.0	58.0	61.7	56.0
Unigram	62.0	81.3	70.0	**80.0**
Bigrams	73.3	84.7	80.8	76.0
Γ	66.0	81.3	70.0	76.0
Γ^G	62.0	69.3	**86.7**	60.0
γ	62.0	81.3	78.3	76.0
γ^G	69.3	77.3	77.5	68.0
Γ+Uni	62.0	85.3	70.0	76.0
Γ^G+Uni	54.7	81.3	70.0	76.0
γ+Uni	58.0	**89.3**	70.0	76.0
γ^G+Uni	58.0	84.7	70.0	76.0
PHR	46.0	42.7	65.8	**80.0**
PHR+CLR	**76.7**	31.3	65.8	**80.0**
PHR+Uni	62.0	81.3	70.0	**80.0**
PHR+CLR+Uni	62.0	81.3	70.0	**80.0**

Table 10: Classification results on movie dialogue data (rating ≥ 8 vs rating ≤ 5.5).

We use the Movie Script Dataset introduced in Danescu-Niculescu-Mizil and Lee (2011). It includes the dialogue scripts of 617 movies. The average rating of all movies is 6.87. We consider movies with IMDb rating ≥ 8 as "more successful", the ones with IMDb rating ≤ 5.5 as "less successful". We combine all the dialogues of each movie and filter out the movies with less than 200 sentences. There are 11 genres ("ADVENTURE", "FANTASY", "ROMANCE", "THRILLER", "ACTION", "COMEDY", "CRIME", "DRAMA", "HORROR", "MYSTERY", "SCI-FI") with 15 movies or more per class, we take 15 movies per class and perform classification tasks with the same experiment setting as Table 2.

Table 10, we show some of the example genres with relatively successful outcome, reaching as high as 89.3% accuracy in FANTASY genre. We would like to note however that in many other genres, the prediction did not work as well as it did for the novel domain. We suspect that there are at least two reasons for this: it must be partly due to very limited data size — only 15 instances per class with the rating threshold we selected for defining the success of

movies. The second reason is due to many other external factors that can also influence the success of movies, as discussed earlier.

8 Related Work

Predicting success of novels and movies: To the best of our knowledge, our work is the first that provides quantitative insights into the unstudied connection between the writing style and the success of literary works. There have been some previous work that aims to gain insights into the secret recipe of successful books, but most were qualitative, based only on a dozen of books, focusing mainly on the high-level content of the books, such as the personalities of protagonists, antagonists, the nature of plots (e.g., Harvey (1953), Yun (2011)). In contrast, our work examines a considerably larger collection of books (800 in total) over eight different sub-genres, providing insights into lexical, syntactic, and discourse patterns that characterize the writing styles commonly shared among the successful literature. Another relevant work has been on a different domain of movies (Yun, 2011), however, the prediction is based only on external, non-textual information such as the reputation of actors and directors, and the power of distribution systems etc, without analyzing the actual content of the movie scripts.

Text quality and readability: Louis (2012) explored various features that measure the quality of text, which has some high-level connections to our work. Combining the insights from Louis (2012) with our results, we find that the characteristics of text quality explored in Louis (2012), readability of text in particular, do not correspond to the prominent writing style of highly successful literature. There have been a number of other work that focused on predicting and measuring readability (e.g., Kate et al. (2010), Pitler and Nenkova (2008), Schwarm and Ostendorf (2005), Heilman and Eskenazi (2006) and Collins-Thompson et al. (2004)) employing various linguistic features.

There is an important difference however, in regard to the nature of the selected text for analysis: most studies in readability focus on differentiating good writings from noticeably bad writings, often involving machine generated text or those written by ESL students. In contrast, our work essentially deals with differentiating good writings from even better writings. After all, all the books that we analyzed are written by expert writers who passed the scrutinizing eyes of publishers, hence it is reasonable to expect that the writing quality of even less successful books is respectful.

Predicting success among academic papers: In the domain of academic papers, which belongs to the broad genre of non-fiction, the work of Sawyer et al. (2008) investigated the stylistic characteristics of award winning papers in marketing journals, and found that the readability plays an important role. Combined with our study which focuses on fiction and creative writing, it suggests that the recipe for successful publications can be very different depending on whether it belongs to fiction or nonfiction. The work of Bergsma et al. (2012) is also somewhat relevant to ours in that their work included differentiating the writing styles of workshop papers from major conference papers, where the latter would be generally considered to be more successful.

9 Conclusion

We presented the first quantitative study that learns to predict the success of literary works based on their writing styles. Our empirical results demonstrated that statistical stylometry can be surprisingly effective in discriminating successful literature, achieving accuracy up to 84% in the novel domain and 89% in the movie domain. Furthermore, our study resulted in several insights including: lexical and syntactic elements of successful styles, the connection between successful writing style and readability, the connection between sentiment / connotation and the literary success, and last but not least, comparative insights between successful writing styles of fiction and nonfiction.

Acknowledgments This research was supported in part by the Stony Brook University Office of the Vice President for Research, and in part by gift from Google. We thank anonymous reviewers, Steve Greenspan, and Mike Collins for helpful comments and suggestions, Alex Berg for the title, and Arun Nampally for helping with the preliminary work.

References

Omar Ali, Ilias N Flaounas, Tijl De Bie, Nick Mosdell, Justin Lewis, and Nello Cristianini. 2010. Automating news content analysis: An application to gender bias and readability. *Journal of Machine Learning Research-Proceedings Track*, 11:36–43.

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *TEXT-THE HAGUE THEN AMSTERDAM THEN BERLIN-*, 23(3):321–346.

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 327–337. Association for Computational Linguistics.

Joshua E Blumenstock. 2008. Automatically assessing the quality of wikipedia articles.

Kevyn Collins-Thompson, James P. Callan, and James P. Callan. 2004. A language modeling approach to predicting reading difficulty. In *HLT-NAACL*, pages 193–200.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Dan Douglas and Kathleen M Broussard. 2000. Longman grammar of spoken and written english. *TESOL Quarterly*, 34(4):787–788.

Alvar Ellegård. 1962. *A Statistical method for determining authorship: the Junius Letters, 1769-1772*, volume 13. Göteborg: Acta Universitatis Gothoburgensis.

Hugo J Escalante, Thamar Solorio, and M Montes-y Gómez. 2011. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 288–298.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Characterizing stylistic elements in syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1522–1533. Association for Computational Linguistics.

Song Feng, Jun Sak Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, Angust. Association for Computational Linguistics.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Robert Gunning. 1968. *The technique of clear writing*. McGraw-Hill New York.

James W Hall. 2012. *Hit Lit: Cracking the Code of the Twentieth Century's Biggest Bestsellers*. Random House Digital, Inc.

John Harvey. 1953. The content characteristics of best-selling novels. *Public Opinion Quarterly*, 17(1):91–114.

Michael Heilman and Maxine Eskenazi. 2006. Language learning: Challenges for intelligent tutoring systems. In *Proceedings of the workshop of intelligent tutoring systems for ill-defined tutoring systems. Eight international conference on intelligent tutoring systems*, pages 20–28.

Rohit J Kate, Xiaoqiang Luo, Siddharth Patwardhan, Martin Franz, Radu Florian, Raymond J Mooney, Salim Roukos, and Chris Welty. 2010. Learning to predict readability using diverse linguistic features. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 546–554. Association for Computational Linguistics.

Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. 1997. Automatic detection of text genre. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 32–38. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Moshe Koppel and Jonathan Schler. 2003. Exploiting stylistic idiosyncrasies for authorship attribution. In *Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, volume 69, page 72. Citeseer.

Annie Louis. 2012. Automatic metrics for genre-specific text quality. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, page 54.

Jerome McGann. 1998. *The Poetics of Sensibility: A Revolution in Literary Style*. Oxford University Press.

Fuchun Peng, Dale Schuurmans, Shaojun Wang, and Vlado Keselj. 2003. Language independent authorship attribution using character level language models. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 267–274. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: a unified framework for predicting text quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 186–195, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. 2010. Authorship attribution using probabilistic context-free grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 38–42. Association for Computational Linguistics.

Paul Rayson, Andrew Wilson, and Geoffrey Leech. 2001. Grammatical word class variation within the british national corpus sampler. *Language and Computers*, 36(1):295–306.

Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. 2011. Gender attribution: tracing stylometric evidence beyond topic and genre. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 78–86. Association for Computational Linguistics.

Alan G Sawyer, Juliano Laran, and Jun Xu. 2008. The readability of marketing journals: Are award-winning articles better written? *Journal of Marketing*, 72(1):108–117.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arlene E Sierra, Mark A Bisesi, Terry L Rosenbaum, and E James Potchen. 1992. Readability of the radiologic report. *Investigative radiology*, 27(3):236–239.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *In EMNLP/VLC 2000*, pages 63–70.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics.

Sze-Meng Jojo Wong and Mark Dras. 2011. Exploiting parse structures for native language identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1600–1610. Association for Computational Linguistics.

Chang-Joo Yun. 2011. Performance evaluation of intelligent prediction models on the popularity of motion pictures. In *Interaction Sciences (ICIS), 2011 4th International Conference on*, pages 118–123. IEEE.

A Generative Joint, Additive, Sequential Model of Topics and Speech Acts in Patient-Doctor Communication

Byron C. Wallace[†], **Thomas A. Trikalinos**[†], **M. Barton Laws**[†],
Ira B. Wilson[†] and **Eugene Charniak**[‡]
[†]Dept. of Health Services, Policy & Practice, Brown University, Providence, RI
[‡]Dept. of Computer Science, Brown University, Providence, RI
{byron_wallace, thomas_trikalinos, michael_barton_laws
ira_wilson, eugene_charniak}@brown.edu

Abstract

We develop a novel generative model of conversation that jointly captures both the topical content and the speech act type associated with each utterance. Our model expresses both token emission and state transition probabilities as log-linear functions of separate components corresponding to topics and speech acts (and their interactions). We apply this model to a dataset comprising annotated patient-physician visits and show that the proposed joint approach outperforms a baseline univariate model.

1 Introduction

Communication involves at least two aspects: the words one says and the *acts* one performs in saying them. Examples of the latter include asking questions, issuing commands, and so on. These are referred to as *speech acts* under the sociolinguistic theory of Austin (1955), which was further developed by Searle (1969; 1985). Recognizing speech acts is crucial to understanding communication because a speaker's meaning is only partially captured by the words they use; much of their intent is expressed implicitly via speech acts (Searle, 1969).

On this view, conversational utterances can be assigned both a topic and a speech act. The former describes the subject matter of what was said and the latter captures the "social act" (e.g., promising) performed by saying it. For example, the utterance "Obama won the election" is topically *political* and is an example of an *information giving* speech act. "Did Obama win the election?", meanwhile, belongs to the same topic but is a *question*. Both aspects are necessary to understand conversation.

Role	Utterance	Topic	Speech act
D	Let me just write down some of these issues here so I get them straight in my mind.	*Logistics*	*Commissive*
P	Doctor you ain't got to tell me nuttin'.	*Socializing*	*Directive*
P	I'm in very good hands when I'm around you.	*Socializing*	*Give Info.*
P	If push comes to a shove, you open the window and throw me out.	*Socializing*	*Humor/Levity*
D	I wanted to ask you, too -	*Biomedical*	*Conv. Mgmt.*
D	you know you had that colonic polyp -	*Biomedical*	*Ask Q.*
D	- is it two years from now that they're going to be doing the repeat?	*Biomedical*	*Ask Q.*
P	Yeah.	*Biomedical*	*Conv. Mgmt.*
D	We'll do the repeat coloscopy in about two years.	*Biomedical*	*Give Info.*

Table 1: An excerpt from a patient-doctor interaction, annotated with topic and speech act codes. The *D* and *P* roles denote doctor and patient, respectively. *Conv. Mgmt.* abbreviates *conversation management*; *Ask Q.* abbreviates *ask question*.

Previous computational work on speech acts – which we review in Section 6 – has modeled them in isolation (Perrault and Allen, 1980; Stolcke et al., 1998; Stolcke et al., 2000; Kim et al., 2010), i.e., independent of topical content. But a richer model would account for both speech acts *and* the contextualizing topic of each utterance. To this end, we develop a novel joint, generative model of topics and speech acts.

We focus on physician-patient communication as a motivating domain. This is of interest because

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1765–1775,
Seattle, Washington, USA, 18-21 October 2013.

it is widely appreciated that effective communication is an integral part of clinical practice (Irwin and Richardson, 2006; Makoul, 2001; Teutsch, 2003). We provide an excerpt of a conversation between a patient and their doctor annotated with topics and speech acts in Table 1. Such annotations can provide substantive insights into how doctors communicate with patients (Ong et al., 1995).

A concrete example of this is the use of topic and speech act codes to assess the efficacy of an intervention meant to influence physician-patient communication regarding adherence to antiretroviral (ARV) medication (Wilson et al., 2010). To measure the effect of the intervention, investigators performed a randomized control trial in which they quantified change in communication patterns by tallying the number of *information giving* speech acts that fell under the *ARV adherence* topic. Without assigning both topics and speech acts to utterances, this analysis would not have been possible.

In this work, we develop a novel component-based generative model for bivariate, sequentially structured problems. Our approach extends the recently proposed Sparse Additive Generative (SAGE) model (Eisenstein et al., 2011) and similar recently developed additive models (Paul and Dredze, 2012; Paul et al., 2013) to the case of supervised sequential tasks to capture the joint conditional influence of topics and speech acts, both with respect to token generation and state transitions. For brevity, we refer to this generative Joint, Additive, Sequential model as *JAS*. In contrast to previous work on speech acts, JAS provides a single, coherent generative model of conversations. And because it is component-based, this model provides a flexible framework for analyzing communication patterns. We demonstrate that JAS outperforms a generative univariate baseline in topic/speech act prediction. Further, we automatically reproduce an analysis of the aforementioned randomized control trial, and in doing so show that JAS reproduces the results more faithfully than a univariate approach.

2 The Markov-Multinomial Model

We begin by considering a baseline generative approach to modeling topics and speech acts independently. This simple approach was used by Stolcke et al. (2000) to model speech acts. It accounts for only a single output at each time point $y_t \in \mathcal{Y}$, and hence here we model topics and speech acts independently.

A straight-forward (albeit naïve) alternative would be to treat the Cartesian product of topics and speech acts as a single output space on which emissions and transitions are conditioned, but this space is too large and sparse for this approach to be practicable. We note that the *fully coupled HMM* (Brand et al., 1997) suffers from a similar exponential output state problem. The related *factorial HMM* (FHMM) (Ghahramani and Jordan, 1997; Van Gael et al., 2008), meanwhile, imposes unwarranted (in our case) independence assumptions with respect to state transitions along parallel chains, does not obviously lend itself to discrete observations (typically Gaussians are assumed), and does not scale well enough (in terms of training time) to be feasible for our application.

The Markov-Multinomial (MM) comprises two components; transitions and emissions. The former is modeled by making a first-order Markov assumption, specifically:

$$P(y_t|y_0, ..., y_{t-1}) = P(y_t|y_{t-1}) = \lambda_{y_{t-1},y_t} \quad (1)$$

Emissions can be modeled via a multinomial that captures the conditional probabilities of tokens given labels. Denoting an utterance (an utterance comprises the words corresponding to a single speech act; see Section 4) at time t by u_t and its label by y_t, and making the standard naïve assumption that words are generated independently conditioned on a label, we have:

$$P(u_t|y_t) = \prod_{w \in u_t} P(w|y_t) = \prod_{w \in u_t} \tau_{y_t,w} \quad (2)$$

Both sets of parameters (the $\boldsymbol{\lambda}$'s and the $\boldsymbol{\tau}$'s) can be estimated straight-forwardly using maximum likelihood (i.e., using observed counts). We can use Viterbi decoding (Rabiner and Juang, 1986) to make predictions for new sequences, as usual. To make both topic and speech act predictions, we simply induce models for each and make predictions independently.

3 *JAS*: A Joint, Additive, Sequential Model

An obvious shortcoming of the simple MM model outlined above is that it treats topics and speech acts

as statistically independent. They are not (as confirmed at statistical significance $p < .001$ using a χ^2 test). One would prefer a more expressive model that conditions topic and speech act transitions as well as the production of utterances jointly on both the current topic and the current speech act.

More specifically, we would like a model that reflects the assumption that some latent *intent* gives rise to both the topic and the speech act associated with an utterance. This is consistent with Searle's (1969) notion of *perlocutionary* effects; one performs speech acts with the aim of getting someone to do something. Intent gives rise to the current topic and speech act, and the current intent affects the next; this induces a correlation between adjacent topics and speech acts. This conceptual model is depicted graphically in the left-half of Figure 1.

The latent intent may be, e.g., to encourage a patient to take their medication more regularly. In our application the topical content may be *ARV adherence* and the type of speech act would be selected by the provider (presumably to maximize the likelihood of patient adherence). For example, she may opt to urge imperatively ("You really need to take your medicine") or to implore with a question ("Will you please remember to take your medicine?"). Because we have no way of explicitly modeling intent (it is never observed), we instead rely on variables for which we have annotations (i.e., the topics and speech acts; see Figure 1). We next describe the model in more detail.

We refer to the topic set by $\mathcal{Y}$, the speech act set by $\mathcal{S}$ and the vocabulary as $\mathcal{W}$. We denote the (log of the) background probability of word w by θ_w, and we will denote components corresponding to deviations from θ_w due to a specific topic (speech act) by η_w^y (η_w^s). Further, we include the component $\eta_w^{y,s}$ to capture interaction effects between topics and speech acts. We assume that the conditional probability of word w belonging to an utterance u_t with corresponding topic y_t and speech act s_t is log-linear with respect to these components, i.e.:

$$P(w|y_t, s_t) = \frac{1}{Z_w} exp\{\theta_w + \eta_w^{y_t} + \eta_w^{s_t} + \eta_w^{s_t,y_t}\} \quad (3)$$

Where Z_w is a normalizing term (implicitly conditioned on y_t and s_t) defined as:

$$Z_w = \sum_{w' \in \mathcal{W}} exp\{\theta_{w'} + \eta_{w'}^{y_t} + \eta_{w'}^{s_t} + \eta_{w'}^{s_t,y_t}\} \quad (4)$$

We make the standard naïve assumption that words are generated independently, given the topic and speech act of the utterance to which they belong:

$$P(u_t|y_t, s_t) = \prod_{w \in u_t} P(w|y_t, s_t) \quad (5)$$

The per-token emission probability just described falls under the additive generative family of models recently proposed by Eisenstein et al. (2011). However, in addition to conditional token emission probabilities, here we need also to model the *transition* probabilities such that the likelihood of transitioning to topic y_t (and to speech act s_t) reflects both the previous topic *and* the previous speech act, capturing the dependencies illustrated in Figure 1. To this end, we model topic and speech act transition probabilities as log-linear functions of the preceding topic and speech act.

We denote log of the background topic frequencies by $\boldsymbol{\pi}^{\mathcal{Y}}$, and components capturing the influence of transitioning to topic y_t due to the preceding topic and speech act by σ_{y_{t-1},y_t} and σ_{s_{t-1},y_t} respectively. We also include a component $\sigma_{(y_{t-1},s_{t-1}),y_t}$ that corresponds to the interaction effect on topic transition probability due to the preceding topic/speech act pair. We then model the topic transition probability (given the preceding states) as:

$$P(y_t|y_{t-1}, s_{t-1}) = \frac{1}{Z_y} exp\{\pi_{y_t}^{\mathcal{Y}} + \sigma_{y_{t-1},y_t} + \sigma_{s_{t-1},y_t} + \sigma_{(y_{t-1},s_{t-1}),y_t}\} \quad (6)$$

Where Z_y is a normalizing term for the topic transitions (implicitly conditioned on s_{t-1}, y_{t-1}):

$$Z_y = \sum_{y' \in \mathcal{Y}} exp\{\pi_{y'}^{\mathcal{Y}} + \sigma_{y_{t-1},y'} + \sigma_{s_{t-1},y'} + \sigma_{(y_{t-1},s_{t-1}),y'}\} \quad (7)$$

Similarly, denoting by $\boldsymbol{\pi}^{\mathcal{S}}$ log-transformed speech act background frequencies, and including analogous components as above that correspond to the influence of the preceding topic, speech act and their interaction on transitioning into speech act s_t, we

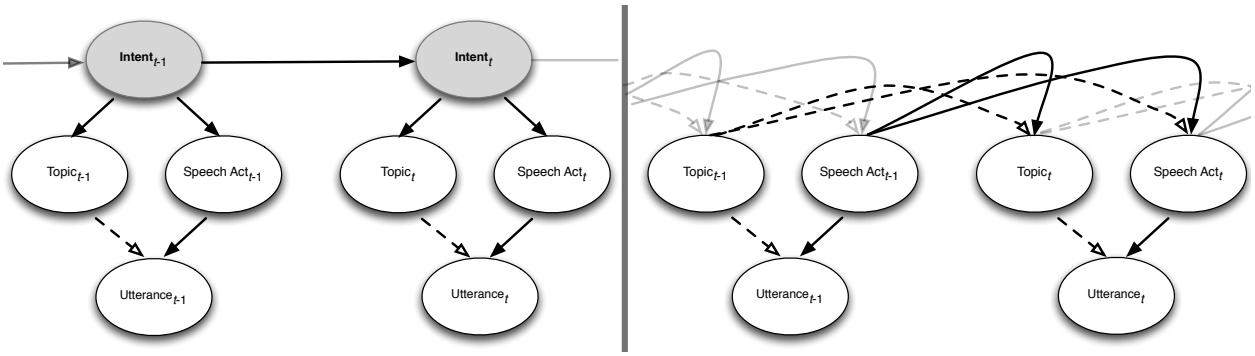


Figure 1: The generative story of utterances, depicted graphically. On the left we show our motivating conceptualization: a latent *intent* gives rise to both the topic and speech acts; these, in turn, jointly induce a distribution over words and transitions. On the right we show our operationalization of this concept. For clarity, we have denoted arrows capturing influence due to topics with dotted lines.

have:

$$P(s_t|s_{t-1}, y_{t-1}) = \frac{1}{Z_s} exp\{\pi^{\mathcal{S}}_{s_t} + \sigma_{s_{t-1},s_t} + \sigma_{y_{t-1},s_t} + \sigma_{(y_{t-1},s_{t-1}),s_t}\} \quad (8)$$

Where Z_s is a normalizing constant for speech acts analogous to Equation 7. Putting things together:

$$P(y_t, s_t|s_{t-1}, y_{t-1}, u_t) = P(u_t|y_t, s_t) \cdot P(y_t|y_{t-1}, s_{t-1}) \cdot P(s_t|s_{t-1}, y_{t-1}) \quad (9)$$

As implied by Figure 1, this model assumes that the topic and speech act at time t are conditionally independent given the preceding topic and speech act (y_{t-1} and s_{t-1}). This is intuitively agreeable because time intervenes as a blocking factor; conditioning the *current* topic on the *current* speech act (or vice versa) would contradict the fact that these occur simultaneously. Instead, the correlation is induced by the preceding topic/speech act pair. (That said, this is still a simplifying assumption, as one may instead choose to model speech act selection as conditional on topic (Traum and Larsson, 2003).)

Predictions can again be made via Viterbi decoding (Rabiner and Juang, 1986) over a matrix of *pairs* of joint topic/speech act states. The strategy of modeling (additive) components allows JAS to avoid problems due to sparsity in this large output space.

Model parameters can be estimated using standard optimization techniques. We fix the 'background' frequencies $\boldsymbol{\theta}$, $\boldsymbol{\pi}^{\mathcal{Y}}$, $\boldsymbol{\pi}^{\mathcal{S}}$ to the log of the corresponding observed proportions of words, topics and speech acts, respectively. For the remaining parameters, one can use descent-based optimization methods. The partial derivative for the topic-to-topic transition component $\sigma_{y,y'}$ with respect to the likelihood, for example, is:

$$\frac{\partial}{\partial \sigma_{y,y'}} = \sum_{s \in \mathcal{S}} C_{(y,s),y'} - P(y'|y, s) C_{(y,s),*} \quad (10)$$

Where $C_{(y,s),y'}$ denotes the observed count of transitions from topic/speech act pair (y, s) to y', and $C_{(y,s),*}$ denotes the total number of observed transitions out of this pair. The term $P(y'|y, s)$ is with respect to the current parameter estimates and is defined in Equation 6. The partial derivatives for the other component parameters (both transition and emission) are analogous. We use a Newton optimization method similar to the approach outlined by Eisenstein et al. (2011).[1] We assess convergence by calculating predictive performance on a held-out portion (5%) of the training dataset at each step, halting the descent when this declines.

4 Dataset

We use a corpus of patient-provider visits annotated with *Generalized Medical Interaction Anaylsis System* (GMIAS) codes. The GMIAS has been used to: characterize interaction processes in physician-patient communication about ARV adherence in the

[1] With the exception that we do not explicitly model the distribution over component variances.

Topic; *Speech act*	*Count (prevalence)*
ARV Adherence; Ask Q	2939 (0.013)
ARV Adherence; Commissive	245 (0.001)
ARV Adherence; Continuation	328 (0.001)
ARV Adherence; Conv. Management	4298 (0.018)
ARV Adherence; Directive	1650 (0.007)
ARV Adherence; Empathy	111 (0.000)
ARV Adherence; Give Information	12796 (0.055)
ARV Adherence; Humor/Levity	46 (0.000)
ARV Adherence; Missing/other	977 (0.004)
ARV Adherence; Social-Ritual	15 (0.000)
Biomedical; Ask Q	13753 (0.059)
Biomedical; Commissive	1049 (0.005)
Biomedical; Continuation	1005 (0.004)
Biomedical; Conv. Management	17611 (0.076)
Biomedical; Directive	4617 (0.020)
Biomedical; Empathy	423 (0.002)
Biomedical; Give Information	54231 (0.233)
Biomedical; Humor/Levity	255 (0.001)
Biomedical; Missing/other	4426 (0.019)
Biomedical; Social-Ritual	119 (0.001)
Logistics; Ask Q	5517 (0.024)
Logistics; Commissive	2308 (0.010)
Logistics; Continuation	435 (0.002)
Logistics; Conv. Management	9672 (0.042)
Logistics; Directive	5148 (0.022)
Logistics; Empathy	100 (0.000)
Logistics; Give Information	23351 (0.101)
Logistics; Humor/Levity	135 (0.001)
Logistics; Missing/other	2732 (0.012)
Logistics; Social-Ritual	285 (0.001)
Missing/other; Ask Q	820 (0.004)
Missing/other; Commissive	70 (0.000)
Missing/other; Continuation	1173 (0.005)
Missing/other; Conv. Management	1605 (0.007)
Missing/other; Directive	523 (0.002)
Missing/other; Empathy	48 (0.000)
Missing/other; Give Information	3994 (0.017)
Missing/other; Humor/Levity	27 (0.000)
Missing/other; Missing/other	12103 (0.052)
Missing/other; Social-Ritual	69 (0.000)
Psycho-Social; Ask Q	2933 (0.013)
Psycho-Social; Commissive	164 (0.001)
Psycho-Social; Continuation	208 (0.001)
Psycho-Social; Conv. Management	4433 (0.019)
Psycho-Social; Directive	787 (0.003)
Psycho-Social; Empathy	262 (0.001)
Psycho-Social; Give Information	15521 (0.067)
Psycho-Social; Humor/Levity	63 (0.000)
Psycho-Social; Missing/other	1199 (0.005)
Psycho-Social; Social-Ritual	36 (0.000)
Socializing; Ask Q	1283 (0.006)
Socializing; Commissive	79 (0.000)
Socializing; Continuation	85 (0.000)
Socializing; Conv. Management	2166 (0.009)
Socializing; Directive	222 (0.001)
Socializing; Empathy	73 (0.000)
Socializing; Give Information	8981 (0.039)
Socializing; Humor/Levity	306 (0.001)
Socializing; Missing/other	849 (0.004)
Socializing; Social-Ritual	1685 (0.007)

Table 2: Topic/speech act pairs and their counts.

context of an intervention trial (Wilson et al., 2010); analyze communication about sexual risk behavior (Laws et al., 2011a); elucidate the association of visit length with constructs of patient-centeredness (Laws et al., 2011b); and to describe provider-patient communication regarding ARV adherence compared with communication about other issues (Laws et al., 2012). GMIAS annotation is described at length elsewhere,[2] but we summarize it here for completeness.

GMIAS segments conversation into *utterances*. An utterance is here defined as a single completed speech act. Previous coding systems have simply defined an utterance as conveying a single thought (Roter and Larson, 2002) or any independent or unrestrictive dependent clause of a sentence (Ford and Ford, 1995). Stolcke et al. (2000) followed Meteer et al. (1995) in using "sentence-level units". These definitions provide helpful guidance to coders, but many speech acts are poorly formed grammatically, and cannot be described as a "clause". Further, some speech acts cannot be said to convey a "thought" (or sentence) at all, but rather are pre-syntactical (e.g., interjections and non-lexical utterances like laughter). In any case, most natural segmentations of conversations probably largely agree with intuition, and are not likely to differ substantially.

The model we develop in this work assumes that transcripts have been manually segmented. While this comes at some cost, segmenting is still much cheaper than *annotating* transcripts. Manually annotating a single visit with GMIAS codes takes 2-4 hours and must be performed by someone with substantive domain expertise. By contrast, segmenting transcripts into utterances takes at most 1/4th of the time as annotation and can be done by a less highly-skilled individual. That said, in future work we hope to explore incorporating automatic segmentation methods (Galley et al., 2003; Eisenstein and Barzilay, 2008) into our approach.

Each utterance is assigned a single topic code and a single speech act code. Inter-rater agreement has been observed to be relatively high for this task: Kappa between three trained annotators and a reference annotation ranged from 0.89 to 1.0 for topics and 0.81 to 0.95 for speech acts. We next de-

[2] https://sites.google.com/a/brown.edu/m-barton-laws/home/gmias

scribe the topics and speech acts we consider in more detail; Table 2 enumerates all pairs of these and their respective counts in the corpus. We note that GMIAS defines a hierarchy of both topic and speech act codes, but here we only attempt to capture the highest level codes in these hierarchies.

Topics comprise six major categories: *ARV adherence*, *biomedical*, *logistics*, *missing/other*, *psycho-social* and *socializing*. *Antiretroviral (ARV) adherence* applies to utterances that address ARV medication usage. *Biomedical* utterances subsume clinical observations and diagnostic conclusions. Utterances that concern the business of conducting a physical examination fall under *logistics*. The *missing/other* topic covers a few cases, including utterances that are effectively outside of the GMIAS universe and inaudible utterances; however we note that *missing/other* is a topic explicitly assigned by human annotators. The *psycho-social* topic includes such issues as substance abuse, recovery, employment and relationships. Finally, *socializing* refers to casual conversation unrelated to the business of the medical visit, and to social rituals such as greetings.

There are 10 speech acts:[3] *ask question*, *commissive*, *continuation*, *conversation management*, *directive*, *empathy*, *give information*, *humor/levity*, *missing/other*, and *social-ritual*. *Ask question* is self-explanatory. Utterances in which the speaker makes a promise or resolves to take action are *commissives*. A *continuation* refers to the completion of a previously interrupted speech act (these are rare). *Conversation management* describes utterances that facilitate turn-taking or guide discussion ('talk about talk'). *Directives* refer to statements that look to control or influence the behavior of the interlocutor. Utterances that express responses to emotions, concerns or feelings are coded under *empathy*. Communication of (purported) facts falls under *give information*. *Humor/levity* captures jokes and jovial conversation. *Missing/other* is the same as for topics. Finally, *social-ritual* utterances represent formalities (e.g., "thank you").

The corpus we use includes 360 GMIAS annotated patient-provider interactions (median length: 605 utterances). This data originated as part of a study designed to assess the role of the patient-provider relationship in explaining racial/ethnic disparities in HIV care. Study subjects were HIV care providers and their patients at four US care sites. The group responsible for the data are awaiting a decision from the institutional review board (IRB) regarding whether we can make this data publicly available in some form.

5 Experimental Results

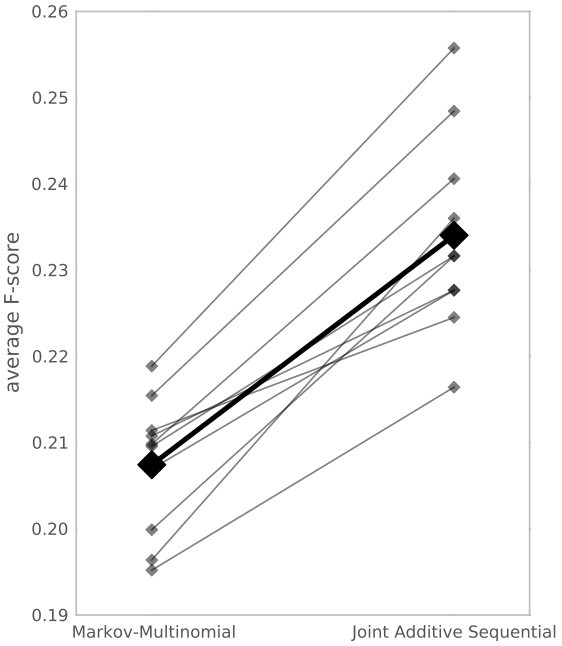


Figure 2: Mean F-scores across all topic/speech act pairs for the Markov-Multinomial (MM; left) and the proposed Joint Additive Sequential (JAS; right) models. The thick black line shows the mean difference over ten different folds; the thin grey lines describe per-fold differences. The proposed JAS model outperforms the baseline MM model for all folds

Our evaluation includes two parts.[4] First, we perform standard cross-validation over the aforementioned 360 annotated interactions, evaluating F-measure for each topic/speech act pair. Second, we look to automatically reproduce an analysis of

[3]These are high-level speech acts; technically each constitutes a *category* of speech act types.

[4]Source code at: https://github.com/bwallace/JAS; unfortunately we do not yet have permission to post the data.

a randomized control trial that assessed the efficacy of an intervention meant to alter physician-patient communication. We show that JAS outperforms the baseline approach with respect to both tasks.

We emphasize that while we are here comparing predictive performance, we are specifically interested in fully generative models of conversations due to the longer-term applications we have in mind. We would like, e.g., to use this model to assess the variation in communicative approaches across different doctors, and generative models are more naturally amenable to answering such exploratory questions. Indeed, perhaps the main strength of the additive component based sequential model we have proposed here is that it will allow us to easily incorporate physician-specific parameters that capture deviations in provider speech act and/or topic transition patterns. Further, we may soon have access to many unannotated transcripts, and we would like to learn from these; generative approaches allow straight-forward exploitation of unlabeled data. For these reasons, we did not experiment with discriminative models, e.g., Dynamic Conditional Random Fields (DCRFs) (Sutton et al., 2007) for this work.

5.1 Cross-fold Validation

Our aim is to measure model performance in terms of correctly identifying both the topic and speech act corresponding to each utterance. We quantify this via the F-score calculated for each topic/speech act pair that is observed at least once. One can see in Table 2 that many such pairs have low prevalence; this can result in undefined F-scores (e.g., when no utterances are assigned to a given pair). In this case, it is reasonable to treat these as zero values, as is commonly done (Forman and Scholz, 2010). This penalizes models when they completely fail to identify an entire class of utterances.

We first report macro-averages, that is, averages of the individual topic/speech act pair F-scores. Figure 2 displays the macro-averaged F-score for each of the 10 folds (grey lines connect folds) and the average of these (thick black line). The JAS model achieves an average macro-averaged F-score of .234 versus the .207 achieved by baseline Markov-Multinomial (MM) model; JAS outperforms MM on every fold.

For a more granular picture, Figure 3 displays average F-score differences with respect to every individual topic and speech act pair for which this difference was non-zero. This is the (signed) difference of the F-score achieved using JAS minus that achieved using the MM model; black lines thus correspond to pairs for which JAS outperformed MM, and red lines to pairs for which MM outperformed JAS. The latter achieves an improvement of $>= .05$ for 10 pairs, and results in an F-score of $> .02$ below that attained by MM only once.

The relatively low F-scores for the metrics quantifying performance with respect to the cross of topic and speech act codes belie relatively good *overall* (marginal) predictive performance. That is, we achieve much better performance with respect to metrics that measure topic and speech act predictions independently of one another. This is due to the very large output space under consideration (see Table 2). Specifically, averaged over ten runs, the MM model achieves a marginal mean topic F-score of .667 and marginal mean speech act F-score of .516. JAS begets a marginal mean topic F-score of .661 and a marginal mean speech act F-score of .544; hence the JAS model incurs an F-score loss of .006 (a 0.9% decrease) with respect to marginal topic code prediction, but improves the marginal speech act F-score by .028 (a 5.4% increase).

5.2 (Re-)Analysis of Randomized Control Trial

We also evaluated performance by tallying model predictions over 116 held-out cases collected from a randomized, cross-over study of an intervention aimed at improving physicians knowledge of patients anti-retroviral (ARV) adherence (Wilson et al., 2010). The intervention was a report given to the physician before a routine office visit that contained information regarding the patients ARV usage and their beliefs about ARV therapy. To explore the efficacy of this intervention, 58 paired (116 total) audio recorded visits were annotated with GMIAS; 58 correspond to visits before which the provider was not provided with the report (control cases), while the other 58 correspond to visits before which they were (intervention cases).

Wilson et al. (2010) demonstrated that the intervention indeed increased adherence-related dialogue, and specifically the number of *information giving* speech acts performed by the physician un-

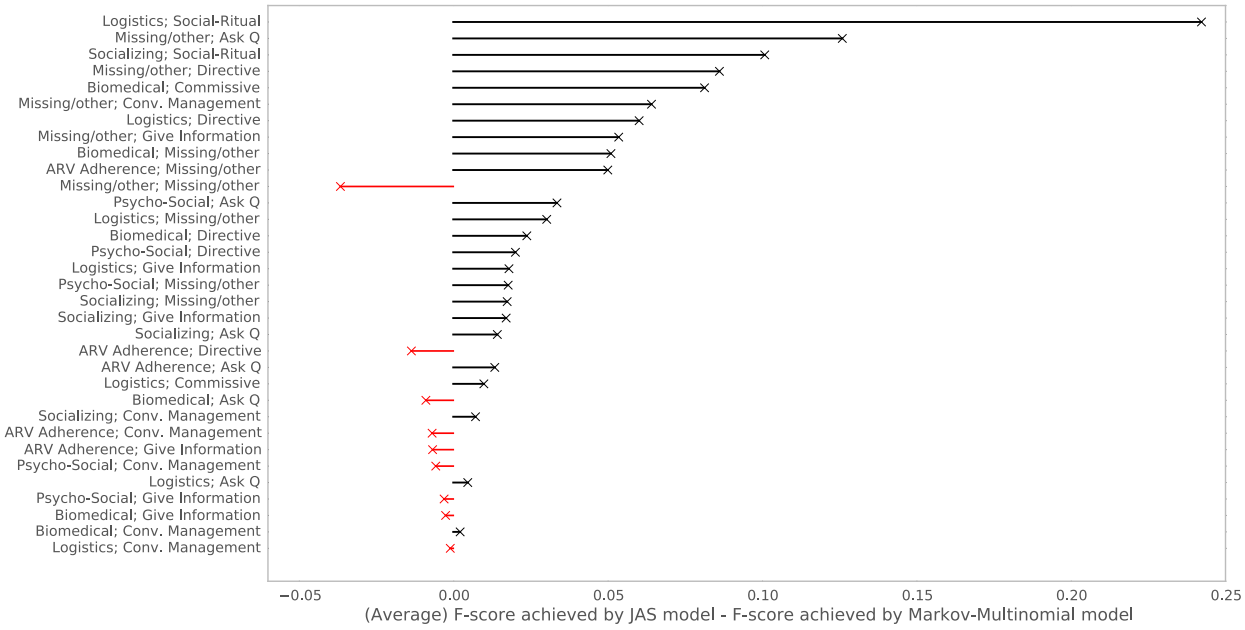


Figure 3: Average difference in F-scores corresponding to specific topic/speech act pairs, sorted by magnitude. Black lines (extending rightward) represent pairs for which JAS outperforms the baseline model; red lines (leftward) are pairs for which baseline performs better.

True		MM		JAS	
control	intervention	control	intervention	control	intervention
10 (4, 28)	23 (11, 39)	13 (5, 33)	27 (16, 44)	12 (5, 28)	23 (14, 40)

Table 3: Utterance counts {Median (25th, 75th percentile)} for the *ARV/information giving* topic and speech act pair. We show the 'gold standard' (True) tallies, which were assigned by humans, and the counts taken using the two models, MM and JAS. The JAS model predictions are closer to the true numbers.

derneath this topic. We attempted to reproduce this finding using automated rather manual annotations. To this end, we trained MM and JAS models over the aforementioned 360 annotated visits and then used this model to generate topic and speech act code predictions for the utterances comprising the 116 held-out visits used for the analysis (these were *not* part of the training set). We then assessed the direction and magnitude of the change in the number of *ARV adherence/information giving* utterances in the paired control versus intervention cases. We compared the results for this analysis calculated using the true (manually assigned) codes to the results calculated using the predicted codes.

Following the original analysis (Wilson et al., 2010), we report the median number of *ARV/information giving* utterances and corresponding 25th and 75th percentiles over the 58 control and intervention visits, as counted using the true (human) annotations and using the codes predicted by the MM and JAS models. These are reported in Table 3. The JAS model predictions better match the true labels in all except one case (the lower 25th for the controls, for which it predicts the same number as the MM model).

6 Related work

There is a relatively long history of research into modeling conversational speech acts in computational linguistics. Perrault and Allen (1980) conducted pioneering work on computationally formalizing speech acts, though their work pre-dates statistical NLP and is therefore not directly relevant to the present work.

Stolcke et al. (2000; 1998) proposed a probabilistic approach to modeling conversational speech acts based on the Hidden Markov Model (HMM) (Rabiner and Juang, 1986). They were interested in modeling an unrestricted set of conversations, and did not impose a hierarchy on the speech acts; they

therefore enumerated many more speech acts (42) than we do in the present work (recall that we use 10 'high-level' speech acts).[5] Their model has served as the baseline approach in the present work. Stolcke et al. also considered jointly performing speech *recognition* and speech act classification.

Others have investigated visual structures of patient-provider interactions to qualitatively assess communication in care. Specifically, (Cretchley et al., 2010) leveraged concept maps to explore conversations between people with schizophrenia and their carers. Briefly, this approach allowed them to (qualitatively) identify two distinct conversational strategies used by care-takers and their patients. Angus et al. (Angus et al., 2012) presented a similar approach in which they used text visualization software to explore patterns of (inferred) topics in consultations.

Another thread of research has investigated classifying speech acts in emails into one of a small set of "email speech acts", e.g., *request*, *propose*, *commit* (Cohen et al., 2004; Goldstein et al., 2006). Cohen et al. (2004) demonstrated that good performance can be achieved for this task via existing text classification technologies. Elsewhere, researchers have explored automatically inferring "speech acts" in various other online social mediums, including message board posts (Qadir and Riloff, 2011), Wikipedia talk pages (Ferschke et al., 2012) and Twitter (Zhang et al., 2012).

A separate line of inquiry concerns classifying dialogue acts in chat. Researchers have attempted dialogue act classification both for 1-on-1 (Kim et al., 2010) and multi-party (Kim et al., 2012; Clark and Popescu-Belis, 2004) online chats. Ang et al. (2005) considered the task of jointly segmenting and classifying utterances comprising multiparty meetings, while Hsueh and Moore (2006) proposed analogous methods for *topic* segmentation and labeling (other works on topic segmentation include (Galley et al., 2003) and (Eisenstein and Barzilay, 2008)). Incorporating such segmentation methods into the proposed model (rather than relying on inputs to be manually segmented beforehand) would be a natural extension of this work.

Additive component models of text have recently gained traction (Eisenstein et al., 2011; Paul, 2012; Paul and Dredze, 2012; Paul et al., 2013). To our knowledge, this is the first extension of supervised additive component models to a sequential task.[6]

7 Conclusions and Future Directions

We have proposed a novel Joint, Additive, Sequential (JAS) model of conversational topics and speech acts. In contrast to previous approaches to modeling conversational exchanges, this model factors both the current topic and the current speech act into token emission *and* state transition probabilities. We demonstrated that this model consistently outperforms a univariate generative baseline that treats speech acts and topics independently. Furthermore, we showed JAS can automatically re-produce the analysis of a randomized control trial designed to assess the efficacy of an intervention to alter physician communication habits with high-fidelity.

The generative component-based framework we have introduced in this work provides a means of exploring factors in patient-physician communication. One limitation of the model we have presented is that it makes several simplifying assumptions around dialogue. For example, we have ignored non-linearities and 'back-channels' in conversation, and we have ignored differences across physicians with respect to communication styles.

Going forward, we hope to address these limitations. We also plan on extending this model to investigate qualitative questions surrounding patient-physician communication quantitatively. For example, we are interested in investigating how communication varies across hospitals and physicians. To explore this, we can add additional components to the transition probability terms corresponding to different hospitals and doctors. Ultimately, we would like to correlate patterns in physician communication (as gleaned from the model) with objective, measured health outcomes (e.g., patient satisfaction and adherence to ARVs).

[5] We note that only 8 of the 42 speech acts appeared with greater than 1% frequency in Stolcke et al.'s corpus.

[6] Though Paul (2012) recently proposed 'mixed-membership' Markov models for *unsupervised* conversation modeling.

8 Acknowledgements

The authors thank members of the Brown Laboratory for Linguistic Information Processing (BLLIP) and Kevin Small for providing helpful feedback on earlier versions of this work. We also thank the three anonymous EMNLP reviewers for insightful comments. This work was partially supported by the National Institute of Mental Health (2 K24MH092242, R34MH089279 and R01MH083595) and by NIDA (R01DA015679).

References

Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *Proc. ICASSP*, volume 1, pages 1061–1064.

Daniel Angus, Bernadette Watson, Andrew Smith, Cindy Gallois, and Janet Wiles. 2012. Visualising conversation structure across time: Insights into effective doctor-patient consultations. *PloS one*, 7(6).

John Langshaw Austin. 1955. *How to do things with words*, volume 88. Harvard University Press.

Matthew Brand, Nuria Oliver, and Alex Pentland. 1997. Coupled hidden Markov models for complex action recognition. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 994–999. IEEE.

Alexander Clark and Andrei Popescu-Belis. 2004. Multi-level dialogue act tags. In *Proc. SIGdial*, pages 163–170.

William W Cohen, Vitor R Carvalho, and Tom M Mitchell. 2004. Learning to classify email into speech acts. In *Proceedings of EMNLP*, volume 4. sn.

Julia Cretchley, Cindy Gallois, Helen Chenery, and Andrew Smith. 2010. Conversations between carers and people with schizophrenia: a qualitative analysis using leximancer. *Qualitative Health Research*, 20(12):1611–1628.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 334–343. Association for Computational Linguistics.

J. Eisenstein, A. Ahmed, and E.P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of ICML*, pages 1041–1048.

Oliver Ferschke, Iryna Gurevych, and Yevgen Chebotar. 2012. Behind the article: Recognizing dialog acts in wikipedia talk pages. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 777–786. Citeseer.

Jeffrey D Ford and Laurie W Ford. 1995. The role of conversations in producing intentional change in organizations. *Academy of Management Review*, pages 541–570.

George Forman and Martin Scholz. 2010. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. volume 12, pages 49–57. ACM.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 562–569. Association for Computational Linguistics.

Zoubin Ghahramani and Michael I Jordan. 1997. Factorial hidden Markov models. *Machine learning*, 29(2-3):245–273.

Jade Goldstein, Andrew Kwasinski, Paul Kingsbury, R Sabin, and Albert McDowell. 2006. Annotating subsets of the enron email corpus. In *Proceedings of the Third Conference on Email and Anti-Spam*. Citeseer.

P-Y Hsueh and Johanna D Moore. 2006. Automatic topic segmentation and labeling in multiparty dialogue. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 98–101. IEEE.

Richard S Irwin and Naomi D Richardson. 2006. Patient-focused careusing the right tools. *CHEST Journal*, 130(1_suppl):73S–82S.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 862–871. Association for Computational Linguistics.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2012. Classifying dialogue acts in multi-party live chats.

Michael Barton Laws, Ylisabyth S Bradshaw, Steven A Safren, Mary Catherine Beach, Yoojin Lee, William Rogers, and Ira B Wilson. 2011a. Discussion of sexual risk behavior in HIV care is infrequent and appears ineffectual: a mixed methods study. *AIDS and Behavior*, 15(4):812–822.

Michael Barton Laws, Lauren Epstein, Yoojin Lee, William Rogers, Mary Catherine Beach, and Ira B Wilson. 2011b. The association of visit length and measures of patient-centered communication in HIV care: A mixed methods study. *Patient Education and Counseling*, 85(3):e183–e188.

Michael Barton Laws, Mary Catherine Beach, Yoojin Lee, William H Rogers, Somnath Saha, P Todd Korthuis, Victoria Sharp, and Ira B Wilson. 2012. Provider-patient adherence dialogue in HIV care: results of a multisite study. *AIDS and Behavior*, pages 1–12.

Gregory Makoul. 2001. Essential elements of communication in medical encounters: the kalamazoo consensus statement. *Academic Medicine*, 76(4):390–393.

Marie W Meteer, Ann A Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Lucille ML Ong, Johanna CJM De Haes, Alaysia M Hoos, and Frits B Lammes. 1995. Doctor-patient communication: a review of the literature. *Social science & medicine*, 40(7):903–918.

Michael Paul and Mark Dredze. 2012. Factorial lda: Sparse multi-dimensional text models. In *Advances in Neural Information Processing Systems 25*, pages 2591–2599.

Michael J. Paul, Byron C. Wallace, and Mark Dredze. 2013. What affects patient (dis)satisfaction? analyzing online doctor ratings with a joint topic-sentiment model. In *AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI)*.

Michael J Paul. 2012. Mixed membership Markov models for unsupervised conversation modeling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 94–104. Association for Computational Linguistics.

C Raymond Perrault and James F Allen. 1980. A plan-based analysis of indirect speech acts. *Computational Linguistics*, 6(3-4):167–182.

Ashequl Qadir and Ellen Riloff. 2011. Classifying sentences as speech acts in message board posts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 748–758. Association for Computational Linguistics.

Lawrence Rabiner and B Juang. 1986. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16.

Debra Roter and Susan Larson. 2002. The roter interaction analysis system (rias): utility and flexibility for analysis of medical interactions. *Patient education and counseling*, 46(4):243–251.

John R Searle. 1969. *Speech acts: An essay in the philosophy of language*. Cambridge university press.

John R Searle. 1985. *Expression and meaning: Studies in the theory of speech acts*. Cambridge University Press.

Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Noah Coccaro, Daniel Jurafsky, Rachel Martin, Marie Meteer, Klaus Ries, Paul Taylor, and Carol Van Ess-Dykema. 1998. Dialog act modeling for conversational speech. In *AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 98–105.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723.

Carol Teutsch. 2003. Patient-doctor communication. *The medical clinics of North America*, 87(5):1115.

David R Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353. Springer.

Jurgen Van Gael, Yee Whye Teh, and Zoubin Ghahramani. 2008. The infinite factorial hidden Markov model. In *Neural Information Processing Systems*, volume 21.

Ira B Wilson, M Barton Laws, Steven A Safren, Yoojin Lee, Minyi Lu, William Coady, Paul R Skolnik, and William H Rogers. 2010. Provider-focused intervention increases adherence-related dialogue, but does not improve antiretroviral therapy adherence in persons with HIV. *Journal of acquired immune deficiency syndromes*, 53(3):338.

Renxian Zhang, Dehong Gao, and Wenjie Li. 2012. Towards scalable speech act recognition in twitter: Tackling insufficient training data. *EACL 2012*, page 18.

Harvesting Parallel News Streams to Generate Paraphrases of Event Relations

Congle Zhang, Daniel S. Weld
Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{clzhang,weld}@cs.washington.edu

Abstract

The distributional hypothesis, which states that words that occur in similar contexts tend to have similar meanings, has inspired several Web mining algorithms for paraphrasing semantically equivalent phrases. Unfortunately, these methods have several drawbacks, such as confusing synonyms with antonyms and causes with effects. This paper introduces three Temporal Correspondence Heuristics, that characterize regularities in parallel news streams, and shows how they may be used to generate high precision paraphrases for event relations. We encode the heuristics in a probabilistic graphical model to create the NEWSSPIKE algorithm for mining news streams. We present experiments demonstrating that NEWSSPIKE significantly outperforms several competitive baselines. In order to spur further research, we provide a large annotated corpus of timestamped news articles as well as the paraphrases produced by NEWSSPIKE.

1 Introduction

Paraphrasing, the task of finding sets of semantically equivalent surface forms, is crucial to many natural language processing applications, including relation extraction (Bhagat and Ravichandran, 2008), question answering (Fader et al., 2013), summarization (Barzilay et al., 1999) and machine translation (Callison-Burch et al., 2006). While the benefits of paraphrasing have been demonstrated, creating a large-scale corpus of high precision paraphrases remains a challenge — especially for event relations.

Many researchers have considered generating paraphrases by mining the Web guided by the *distributional hypothesis*, which states that words occurring in similar contexts tend to have similar meanings (Harris, 1954). For example, DIRT (Lin and Pantel, 2001) and Resolver (Yates and Etzioni, 2009) identify synonymous relation phrases by the distributions of their arguments. However, the distributional hypothesis has several drawbacks. First, it can confuse antonyms with synonyms because antonymous phrases appear in similar contexts as often as synonymous phrases. For the same reasons, it also often confuses causes with effects. For example, DIRT reports that the closest phrase to *fall* is *rise*, and the closest phrase to *shoot* is *kill*.[1] Second, the distributional hypothesis relies on statistics over large corpora to produce accurate similarity statistics. It remains unclear how to accurately paraphrase less frequent relations with the distributional hypothesis.

Another common approach employs the use of parallel corpora. News articles are an interesting target, because there often exist articles from different sources describing the same daily events. This peculiar property allows the use of the temporal assumption, which assumes that phrases in articles published at the same time tend to have similar meanings. For example, the approaches by Dolan *et al.* (2004) and Barzilay *et al.* (2003) identify pairs of sentential paraphrases in similar articles that have appeared in the same period of time. While these approaches use temporal information as a coarse filter in the data generation stage, they still largely rely on text metrics in the prediction stage. This not only reduces precision, but also limits the discovery of paraphrases with dissimilar sur-

[1] http://demo.patrickpantel.com/demos/lexsem/paraphrase.htm

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1776–1786,
Seattle, Washington, USA, 18-21 October 2013.

face strings.

The goal of our research is to develop a technique to generate paraphrases for large numbers of event relation with high precision, using only minimal human effort. The key to our approach is a joint cluster model using the temporal attributes of news streams, which allows us to identify semantic equivalence of event relation phrases with greater precision. In summary, this paper makes the following contributions:

- We formulate a set of three *temporal correspondence heuristics* that characterize regularities over parallel news streams.
- We develop a novel program, NEWSSPIKE, based on a probabilistic graphical model that jointly encodes these heuristics. We present inference and learning algorithms for our model.
- We present a series of detailed experiments demonstrating that NEWSSPIKE outperforms several competitive baselines, and show through ablation tests how each of the temporal heuristics affects performance.
- To spur further research on this topic, we provide both our generated paraphrase clusters and a corpus of 0.5M time-stamped news articles[2], collected over a period of about 50 days from hundreds of news sources.

2 System Overview

The main goal of this work is to generate high precision paraphrases for relation phrases. News streams are a promising resource, since articles from different sources tend to use semantically equivalent phrases to describe the same daily events. For example, when a recent scandal hit, headlines read: *"Armstrong steps down from Livestrong"*; *"Armstrong resigns from Livestrong"* and *"Armstrong cuts ties with Livestrong"*. From these we can conclude that the following relation phrases are semantically similar: {*step down from, resign from, cut ties with*}.

To realize this intuition, our first challenge is to represent an event. In practice, a question like *"What happened to Armstrong and Livestrong on Oct 17?"* could often lead to a unique answer. It implies that using an argument pair and a time-stamp could be an effective way to identify an event (*e.g. (Armstrong, Livestrong, Oct 17)* for the previous question). Based on this observation, this paper introduces a novel mechanism to paraphrase relations as summarized in Figure 1.

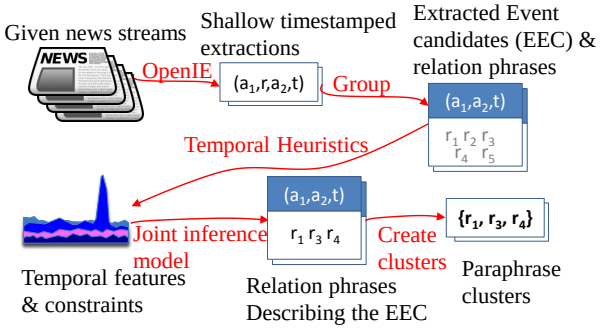


Figure 1: NEWSSPIKE first applies open information extraction to articles in the news streams, obtaining shallow extractions with time-stamps. Next, an *extracted event candidate* (EEC) is obtained after grouping daily extractions by argument pairs. Temporal features and constraints are developed based on our temporal correspondence heuristics and encoded into a joint inference model. The model finally creates the paraphrase clusters by predicting the relation phrases that describe the EEC.

NEWSSPIKE first applies the ReVerb open information extraction (IE) system (Fader et al., 2011) on the news streams to obtain a set of (a_1, r, a_2, t) tuples, where the a_i are the arguments, r is a relation phrase, and t is the time-stamp of the corresponding news article. When (a_1, a_2, t) suggests a real word event, the relation r of (a_1, r, a_2, t) is likely to describe that event (*e.g. (Armstrong, resign from, Livestrong, Oct 17*). We call every (a_1, a_2, t) an extracted event candidate (EEC), and every relation describing the event an *event-mention*.

For each EEC (a_1, a_2, t), suppose there are m extraction tuples $(a_1, r_1, a_2, t) \ldots (a_1, r_m, a_2, t)$ sharing the values of a_1, a_2, and t. We refer to this set of extraction tuples as the *EEC-set*, and denote it $(a_1, a_2, t, \{r_1 \ldots r_m\})$. All the event-mentions in the EEC-set may be semantically equivalent and are hence candidates for a good paraphrase cluster.

Thus, the paraphrasing problem becomes a prediction problem: for each relation r_i in the EEC-set, does it or does it not describe the hypothesized event? We solve this problem in two steps. The

[2] https://www.cs.washington.edu/node/9473/

next section proposes a set of temporal correspondence heuristics that partially characterize semantically equivalent EEC-sets. Then, in Section 4, we present a joint inference model designed to use these heuristics to solve the prediction problem and to generate paraphrase clusters.

3 Temporal Correspondence Heuristics

In this section, we propose a set of temporal heuristics that are useful to generate paraphrases at high precision. Our heuristics start from the basic observation mentioned previously — events can often be uniquely determined by their arguments and time. Additionally, we find that it is not just the *publication time* of the news story that matters, the *verb tenses* of the sentences are also important. For example, the two sentences *"Armstrong was the chairman of Livestrong"* and *"Armstrong steps down from Livestrong"* have past and present tense respectively, which suggests that the relation phrases are less likely to describe the same event and are thus not semantically equivalent. To capture these intuitions, we propose the *Temporal Functionality Heuristic*:

Temporal Functionality Heuristic. *News articles published at the same time that mention the same entities and use the same tense tend to describe the same events.*

Unfortunately, we find that not all the event candidates, (a_1, a_2, t), are equally good for paraphrasing. For example, today's news might include both *"Barack Obama heads to the White House"* and *"Barack Obama greets reporters at the White House"*. Although the two sentences are highly similar, sharing a_1 = "Barack Obama" and a_2 = "White House," and were published at the same time, they describe different events.

From a probabilistic point of view, we can treat each sentence as being generated by a particular hidden event which involves several actors. Clearly, some of these actors, like Obama, participate in many more events than others, and in such cases we observe sentences generated from a *mixture* of events. Since two event mentions from such a mixture are much less likely to denote the same event or relation, we wish to distinguish them from the better (semantically homogeneous) EECs like the *(Armstrong, Livestrong)* example. The question becomes "How one can distinguish good entity pairs from bad?"

Our method rests on the simple observation that an entity which participates in many different events on one day is likely to have participated in events in recent days. Therefore we can judge whether an entity pair is good for paraphrasing by looking at the *history of the frequencies* that the entity pair is mentioned in the news streams, which is the *time series* of that entity pair. The time series of the entity pair *(Barack Obama, the White House)* tends to be high over time, while the time series of the entity pair *(Armstrong, Livestrong)* is flat for a long time and suddenly spikes upwards on a single day. This observation leads to:

Temporal Burstiness Heuristic. *If an entity or an entity pair appears significantly more frequently in one day's news than in recent history, the corresponding event candidates are likely to be good to generate paraphrase.*

The temporal burstiness heuristic implies that a good EEC (a_1, a_2, t) tends to have a *spike* in the time series of its entities a_i, or argument pair (a_1, a_2), on day t.

However, even if we have selected a good EEC for paraphrasing, it is likely that it contains a few relation phrases that are related to (but not synonymous with) the other relations included in the EEC. For example, it's likely that the news story reporting *"Armstrong steps down from Livestrong."* might also mention *"Armstrong is the founder of Livestrong."* and so both "steps down from" and "is the founder of" relation phrases would be part of the same EEC-set. Inspired by the idea of one sense per discourse from (Gale et al., 1992), we propose:

One Event-Mention Per Discourse Heuristic. *A news article tends not to state the same fact more than once.*

The one event-mention per discourse heuristic is proposed in order to gain precision at the expense of recall — the heuristic directs an algorithm to choose, from a news story, the single "best" relation phrase connecting a pair of two entities. Of course, this doesn't answer the question of deciding which phrase is "best." In Section 4.3, we describe how to learn a probabilistic graphical model which does exactly this.

4 Exploiting the Temporal Heuristics

In this section we propose several models to capture the temporal correspondence heuristics, and discuss their pros and cons.

4.1 Baseline Model

An easy way to use an EEC-set is to simply predict that all r_i in the EEC-set are event-mentions, and hence are semantically equivalent. That is, given EEC-set $(a_1, a_2, t, \{r_1 \dots r_m\})$, the output cluster is $\{r_1 \dots r_m\}$.

This baseline model captures the most of the temporal functionality heuristic, except for the tense requirement. Our empirical study shows that it performs surprisingly well. This demonstrates that the quality of our input for the learning model is good: the EEC-sets are promising resources for paraphrasing.

Unfortunately, the baseline model cannot deal with the other heuristics, a problem we will remedy in the following sections.

4.2 Pairwise Model

The temporal functionality heuristic suggests we exploit the tenses of the relations in an EEC-set; while the temporal burstiness heuristic suggests we exploit the time series of its arguments. A pairwise model can be designed to capture them: we compare pairs of relations in the EEC-set, and predict whether each pair is synonymous or non-synonymous. Paraphrase clusters are then generated according to some heuristic rules (*e.g.* assuming transitivity among synonyms). The tenses of the relations and time series of the arguments are encoded as features, which we call *tense features* and *spike features* respectively. An example tense feature is whether one relation is past tense while the other relation is present tense; an example spike feature is the covariance of the time series.

The pairwise model can be considered similar to paraphrasing techniques which examine two sentences and determine whether they are semantically equivalent (Dolan and Brockett, 2005; Socher et al., 2011). Unfortunately, these techniques often based purely on text metrics and does not consider any temporal attributes. In section 5, we evaluate the effect of applying these techniques.

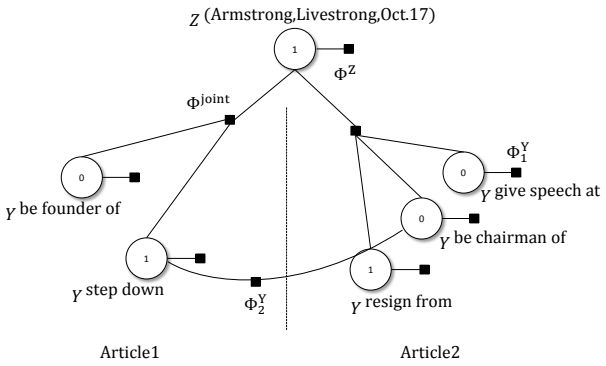


Figure 2: an example model for EEC (Armstrong, Livestrong, Oct 17). Y and Z are binary random variables. Φ^Y, Φ^Z and Φ^{joint} are factors. *be founder of* and *step down* come from article 1 while *give speech at*, *be chairman of* and *resign from* come from article 2.

4.3 Joint Cluster Model

The pairwise model has several drawbacks: *1)* it lacks the ability to handle constraints, such as the mutual exclusion constraint implied by the one-mention per discourse heuristic; *2)* ad-hoc rules, rather than formal optimizations, are required to generate clusters containing more than two relations.

A common approach to overcome the drawbacks of the pairwise model and to combine heuristics together is to introduce a joint cluster model, in which heuristics are encoded as features and constraints. Data, instead of ad-hoc rules, determines the relevance of different insights, which can be learned as parameters. The advantage of the joint model is analogous to that of cluster-based approaches for coreference resolution (CR). In particular, a joint model can better capture constraints on multiple variables and can yield higher quality results than pairwise CR models (Rahman and Ng, 2009).

We propose an undirected graphical model, NEWSSPIKE, which jointly clusters relations. Constraints are captured by factors connecting multiple random variables. We introduce random variables, the factors, the objective function, the inference algorithm, and the learning algorithm in the following sections. Figure 2 shows an example model for EEC *(Armstrong, Livestrong, Oct 17)*.

4.3.1 Random Variables

For the EEC-set $(a_1, a_2, t, \{r_1, \dots r_m\})$, we introduce one event variable and m relation variables, all boolean valued. The event variable $Z^{(a_1,a_2,t)}$ indi-

cates whether (a_1, a_2, t) is a good event for paraphrasing. It is designed in accordance with the temporal burstiness heuristic: for the EEC *(Barack Obama, the White House, Oct 17)*, Z should be assigned the value 0.

The relation variable Y^r indicates whether relation r describes the EEC (a_1, a_2, t) or not (*i.e.* r is an event-mention or not). The set of all event-mentions with $Y^r = 1$ define a paraphrase cluster, containing relation phrases. For example, the assignments $Y^{\textit{step down}} = Y^{\textit{resign from}} = 1$ produce a paraphrase cluster {*step down, resign from*}.

4.3.2 Factors and the Joint Distribution

In this section, we introduce a conditional probability model defining a joint distribution over all of the event and relation variables. The joint distribution is a function over *factors*. Our model contains *event factors*, *relation factors* and *joint factors*.

The event factor Φ^Z is a log-linear function with spike features, used to distinguish good events. A relation factor Φ^Y is also a log-linear function. It can be defined for individual relation variables (*e.g.* Φ_1^Y in Figure 2) with features such as whether a relation phrase comes from a clausal complement[3]. A relation factor can also be defined for a pair of relation variables (*e.g.* Φ_2^Y in Figure 2) with features capturing the pairwise evidence for paraphrasing, such as if two relation phrases have the same tense.

The joint factors Φ^{joint} are defined to apply constraints implied by the temporal heuristics. They play two roles in our model: *1)* to satisfy the temporal burstiness heuristic, when the value of the event variable is false, the EEC is not appropriate for paraphrasing, and so all relation variables should also be false; and *2)* to satisfy the one-mention per discourse heuristic, at most one relation variable from a single article could be true.

We define the joint distribution over these variables and factors as follows. Let $\mathbf{Y} = (Y^{r_1} \ldots Y^{r_m})$ be the vector of relation variables; let $\mathbf{x}$ be the features. The joint distribution is:

$$p(Z = z, \mathbf{Y} = \mathbf{y}|\mathbf{x}; \Theta) \stackrel{\text{def}}{=} \frac{1}{Z_x} \Phi^Z(z, \mathbf{x}) \times \prod_d \Phi^{\text{joint}}(z, \mathbf{y}_d, \mathbf{x}) \prod_{i,j} \Phi^Y(y_i, y_j, \mathbf{x})$$

where $\mathbf{y}_d$ indicates the subset of relation variables from a particular article d, and the parameter vector Θ is the weight vector of the features in Φ^Z and Φ^Y, which are log-linear functions; *i.e.*,

$$\Phi^Y(y_i, y_j, \mathbf{x}) \stackrel{\text{def}}{=} \exp\left(\sum_j \theta_j \phi_j(y_i, y_j, \mathbf{x})\right)$$

where ϕ_j is the jth feature function.

The joint factors Φ^{joint} are used to apply the temporal burstiness heuristic and the one event-mention per discourse heuristic. Φ^{joint} is zero when the EEC is not good for paraphrasing, but some $y^r = 1$; or when there is more than one r in a single article such that $y^r = 1$. Formally, it is calculated as:

$$\Phi^{\text{joint}}(z, \mathbf{y}_d, \mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } z = 0 \wedge \exists y^r = 1 \\ 0 & \text{if } \sum_{y^r \in \mathbf{y}_d} y^r > 1 \\ 1 & \text{otherwise} \end{cases}$$

4.3.3 Maximum a Posteriori Inference

The goal of inference is to find the predictions $z, \mathbf{y}$ which yield the greatest probability, *i.e.*,

$$z^*, \mathbf{y}^* = \arg\max_{z,\mathbf{y}} p(Z = z, \mathbf{Y} = \mathbf{y}|\mathbf{x}; \Theta)$$

This can be viewed as a MAP inference problem. In general, inference in a graphical model is challenging. Fortunately, the joint factors in our model are linear, and the event and relation factors are log-linear; we can cast MAP inference as an integer linear programming (ILP) problem, and then compute an approximation in polynomial time by means of linear programming using randomized rounding, as proposed in (Yannakakis, 1992).

We build one ILP problem for every EEC. The variables of the ILP are Z and $\mathbf{Y}$, which only take values of 0 or 1. The objective function is the sum of logs of the event and relation factors Φ^Z and Φ^Y. The temporal burstiness heuristic of Φ^{joint} is encoded as a linear inequality constraint $z \geq y_i$; the one-mention per discourse heuristic of Φ^{joint} is encoded as the constraint $\sum_{y_i \in \mathbf{y}_d} y_i \leq 1$.

[3]Relation phrases in clausal complement are less useful for paraphrasing because they often do not describe a fact. For example, in the sentence *He heard Romney had won the election*, the extraction (Romney, had won, the election) is not a fact at all.

4.3.4 Learning

Our training data consists a set of $N = 500$ labeled EEC-sets each in the form of $\{(R_i, R_i^{\text{gold}}) \mid_{i=1}^{N}\}$. Each R is the set of all relations in the EEC-set while R^{gold} is a manually selected subset of R containing relations describing the EEC. R^{gold} could be empty if the EEC was deemed poor for paraphrasing. For our model, the gold assignment $y^{r\,\text{gold}} = 1$ if $r \in R^{\text{gold}}$; the gold assignment $z^{\text{gold}} = 1$ if R^{gold} is not empty.

Given $\{(R_i, R_i^{\text{gold}}) \mid_{i=1}^{N}\}$, learning over similar models is commonly done via maximum likelihood estimation as follows:

$$L(\Theta) = \log \prod_i p(Z_i = z_i^{\text{gold}}, \mathbf{Y}_i = \mathbf{y}_i^{\text{gold}} \mid \mathbf{x}_i, \Theta)$$

For features in relation factors, the partial derivative for the ith model is:

$$\Phi_j(\mathbf{y}_i^{\text{gold}}, \mathbf{x}_i) - E_{p(z_i, \mathbf{y}_i|, \mathbf{x}_i, \Theta)} \Phi_j(\mathbf{y}_i, \mathbf{x}_i)$$

where $\Phi_j(\mathbf{y}_i, \mathbf{x}_i) = \sum \phi_j(X, Y, \mathbf{x})$, the sum of values for the jth feature in the ith model; and values of X, Y come from the assignment $\mathbf{y}_i$. For features in event factors, the partial derivative is derived similarly as

$$\phi_j(z_i^{\text{gold}}, \mathbf{x}_i) - E_{p(z_i, \mathbf{y}_i|, \mathbf{x}_i, \Theta)} \phi_j(z_i, \mathbf{x}_i)$$

It is unclear how to efficiently compute the expectations in the above formula, a brute force approach requires enumerating all assignments of $\mathbf{y}_i$, which is exponentially large with the number of relations. Instead, we opt to use a more tractable perceptron learning approach (Collins, 2002; Hoffmann et al., 2011). Instead of computing the expectations, we simply compute $\phi_j(z_i^*, \mathbf{x}_i)$ and $\Phi_j(\mathbf{y}_i^*, \mathbf{x}_i)$, where $z_i^*, \mathbf{y}_i^*$ is the assignment with the highest probability, generated by the MAP inference algorithm using the current weight vector. The weight updates are the following:

$$\Phi_j(\mathbf{y}_i^{\text{gold}}, \mathbf{x}_i) - \Phi_j(\mathbf{y}_i^*, \mathbf{x}_i) \quad (1)$$

$$\phi_j(z_i^{\text{gold}}, \mathbf{x}_i) - \phi_j(z_i^*, \mathbf{x}_i) \quad (2)$$

The updates can be intuitively explained as penalties on errors. In sum, our learning algorithm consists of iterating the following two steps: (1) infer the most probable assignment given the current weights; (2) update the weights by comparing inferred assignments and the truth assignment.

5 Empirical Study

We first introduce the experimental setup for our empirical study, and then we attempt to answer two questions in sections 5.2 and 5.3 respectively: First, does the NEWSSPIKE algorithm effectively exploit the proposed heuristics and outperform other approaches which also use news streams? Secondly, do the proposed temporal heuristics paraphrase relations with greater precision than the distributional hypothesis?

5.1 Experimental Setup

Since we were unable to find any elaborate time-stamped, parallel, news corpus, we collected data using the following procedure:

- Collect RSS news seeds, which contain the title, time-stamp, and abstract of the news items.
- Use these titles to query the Bing news search engine API and collect additional time-stamped news articles.
- Strip HTML tags from the news articles using Boilerpipe (Kohlschütter et al., 2010); keep only the title and first paragraph of each article.
- Extract shallow relation tuples using the OpenIE system (Fader et al., 2011).

We performed these steps every day from January 1 to February 22, 2013. In total, we collected 546,713 news articles, for which 2.6 million extractions had 529 thousand unique relations.

We used several types of features for paraphrasing: *1)* spike features obtained from time series; *2)* tense features, such as whether two relation phrases are both in the present tense; *3)* cause-effect features, such as whether two relation phrases often appear successively in the news articles; *4)* text features, such as whether sentences are similar; *5)* syntactic features, such as whether a relation phrase appears in a clausal complement; and *6)* semantic features, such as whether a relation phrase contains negative words.

Text and semantic features are encoded using the relation factors of section 4.3.2. For example, in Figure 2, the factor Φ_2^Y includes the textual similarity between the sentences containing the phrases *"step down"* and *"be chairman of"* respectively; it also includes the feature that the tense of *"step down"* (present) is different from the tense of *"be chairman*

output	{**go into**, **go to**, *speak*, *return*, **head to**}
gold	{**go into**, **go to**, *approach*, **head to**}
gold$_{\text{div}}$	{**go** $*$, *approach*, **head to**}
P/R	precision $= 3/5$ recall $= 3/4$
P/R$_{\text{div}}$	precision$_{\text{div}} = 2/4$ recall$_{\text{div}} = 2/3$

Figure 3: an example pair of the output cluster and the gold cluster, and the corresponding precision recall numbers.

of" (past).

5.2 Comparison with Methods using Parallel News Corpora

We evaluated NEWSSPIKE against other methods that also use time-stamped news. These include the models mentioned in section 3 and state-of-the-art paraphrasing techniques.

Human annotators created gold paraphrase clusters for 500 EEC-sets; note that some EEC-sets yield no gold cluster, since at least two synonymous phrases. Two annotators were shown a set of candidate relation phrases in context and asked to select a subset of these that described a shared event (if one existed). There was 98% phrase-level agreement. Precision and recall were computed by comparing an algorithm's output clusters to the gold cluster of each EEC. We consider paraphrases with minor lexical diversity, *e.g. (go to, go into)*, to be of lesser interest. Since counting these trivial paraphrases tends to exaggerate the performance of a system, we also report precision and recall on *diverse clusters i.e.*, those whose relation phrases all have different head verbs. Figure 3 illustrates these metrics with an example; note under our diverse metrics, all phrases matching *go* * count as one when computing both precision and recall. We conduct 5-fold cross validation on our labeled dataset to get precision and recall numbers when the system requires training.

We compare NEWSSPIKE with the models in Section 4, and also with the state-of-the-art paraphrase extraction method:

Baseline: the model discussed in Section 4.1. This system does not need any training, and generates outputs with perfect recall.

Pairwise: the pairwise model discussed in Section 4.2 and using the same set of features as used by NEWSSPIKE. To generate output clusters, transitivity is assumed inside the EEC-set. For example, when the pairwise model predicts that (r_1, r_2) and (r_1, r_3) are both paraphrases, the resulting cluster is $\{r_1, r_2, r_3\}$.

System	P/R		P/R diverse	
	prec	rec	prec	rec
Baseline	0.67	1.00	0.53	1.00
Pairwise	0.90	0.60	0.81	0.37
Socher	0.81	0.35	0.68	0.29
NEWSSPIKE	**0.92**	0.55	**0.87**	0.31

Table 1: Comparison with methods using parallel news corpora

Socher: Socher *et al.* (2011) achieved the best results on the Dolan *et al.* (2004) dataset, and released their code and models. We used their off-the-shelf predictor to replace the classifier in our Pairwise model. Given sentential paraphrases, aligning relation phrases is natural, because OpenIE has already identified the relation phrases.

Table 1 shows precision and recall numbers. It is interesting that the basic model already obtains 0.67 precision overall and 0.53 in the diverse condition. This demonstrates that the EEC-sets generated from the news streams are a promising resource for paraphrasing. Socher's method performs better, but not as well as Pairwise or NEWSSPIKE, especially in the diverse cases. This is probably due to the fact that Socher's method is based purely on text metrics and does not consider any temporal attributes. Taking into account the features used by NEWSSPIKE, Pairwise significantly improves the precision, which demonstrates the power of our temporal correspondence heuristics. Our joint cluster model, NEWSSPIKE, which considers both temporal features and constraints, gets the best performance in both conditions.

We conducted ablation testing to evaluate how spike features and tense features, which are particularly relevant to the temporal aspects of news streams, can improve performance. Figure 4 compares the precision/recall curves for three systems in the diverse condition: (1) NEWSSPIKE; (2) w/oSpike: turning off all spike features; and (3) w/oTense: turning off all features about tense. (4) w/oDiscourse: turning off one event-mention per discourse heuristic. There are some dips in

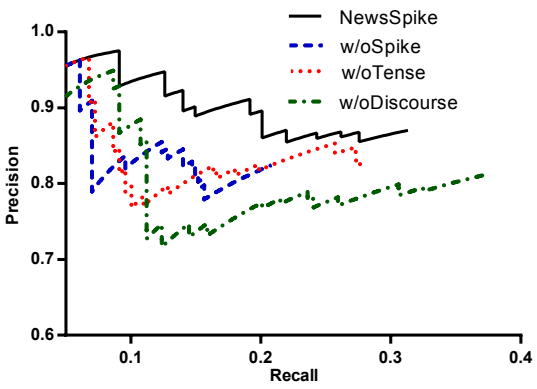


Figure 4: Precision recall curves on hard, diverse cases for NewsSpike, w/oSpike, w/oTense and w/oDiscourse.

the curves because they are drawn after sorting the predictions by the value of the corresponding ILP objective functions, which do not perfectly reflect prediction accuracy. However, it is clear that NEWSSPIKE produces greater precision over all ranges of recall.

5.3 Comparison with Methods using the Distributional Hypothesis

We evaluated our model against methods based on the distributional hypothesis. We ran NEWSSPIKE over all EEC-sets except for the development set and compared to the following systems:

Resolver: Resolver (Yates and Etzioni, 2009) uses a set of extraction tuples in the form of (a_1, r, a_2) as the input and creates a set of relation clusters as the output paraphrases. Resolver also produces argument clusters, but this paper only evaluates relation clustering. We evaluated Resolver's performance with an input of the 2.6 million extractions described in section 5.1, using Resolver's default parameters.

ResolverNYT: Since Resolver is supposed to perform better when given more accurate statistics from a larger corpus, we tried giving it more data. Specifically, we ran ReVerb on 1.8 million NY Times articles published between 1987 and 2007 obtain 60 million extractions (Sandhaus, 2008). We ran Resolver on the union of this and our standard test set, but report performance only on clusters whose relations were seen in our news stream.

System	all		diverse	
	prec	#rels	prec	#rels
Resolver	0.78	129	0.65	57
ResolverNyt	0.64	1461	0.52	841
ResolverNytTop	0.83	207	0.72	79
Cosine	0.65	17	0.33	9
CosineNyt	0.56	73	0.46	59
NEWSSPIKE	**0.93**	24843	**0.87**	5574

Table 2: Comparison with methods using the distributional hypothesis

ResolverNytTop: Resolver is designed to achieve good performance on its top results. We thus ranked the ResolverNYT outputs by their scores and report the precision of the top 100 clusters.

Cosine: Cosine similarity is a basic metric for the distributional hypothesis. This system employs the same setup as Resolver in order to generate paraphrase clusters, except that Resolver's similarity metric is replaced with the cosine. Each relation is represented by a vector of argument pairs. The similarity threshold to merge two clusters was 0.5.

CosineNYT: As for ResolverNYT, we ran CosineNYT with an extra 60 million extractions and reported the performance on relations seen in our news stream.

We measured the precision of each system by manually labeling all output if 100 or fewer clusters were generated (*e.g.* ResolverNytTop), otherwise 100 randomly chosen clusters were sampled. Annotators first determined the meaning of every output cluster and then created a gold cluster by choosing the correct relations. The gold cluster could be empty if the output cluster was nonsensical. Unlike many papers that simply report recall on the most frequent relations, we evaluated the total number of returned relations in the output clusters. As in Section 5.2, we also report numbers for the case of lexically diverse relation phrases.

As can be seen in Table 2, NEWSSPIKE outperformed methods based on the distributional hypothesis. The performance of the Cosine and CosineNyt was very low, suggesting that simple similarity metrics are insufficient for handling the paraphrasing problem, even when large-scale input is involved. Resolver and ResolverNyt employ an advanced similarity measurement and achieve better results. However, it is surprising that Resolver results in a greater precision than ResolverNyt. It

is possible that argument pairs from news streams spanning 20 years sometimes provide incorrect evidence for paraphrasing. For example, there were extractions like *(the Rangers, be third in, the NHL)* and *(the Rangers, be fourth in, the NHL)* from news in 2007 and 2003 respectively. Using these phrases, ResolverNyt produced the incorrect cluster {*be third in, be fourth in*}. NEWSSPIKE achieves greater precision than even the best results from ResolverNyt-Top, because NEWSSPIKE successfully captures the temporal heuristics, and does not confuse synonyms with antonyms, or causes with effects. NEWSSPIKE also returned on order of magnitude more relations than other methods.

5.4 Discussion

Unlike some domain-specific clustering methods, we tested on all relation phrases extracted by OpenIE on the collected news streams. There are no restrictions on the types of relations. Output paraphrases cover a broad range, including politics, sports, entertainment, health, science, etc. There are 10 thousand nonempty clusters over 17 thousand distinct phrases with average size 2.4. Unlike methods based on distributional similarity, NewsSpike correctly clusters infrequently appearing phrases.

Since we focus on high precision, it is not surprising that most clusters are of size 2 and 3. These high precision clusters can contribute a lot to generate larger paraphrase clusters. For example, one can invent the technique to merge smaller clusters together. The work presented here provides a foundation for future work to more closely examine these challenges.

While this paper gives promising results, there are still behaviors found in news streams that prove challenging. Many errors are due to the discourse context: the two sentences are synonymous in the given EEC-set, but the relation phrases are not paraphrases in general. For example, consider the following two sentences: *"DA14 narrowly misses Earth"* and *"DA14 flies so close to Earth"*. Statistics information from large corpus would be helpful to handle such challenges. Note in this paper, in order to fairly compare with the distributional hypothesis, we purposely forced NEWSSPIKE not to rely on any distributional similarity. But NEWSSPIKE's graphical model has the flexibility to incorporate any similarity metrics as features. Such a hybrid model has great potential to increase both precision and recall, which is one goal for future work.

6 Related Work

The vast majority of paraphrasing work falls into two categories: approaches based on the distributional hypothesis or those exploiting on correspondences between parallel corpora (Androutsopoulos and Malakasiotis, 2010; Madnani and Dorr, 2010).

Using Distribution Similarity: Lin and Pantel's (2001) DIRT employ mutual information statistics to compute the similarity between relations represented in dependency paths. Resolver (Yates and Etzioni, 2009) introduces a new similarity metric called the Extracted Shared Property (ESP) and uses a probabilistic model to merge ESP with surface string similarity.

Identifying the semantic equivalence of relation phrases is also called *relation discovery* or *unsupervised semantic parsing*. Often techniques don't compute the similarity explicitly but rely implicitly on the distributional hypothesis. Poon and Domingos' (2009) USP clusters relations represented with fragments of dependency trees by repeatedly merging relations having similar context. Yao *et al.* (2011; 2012) introduces generative models for relation discovery using LDA-style algorithm over a relation-feature matrix. Chen *et al.* (2011) focuses on domain-dependent relation discovery, extending a generative model with meta-constraints from lexical, syntactic and discourse regularities.

Our work solves a major problem with these approaches, avoiding errors such as confusing synonyms with antonyms and causes with effects. Furthermore, NEWSSPIKE doesn't require massive statistical evidence as do most approaches based on the distributional hypothesis.

Using Parallel Corpora: Comparable and parallel corpora, including news streams and multiple translations of the same story, have been used to generate paraphrases, both sentential (Barzilay and Lee, 2003; Dolan et al., 2004; Shinyama and Sekine, 2003) and phrasal (Barzilay and McKeown, 2001; Shen et al., 2006; Pang et al., 2003). Typical methods first gather relevant articles and then pair sentences that are potential paraphrases. Given a training set of paraphrases, models are learned and applied to unlabeled pairs (Dolan and Brockett, 2005;

Socher et al., 2011). Phrasal paraphrases are often obtained by running an alignment algorithm over the paraphrased sentence pairs.

While prior work uses the temporal aspects of news streams as a coarse filter, it largely relies on text metrics, such as context similarity and edit distance, to make predictions and alignments. These metrics are usually insufficient to produce high precision results; moreover they tend to produce paraphrases that are simple lexical variants (*e.g.* {*go to, go into*}.). In contrast, NEWSSPIKE generates paraphrase clusters with both high precision and high diversity.

Others: Textual entailment (Dagan et al., 2009), which finds a phrase implying another phrase, is closely related to the paraphrasing task. Berant *et al.* (2011) notes the flaws in distributional similarity and proposes local entailment classifiers, which are able to combine many features. Lin *et al.* (2012) also uses temporal information to detect the semantics of entities. In a manner similar to our approach, Recasens *et al.* (2013) mines parallel news stories to find opaque coreferent mentions.

7 Conclusion

Paraphrasing event relations is crucial to many natural language processing applications, including relation extraction, question answering, summarization, and machine translation. Unfortunately, previous approaches based on distribution similarity and parallel corpora, often produce low precision clusters. This paper introduces three Temporal Correspondence Heuristics that characterize semantically equivalent phrases in news streams. We present a novel algorithm, NEWSSPIKE, based on a probabilistic graphical model encoding these heuristics, which harvests high-quality paraphrases of event relations.

Experiments show NEWSSPIKE's improvement relative to several other methods, especially at producing lexically diverse clusters. Ablation tests confirm that our temporal features are crucial to NEWSSPIKE's precision. In order to spur future research, we are releasing an annotated corpus of time-stamped news articles and our harvested relation clusters.

Acknowledgments

We thank Oren Etzioni, Anthony Fader, Raphael Hoffmann, Ben Taskar, Luke Zettlemoyer, and the anonymous reviewers for providing valuable advice. We also thank Shengliang Xu for annotating the datasets. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, ONR grant N00014-12-1-0211, a gift from Google, and the WRF / TJ Cable Professorship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. In *Journal of Artificial Intelligence Research*, pages 135–187.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL*, pages 16–23. Association for Computational Linguistics.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *ACL*, pages 50–57. Association for Computational Linguistics.

Regina Barzilay, Kathleen R McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *ACL*, pages 550–557. Association for Computational Linguistics.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *ACL-HLT*, pages 610–619. Association for Computational Linguistics.

Rahul Bhagat and Deepak Ravichandran. 2008. Large scale acquisition of paraphrases for learning surface patterns. In *ACL*, volume 8, pages 674–682. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *NAACL*, pages 17–24. Association for Computational Linguistics.

Harr Chen, Edward Benson, Tahira Naseem, and Regina Barzilay. 2011. In-domain relation discovery with meta-constraints via posterior regularization. In *ACL-HLT*, pages 530–540. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *ACL*, pages 1–8. Association for Computational Linguistics.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(04):i–xvii.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of IWP*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Computational Linguistics*, page 350. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*. Association for Computational Linguistics, July 27-31.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*. Association for Computational Linguistics.

William A Gale, Kenneth W Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics.

Zellig S Harris. 1954. Distributional structure. *Word*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL-HLT*, pages 541–550.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *WSDM*, pages 441–450. ACM.

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

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *EMNLP*, pages 893–903. Association for Computational Linguistics.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *NAACL*, pages 102–109. Association for Computational Linguistics.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*, pages 1–10. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *EMNLP*, pages 968–977. Association for Computational Linguistics.

Marta Recasens, Matthew Can, and Dan Jurafsky. 2013. Same referent, different words: Unsupervised mining of opaque coreferent mentions. In *Proceedings of NAACL-HLT*, pages 897–906.

Evan Sandhaus. 2008. *The New York Times annotated corpus*. Linguistic Data Consortium.

Siwei Shen, Dragomir R Radev, Agam Patel, and Güneş Erkan. 2006. Adding syntax to dynamic programming for aligning comparable texts for the generation of paraphrases. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 747–754. Association for Computational Linguistics.

Yusuke Shinyama and Satoshi Sekine. 2003. Paraphrase acquisition for information extraction. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 65–71. Association for Computational Linguistics.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *NIPS*, 24:801–809.

Mihalis Yannakakis. 1992. On the approximation of maximum satisfiability. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, SODA '92, pages 1–9.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *EMNLP*, pages 1456–1466. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *ACL*, pages 712–720. Association for Computational Linguistics.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34(1):255.

Relational Inference for Wikification

Xiao Cheng **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{cheng88,danr}@illinois.edu

Abstract

Wikification, commonly referred to as Disambiguation to Wikipedia (D2W), is the task of identifying concepts and entities in text and disambiguating them into the most specific corresponding Wikipedia pages. Previous approaches to D2W focused on the use of local and global statistics over the given text, Wikipedia articles and its link structures, to evaluate context compatibility among a list of probable candidates. However, these methods fail (often, embarrassingly), when some level of text understanding is needed to support Wikification. In this paper we introduce a novel approach to Wikification by incorporating, along with statistical methods, richer relational analysis of the text. We provide an extensible, efficient and modular Integer Linear Programming (ILP) formulation of Wikification that incorporates the entity-relation inference problem, and show that the ability to identify relations in text helps both candidate generation and ranking Wikipedia titles considerably. Our results show significant improvements in both Wikification and the TAC Entity Linking task.

1 Introduction

Wikification (D2W), the task of identifying concepts and entities in text and disambiguating them into their corresponding Wikipedia page, is an important step toward supporting deeper textual understanding, by augmenting the ability to ground text in existing knowledge and facilitating knowledge expansion.

D2W has been studied extensively recently (Cucerzan, 2007; Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ferragina and Scaiella, 2010; Ratinov et al., 2011) and has already found broad applications in NLP, Information Extraction, and Knowledge Acquisition from text, from coreference resolution (Ratinov and Roth, 2012) to entity linking and knowledge population (Ellis et al., 2011; Ji et al., 2010; Cucerzan, 2011).

Given a document D containing a set of concept and entity mentions M (referred to later as *surface*), the goal of Wikification is to find the most accurate mapping from mentions to Wikipedia *titles* T; this mapping needs to take into account our understanding of the text as well as background knowledge that is often needed to determine the most appropriate title. We also allow a special NIL title that captures all mentions that are outside Wikipedia.

Earlier approaches treated this task as a word-sense disambiguation (WSD) problem, which was later enhanced with a certain level of global reasoning, but essentially all approaches focused on generic statistical features in order to achieve robust disambiguation. It was shown that by disambiguating to the most likely title for every surface, independently maximizing the conditional probability $\Pr(title|surface)$, we already achieve a very competitive baseline on several Wikification datasets (Ratinov et al., 2011). This strong statistical baseline makes use of the relatively comprehensive coverage of the existing Wikipedia links from surface strings to Wikipedia titles. Although more involved statistical features are required in order to make substantial improvements, global features such as context TF-IDF, better string similarity, etc., statistics-based Wikification systems give a fairly coherent set of disambiguation when sufficient context is available. Consider the following example: *Earth's biosphere*

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1787–1796,
Seattle, Washington, USA, 18-21 October 2013.

*then significantly altered the atmospheric and other basic physical conditions, which enabled the proliferation of organisms. The **atmosphere** is composed of 78.09% nitrogen, 20.95% oxygen, 0.93% argon, 0.039% carbon dioxide, and small amounts of...*

The baseline system we adopted (Ratinov et al., 2011), one of the best Wikification systems, already disambiguates **atmosphere** correctly to the title *Earth's atmosphere* instead of the more general title *Atmosphere*, making use of the concept *Earth* in its local context to resolve the mention to the more specific title that better coheres with the topic. However, consider the following example:

Ex. 1 "As **Mubarak**, the wife of deposed Egyptian President Hosni Mubarak got older, her influence..."

The bold faced name should be mapped to *Suzanne Mubarak*, but all existing Wikification systems map both names in this sentence to the dominant page (the most linked page) of *Hosni Mubarak*, failing to understand the relation between them, which should prevent them from being mapped to the same page. A certain level of text understanding is required even to be able to generate a good list of title candidates. For example, in:

Ex. 2 "...ousted long time Yugoslav President Slobodan Milošević in October. Mr. Milošević's **Socialist Party**..."

the bold-faced concept should be mapped to the page of the *Socialist Party of Serbia*, which is far down the list of titles that could be related to "Socialist Party"; making this title a likely candidate requires understanding the possessive relation with Milošević and then making the knowledge-informed decision that he is more related to *Socialist Party of Serbia* than any other possible titles. Finally, in

Ex. 3 "James Senn, director of **Robinson College**'s Center for Global Business Leadership at Georgia State University..."

we must link *Robinson College* to *J. Mack Robinson College of Business* which is located at *Georgia State University* instead of *Robinson College, Cambridge*, which is the only probable title linked by the surface *Robinson College* in the version of the Wikipedia dump we used.

These examples further illustrate that, along with understanding the relation expressed in the text, we need to access background knowledge sources and to deal with variability in surface representation across the text, Wikipedia, and knowledge, in order to reliably address the Wikification problem.

In this paper we focus on understanding those natural language constructs that will allow eliminating these "obvious" (to a human reader) mistakes from Wikification. In particular, we focus on resolving coreference and a collection of local syntactico-semantic relations (Chan and Roth, 2011); better understanding the relational structure of the text allows us to generate title candidates more accurately given the text, rank these candidates better and determine when a mention in text has no corresponding title in Wikipedia and should be mapped to NIL, a key problem in Wikification. Moreover, it allows us to access external knowledge based resources more effectively in order to support these decisions.

We incorporate the outcome of our relational analysis, along with the associated features extracted from external sources and the "standard" wikification statistical features, into an ILP-based inference framework that globally determines the best assignment of mentions to titles in a given document. We show that by leveraging a better understanding of the textual relations, we can substantially improve the Wikification performance. Our system significantly outperforms all the top Wikification systems on the widely adopted standard datasets and shows state-of-the-art results when evaluated (without being trained directly) on the TAC 2011 Entity Linking task.

2 The Wikification Approach

A general Wikification decision consists of three computational components: (1) generating a ranked list of title candidates for each mention, (2) ranking candidates globally, and (3) dealing with NIL mentions. For (1), the "standard" way of using $\Pr(title|surface)$ is often not sufficient; consider the case where the mention is the single word "*President*"; disambiguating such mentions depends heavily on the context, i.e. to determine the relevant country or organization. However, it is intractable to search the entire surface-to-title space, and using an arbitrary top-K list will inevitably leave out a large number of potential solutions. For (2), even though

the anchor texts cover many possible ways of paraphrasing the Wikipedia article titles and thus using the top $\Pr(title|surface)$ is proven to be a fairly strong baseline, it is never comprehensive. There is a need to disambiguate titles that were never linked by any anchor text, and to disambiguate mentions that have never been observed as the linked text. For (3) the Wikifier needs to determine when a mention corresponds to no title, and map it to a NIL entity. Simply training a classifier using coherency features or topical models turns out to be insufficient, since it has a predetermined granularity at which it can distinguish entities.

Next we provide a high-level description (Alg. 1) of our approach to improve Wikification by leveraging textual relations in these three stages.

Algorithm 1 Relational Inference for Wikification

*N*ote: $\Gamma : M \rightarrow T$ is the sought after mapping from all mentions in the document to all candidate titles in Wikipedia.

Require: Document D, Knowledge Base K consisting of relation triples $\sigma = (t_a, p, t_b)$, where p is the relation predicate.

1: Generate initial mentions $M = \{m_i\}$ from D.
2: Generate candidates $t_i = \{t_i^k\}$ for mention m_i and initialize candidate priors $\Pr(t_i^k|m_i)$ with existing Wikification system, for all $m_i \in M$.
3: Instantiate non-coreference relational constraints and add relational candidates.
4: Instantiate coreference relational constraints and add relational candidates.
5: Construct an ILP objective function and solve for the $\arg\max_\Gamma \Pr(\Gamma)$.
6: **return** Γ.

Most of our discussion addresses the relational analysis and its impact on stage (2) and (3) above. We will only briefly discuss improvements to the standard candidate generation stage in Sec. 4.4

3 Problem Formulation

We now describe how we formulate our global decision problem as an Integer Linear Program (ILP).

We use two types of boolean variables: e_i^k is used to denote whether we disambiguate m_i to t_i^k ($\Gamma(m_i) = t_i^k$) or not. $r_{ij}^{(k,l)}$ is used to denote if titles t_i^k and t_j^l are chosen simultaneously, that is, $r_{ij}^{(k,l)} = e_i^k \wedge e_j^l$.

Our models determine two types of score for the boolean variables above: $s_i^k = \Pr(e_i^k) = \Pr(\Gamma(m_i) = t_i^k)$, represents the initial score for the kth candidate title being chosen for mention m_i. For a pair of titles (t_i^k, t_j^l), we denote the confidence of finding a relation between them by $w_{ij}^{(k,l)}$. Its value depends on the textual relation type and on how coherent it is with our existing knowledge.

Our goal is to find the best assignment to variables e_i^k, such that it satisfies some legitimacy (hard) constraints and the soft constraints dictated by the relational constraints (via scores $w_{ij}^{(k,l)}$). To accomplish that we define our objective function as a Constrained Conditional Model (CCM) (Roth and Yih, 2004; Chang et al., 2012) that is used to reward or penalize a pair of candidates t_i^k, t_j^l by $w_{ij}^{(k,l)}$ when they are chosen in the same document. Specifically, we choose the assignment Γ_D that optimizes:

$$\Gamma_D = \arg\max_\Gamma \sum_i \sum_k s_i^k e_i^k + \sum_{i,j} \sum_{k,l} w_{ij}^{(k,l)} r_{ij}^{(k,l)}$$

$$\begin{aligned}
s.t. \quad & r_{ij}^{(k,l)} \in \{0,1\} && \text{Integral constraints} \\
& e_i^k \in \{0,1\} && \text{Integral constraints} \\
& \forall i \textstyle\sum_k e_i^k = 1 && \text{Unique solution} \\
& 2r_{ij}^{(k,l)} \leq e_i^k + e_j^l && \text{Relation definition}
\end{aligned}$$

Note that as in most NLP problems, the problem is very sparse, resulting in a tractable ILP that is solved quickly by off-the-shelf ILP packages (Gurobi Optimization, 2013). In our case the key reason for the sparseness is that $w_{ij}^{(k,l)} = 0$ for most pairs considered, which does not require explicit instantiation of $r_{ij}^{(k,l)}$.

4 Relational Analysis

The key challenge in incorporating relational analysis into the Wikification decision is to systematically construct the relational constraints (the solid edges between candidates in Figure 1) and incorporate them into our inference framework. Two main components are needed: first, we need to extract high precision textual relations from the text; then,

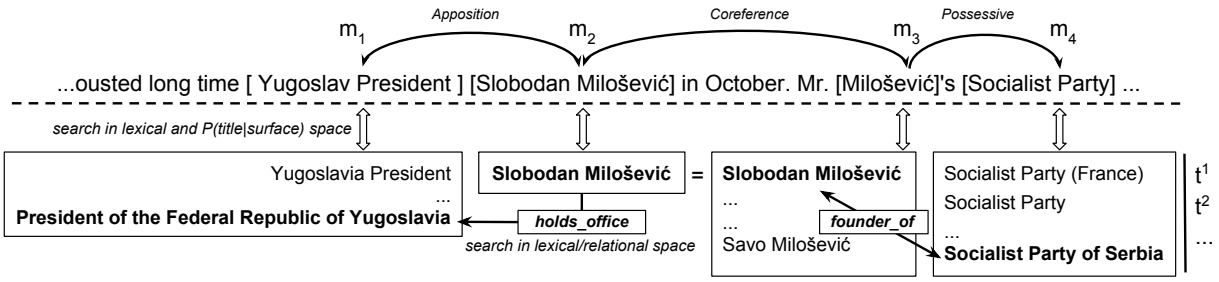


Figure 1: Textual relation inference framework: The goal is to maximize the objective function assigning mentions to titles while enforcing coherency with relations extracted from both text and an external knowledge base. Here, searching the external KB reveals that *Slobodan Milošević* is the founder of the *Socialist Party of Serbia*, which can be referred to by the surface *Socialist Party*; we therefore reward the output containing this pair of candidates. The same idea applies for the relation "*Slobodan Milošević* holds office as *President of the Federal Republic of Yugoslavia*" as well as to the coreference relation between two mentions of *Slobodan Milošević*.

we need to assign weights to these semantic relations. We determine the weights by combining type and confidence of the relation extracted from text with the confidence in relations retrieved from an external Knowledge Base (KB) by using the mention pairs as a query. It is noteworthy that although context window based coherency objective functions capture many proximity relations, using these unfiltered relations as constraints in our experiments introduced excessive amount of false-positives for the intrinsically sparse textual relations and resulted in severe performance hit.

In Sec. 4.1 we describe how we extract relations from text; our goal is to reliably identify *arguments* that we hypothesize to be in a relation; we show that this is essential both to our candidate generation, our ranking and the mapping to NIL. Sec. 4.2 describes how we use an external KB to verify that these arguments are indeed in a relation. Finally, Sec. 4.3 shows how we generate scores for the mentions and relations, as coefficients in the objective function of Sec. 3. The process is illustrated in Figure 1. Overall, our approach is an ambiguity-aware approach that identifies, filters and scores the relevant relations; this is essential due to the ambiguity, variability and noise inherent in directly matching surface forms to titles.

4.1 Relation Extraction

Even though relation extraction is an open problem, analysis on the ACE2004 Relation Detection and Characterization (RDC) dataset shows that approximately 80% of the relations are expressed through syntactico-semantic structures (Chan and Roth, 2011) that are easy to extract with high precision. Unlike the general ACE RDC task, we can restrict relation arguments to be named entities and thus leverage the large number of known relations in existing databases (e.g. Wikipedia infoboxes). We also consider conference relations that potentially aid mapping different mentions to the same title.

4.1.1 Syntactico-semantic Relations

We introduce our approach using the following example. Consider a news article discussing Israeli politics while briefly mentioning:

Ex. 4 An official at the [Iranian]$_1$ [Ministry of Defense]$_2$ told Tehran Radio that...

A purely statistical approach would very likely map the entity *[Ministry of Defense]*$_2$ to *Ministry of Defense (Israel)* instead of *Ministry of Defense and Armed Forces Logistics (Iran)* because the context is more coherent with concepts related to Israel rather than to Iran. Nevertheless, the pre-modifier relation between *[Iranian]*$_1$ and *[Ministry of Defense]*$_2$ demands the answer to be tightly related to Iran. Even though human readers may not know the correct title needed here, understanding the pre-modifier relation allows them to easily filter through a list of candidates and enforce constraints that are derived jointly from the relation expressed in the text and their background knowledge.

In our attempt to mimic this general approach, we employ several high precision classifiers to resolve

a range of local relations that are used to retrieve relevant background knowledge, and consequently integrated into our inference framework. Our input for relation extraction is any segment matched by the regular expression to be mentioned in section 4.4 in the candidate generation stage; we analyze its constituents by decomposing it into the two largest sub-entities that have (in Wikipedia) corresponding candidates. In the above example, *Iranian Ministry of Defense* would be decomposed into **Iranian** and **Ministry of Defense** and our relation extraction process hypothesizes a relation between these arguments.

Note that we do not use any full parsing since it does not address our needs directly nor does it scale well with the typical amount of data used in Wikification.

4.1.2 Coreference Relations

In addition to syntactico-semantic relations, we could also encounter other textual relations. The following example illustrates the importance of understanding co-reference relations in Wikification:

Ex. 5 [Al Goldman]$_1$, chief market strategist at A.G. Edwards, said ... [Goldman]$_2$ told us that...

There is no Wikipedia entry (or redirection) that matches the name *Al Goldman*. Clearly *[Goldman]*$_2$ refers to the same person and should be mapped to the same entity (or to NIL) rather than popular entities frequently referred to as *Goldman*, coherent with context or not, such as *Goldman Sachs*. To accomplish that, we cluster named entities that share tokens or are acronyms of each other when there is no ambiguity (e.g. no other longer named entity mentions containing *Goldman* in the document) and use a voting algorithm (Algorithm 2) to generate candidates locally from within the clusters. We also experimented with using full-fledged coference systems, but found it to be time consuming while providing no significant end-to-end performance difference.

4.1.3 Coreferent Nominal Mentions

Document level coreference also provides important relations between named entities and nominal mentions. Extracting these relations proved to be very useful for classifying NIL entities, as unfamiliar concepts tend to be introduced with these succinct appositional nominal mentions. These descriptions provide a clean "definition" of the entity, allowing us to abstract the inference to a limited "noun phrase entailment problem". That is, it allows us to determine whether the target mention corresponds to a candidate title. Consider, for example, wikifying *Dorothy Byrne* in: **Dorothy Byrne***, a state coordinator for the Florida Green Party, ...*
Identifying the apposition relation allows us to determine that this *Dorothy Byrne* is *not* the baseline Wikipedia title. We use the TF-IDF cosine similarity between the nominal description and the lexical context (Ratinov et al., 2011) of the candidate page, head word attributes and entity relation (i.e. between *Dorothy Byrne* and *Florida Green Party*) to determine whether any candidates of *Dorothy Byrne* can entail the nominal mention.

4.2 Relational Queries

Statistics based candidate generation algorithms always generate the same list of candidates given the same surface string; even though this approach has a competitive coverage rate, it will not work well in some "obvious" (to human) cases; for example, it offers very little information on highly ambiguous surface strings such as "President" for which it is even intractable to rank all the candidates. Top-K lists which were used in previous literature suffer from the same problem. Instead, we make use of relational queries to generate a more likely set of candidates.

Once mention pairs are generated from text using the syntactico-semantic structures and coreference, we use these to query our KB of relational triples. We first indexed all Wikipedia links and DBpedia relations as unordered triples $\sigma = (t_i, p, t_j)$, where the arguments t_i, t_j are tokenized, stemmed and lowercased for best recall. p is either a relation predicate from the DBpedia ontology or the predicate $LINK$ indicating a hyperlink relation. Since our baseline system has approximately 80% accuracy at this stage, it is reasonable to assume that at least one of the argument mentions is correctly disambiguated. Therefore we prune the search space by making only two queries for each mention pair (m_i, m_j): $q_0 = (t_i^*, m_j)$ and $q_1 = (m_i, t_j^*)$ where t_i^*, t_j^* are the strings representing the top titles chosen by the current model for mentions m_i, m_j re-

spectively.

We also aggressively prune the search results in a way similar to the process in Sec. 4.4, only keeping the arguments that are known to be possible or very likely candidates of the mention, based on the ambiguity that exists in the query result.

4.3 Relation Scoring

For the final assignment made using our objective function (Sec. 3) we need to normalize and rescale the output of individual components of our system as they come from different scoring functions. We consider adding new title candidates from two sources, through the coreference module and through the combined DBpedia and Wikipedia inter-page link structures. Next we describe how to compute and combine these scores.

4.3.1 Scoring Knowledge Base Relations

Our model uses both explicit relations $p \neq LINK$ from DBpedia and Wikipedia hyperlinks $p = LINK$ (implicit relation). We want to favor relations with explicit predicate, each weighted as ϕ implicit relation (we use $\phi = 5$ in our experiments, noting the results are insensitive to slight changes of this parameter).

For each query, we denote the score returned by our KB search engine[1] given query q and triple σ as $Sim_{\sigma,q}$. The relational weight $w_{i,j}^{k,l}$ between two candidates (see Sec. 3) is determined as:

$$w_{i,j}^{k,l} = \frac{1}{Z} \sum_{\sigma} \alpha_\sigma Sim_{\sigma,q}$$

where the sum is over the top 20 KB triples, α_σ is the relation type scaling constant (ϕ or 1), and Z is a normalization factor that normalizes all $w_{i,j}^{k,l}$ to the range $[0, 1]$.

Note that we do not check the type of the relation against the textual relation. The key reason is that explicit relations are not as robust, especially considering that we restrict one of the arguments in the relation and constraining the other argument's lexical form. Moreover, we back off to restricting the relations to be between known candidates when multiple lexically matched arguments are retrieved with high ambiguity. Additionally, most of our relations do not have explicit predicates in the text anyhow, and extracting a type would add noise to our decision.

[1] `http://lucene.apache.org/`

Algorithm 2 Coreferent Candidates Voting

Require: Coreference cluster C

Vote collector v_t denotes the score for a candidate t, which by default is 0.
$t_i = \{t_i^1 \dots t_i^n\}$ is the set of candidates of mention m_i.
l_i is the token count of m_i
for all $m_i \in C, l_i \geq 2$ **do**
  for all $t_i^k \in t_i$ **do**
    $v_{t_i^k} = v_{t_i^k} + s_i^k$
  end for
end for
Let $AllSingle$ denote whether $\forall i, l_i = 1$
for all $m_i \in C$ where $l_i = 1$ **do**
  for all $t_i^k \in t_i$ **do**
    if $AllSingle$ or $v_{t_i^k} > 0$ **then**
      $v_{t_i^k} = v_{t_i^k} + s_i^k$
    end if
  end for
end for
return v

4.3.2 Scoring Coreference Relations

For coreference relations, we simply use hard constraints by assigning candidates in the same coreference cluster a high relational weight, which is a cheap approximation to penalizing the output where the coreferent mentions disambiguate to different titles. In practice, using a weight of 10 is sufficient. Another important issue here is that the correct coreferent candidate might not exist in the candidate list of the shorter mentions in the cluster. For example, if a mention has the surface *Richard*, the number of potential candidates is so large that any top K list of titles will not be informative. We therefore ignore candidates generated from short surface strings and give it the same candidate list as the head mentions in its cluster. Figure 2 shows the voting algorithm we use to elect the potential candidates for the cluster.

The reason for separating the votes of longer and shorter mentions is that shorter mentions are inherently more ambiguous. Once a coreferent relation

is determined, longer mentions in the cluster should dictate what this cluster should collectively refer to.

4.4 Candidate Generation

Beyond the algorithmic improvements, the mention and candidate generation stage is aided by a few systematic preprocessing improvement briefly described below.

4.4.1 Mention Segmentation

Since named entities may sometimes overlap with each other, we use regular expressions to match longer surface forms that are often incorrectly segmented or ignored by NER [2] due to different annotation standards. For example, this will capture: *Prime Minister of the United Kingdom*. The regular expression pattern we used for Step 1 in Algorithm 1 simply adds mentions formed by any two consecutive capitalized word chunks connected by up to 2 punctuation marks, prepositions, and the tokens "the", "'s" & "and". These segments are also used as arguments for relation extraction.

4.4.2 Lexical Search

We link certain mentions directly to their exact matching titles in Step 3 when there is very low ambiguity. Specifically, when no title is known for a mention that is relatively long and fuzzily matches the lexically retrieved title, we perform this aggressive linking. The lexical similarity metrics are computed using the publicly available NESim [3] package (Do et al., 2009) with a threshold tuned on a subset of Wikipedia redirects, and by insisting that ORG type entities must have the same head word as the candidate titles. We only accept the link if there exists exactly one title in the lexical searching result after pruning.

5 Experiments and Evaluation

This section describes our experimental evaluation. We compare our system against the top D2W systems and perform several experiments to analyze and better understand the power of our approach. We based our work on the GLOW system from (Ratinov et al., 2011) to initialize the candidates and corresponding priors s_i^k in our objective function. Both the baseline system and our new system are publicly available [4].

5.1 Comparison with other Wikification systems

We first evaluate on the same 4 datasets[5] used in (Ratinov et al., 2011). The AQUAINT dataset, originally introduced in (Milne and Witten, 2008), resembles the Wikipedia annotation structure in that only the first mention of a title is linked, and is thus less sensitive to coreference capabilities. The MSNBC dataset is from (Cucerzan, 2007) and includes many mentions that do not easily map to Wikipedia titles due to rare surface or other idiosyncratic lexicalization (Cucerzan, 2007; Ratinov et al., 2011). Both of these datasets came from the news domain and do not contain any annotated NIL entities. The ACE and Wikipedia datasets are both taken from (Ratinov et al., 2011) where ACE is a subset of ACE2004 Coreference documents annotated by Amazon Mechanical Turkers in a similar standard as in AQUAINT but with NIL entities. The Wikipedia dataset is a sample of Wikipedia pages with its original hyperlink annotation.

The evaluation methodology Bag of Titles (BOT) F1 was used in both (Milne and Witten, 2008; Ratinov et al., 2011). For each document, the gold *bag* of titles is evaluated against our *bag* of system output titles requiring exact segmentation match.

	Dataset			
System	ACE	MSNBC	AQUAINT	Wiki
M&W	72.76	68.49	83.61	80.32
R&R	77.25	74.88	83.94	90.54
RI	**85.30**	**81.20**	**88.88**	**93.09**

Table 1: Performance on Wikification datasets, BOT F1 Performance. Our system, **Relational Inference (RI)** exhibits significant improvements over M&W (Milne and Witten, 2008) and R&R (Ratinov et al., 2011).

[2] We used the IllinoisNER package http://cogcomp.cs.illinois.edu/page/software_view/4

[3] http://cogcomp.cs.illinois.edu/page/software_view/22

[4] http://cogcomp.cs.illinois.edu/page/download_view/Wikifier

[5] http://cogcomp.cs.illinois.edu/page/resource_view/4

5.2 Ablation study

We incrementally add various components to the system and study their impact on the end performance. Due to the changes in Wikipedia since the datasets were generated, some of the pages no longer exist; in order to minimize the interference caused by these inconsistencies to an accurate evaluation of various componenents, we consider all non-NIL gold annotations that do not exist in the current Wikipedia index as NIL entities. Additionally in the MSNBC dataset, 127 out of 756 surface forms are known to be non-recallable. This explains the performance difference between the final rows in Tab. 1 and 2.

	Dataset			
Components	ACE	MSNBC	AQUAINT	Wiki
Baseline	80.68	83.00	83.93	91.93
+Lexical Match	83.47	84.13	88.88	93.41
+Coreference	83.40	87.88	88.88	93.09
RI	85.83	88.16	88.88	93.09

Table 2: Ablation study on Wikification datasets, BOT F1 Performance

The *Baseline* refers to the best performing configuration that was used in (Ratinov et al., 2011) except for using the current Wikipedia redirects. The *Lexical Match* refers to the applying solely the methodology introduced in Sec. 4.4. The *Coreference* performance includes all the inference performed without the KB triples, while the **Relational Inference (RI)** line represents all aspects of the proposed relational inference. It is clear that different datasets show somewhat different characteristics and consequently different gains from the various aspects of our approach but that, overall, all aspects contribute to improved performance.

5.3 TAC Entity Linking 2011

Next we evaluate our approach on the TAC English Entity Linking Task, which provides standardized evaluation metrics, allowing us to compare to a large number of other systems. We did not evaluate on the 2012 English Entity Linking due to the significant amount of ambiguous NIL entities included (Ellis et al., 2011) in the queries and the need to cluster them, which our D2W task definition does not address in depth. We compare our system with the Top 3 TAC 2011 systems (LCC, MS-MLI and NUSchime) as well as our baseline system GLOW that participated in TAC 2011 English Entity Linking (Ratinov and Roth, 2011) in table 3. The evaluation metric is the official modified B^3 and Micro-Average explained in (Ji et al., 2011).

Given the TAC Knowledge Base (TKB), which is a subset of the 2009 Wikipedia Dump, the TAC Entity Linking objective is to answer a named entity query string with either a TKB entry ID or a NIL entity ID, where the NIL entity IDs should be clustered across documents.

It is important to note that we did not retrain our system on the TAC data as the top three systems did, even though the objective function is slightly different. Instead, we ran our system on the TAC documents directly without any query expansion. For the final output of each query, we simply use the most confident candidate among all matched mentions. Due to the clustering requirement, we also trivially cluster NIL entities that either are mapped to the same out-of-KB Wikipedia URL or have the same surface form.

	Performance			
System	MA	B^3 P	B^3 R	B^3 F1
LCC	86.1	84.4	84.7	84.6
MS-MLI	86.8	84.8	83.4	84.1
RI	86.1	82.9	84.5	83.7
NUSchime	86.3	81.5	84.9	83.1
RI-0	81.4	78.6	79.1	78.8
Cogcomp	78.7	75.7	76.5	76.1

Table 3: TAC2011 Entity Linking performance. MA is Micro-Average. LLC (Monahan et al., 2011) is the best performing system in terms of B^3 F1 while MS-MLI (Cucerzan, 2011) is the best in terms of Micro-Average. Cogcomp (Ratinov and Roth, 2011) is the GLOW based system that participated in TAC 2011.**RI** is the complete relational inference system described in this paper; as described in the text, **RI** was not trained on the TAC data, unlike the other top systems.

We performed two runs on the TAC2011 data to study the effects of relational inference. The first run, **RI-0**, uses the current Wikipedia index and

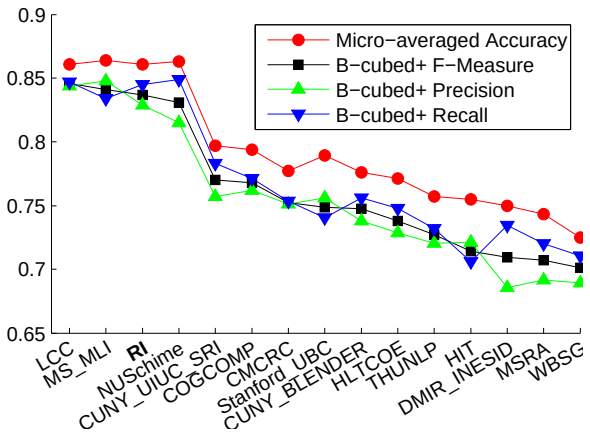


Figure 2: The **RI** compared with the other top 14 TAC2011 English Entity Linking systems ranked by modified B^3 F1 measure. Original figure from (Ji et al., 2011).

redirects for lexical matching without any inference, which scored 2.7% higher than the original GLOW system (Cogcomp). We can regard this performance as the new baseline that benefited from the fuzzy lexical matching capabilities that we have added, as well as the broader set of surface forms and redirects from the current Wikipedia dump. In the second run, **RI**, the complete relational inference described in this paper, scored 4.9% higher than the new baseline and sits on par with the top tier systems despite not being trained on the given data. The LCC system used sophisticated clustering algorithms trained on the TAC development set (Monahan et al., 2011). The second-ranked MS-MLI system relied on topic modeling, external web search engine logs as well as training on the development data (Cucerzan, 2011). This shows the robustness of our methods as well as the general importance of understanding textual relations in the task of Entity Linking and Wikification.

6 Related Work and Discussion

Earlier works on Wikification formulated the task as a WSD problem (Bunescu and Pasca, 2006; Mihalcea and Csomai, 2007) and focused primarily on training a model using local context. Later, various global statistical approaches were proposed to emphasize different coherence measures between the titles of the disambiguated mentions in the same document (Cucerzan, 2007; Milne and Witten, 2008; Ratinov et al., 2011). Built on top of the statistical models, our work focuses on leveraging deeper understanding of the text to more effectively and accurately utilize existing knowledge.

We have demonstrated that, by incorporating textual relations and semantic knowledge as linguistic constraints in an inference framework, it is possible to significantly improve Wikification performance. In particular, we have shown that our system is capable of making "intelligent" inferences that makes use of basic text understanding and has the ability to reason with it and verify it against relevant information sources. This allows our Relational Inference approach to resolve a variety of difficult examples illustrated in the Introduction.

Our system features high modularity since the relations are considered only at inference time; consequently, we can use any underlying Wikification system as long as it outputs a distribution of title candidates for each mention.

One possibility for future work is to supply this framework with a richer set of relations from the text, such as verbal relations. It will also be interesting to incorporate high-level typed relations and relax the relation arguments to be general concepts rather than only named entities.

Acknowledgments

We sincerely thank the three anonymous reviewers for their suggestions on the paper. This material is based on research sponsored by DARPA under agreement number FA8750-13-2-0008, and partly supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20155, by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053, and by the Multimodal Information Access & Synthesis Center at UIUC, part of CCICADA, a DHS Science and Technology Center of Excellence. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official

policies or endorsements, either expressed or implied, of DARPA, IARPA, DoI/NBC, ARL, or the U.S. Government.

References

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the European Chapter of the ACL (EACL)*.

Y. Chan and D. Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Portland, Oregon.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*, pages 708–716.

Silviu Cucerzan. 2011. Tac entity linking by performing full-document entity extraction and disambiguation. In *Proceedings of the Text Analysis Conference*.

Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. 2009. Robust, light-weight approaches to compute lexical similarity. Technical report, Computer Science Department, University of Illinois.

Joe Ellis, Xuansong Li, Kira Griffitt, Stephanie M Strassel, and Jonathan Wright. 2011. Linguistic resources for 2012 knowledge base population evaluations.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1625–1628, New York, NY, USA. ACM.

Inc. Gurobi Optimization. 2013. Gurobi optimizer reference manual.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the tac 2011 knowledge base population track. In *Fourth Text Analysis Conference (TAC 2011)*.

R. Mihalcea and A. Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 233–242.

D. Milne and I. H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 509–518.

Sean Monahan, John Lehmann, Timothy Nyberg, Jesse Plymale, and Arnold Jung. 2011. Cross-lingual cross-document coreference with entity linking. In *Proceedings of the Text Analysis Conference*.

L. Ratinov and D. Roth. 2011. Glow tac-kbp 2011 entity linking system. In *TAC*. Text Analysis Conference, 11.

L. Ratinov and D. Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *EMNLP*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.

Event Schema Induction with a Probabilistic Entity-Driven Model

Nathanael Chambers
United States Naval Academy
Annapolis, MD 21402
nchamber@usna.edu

Abstract

Event schema induction is the task of learning high-level representations of complex events (e.g., a *bombing*) and their entity roles (e.g., *perpetrator* and *victim*) from unlabeled text. Event schemas have important connections to early NLP research on frames and scripts, as well as modern applications like template extraction. Recent research suggests event schemas can be learned from raw text. Inspired by a pipelined learner based on named entity coreference, this paper presents the first generative model for schema induction that integrates coreference chains into learning. Our generative model is conceptually simpler than the pipelined approach and requires far less training data. It also provides an interesting contrast with a recent HMM-based model. We evaluate on a common dataset for template schema extraction. Our generative model matches the pipeline's performance, and outperforms the HMM by 7 F1 points (20%).

1 Introduction

Early research in language understanding focused on high-level semantic representations to drive their models. Many proposals, such as frames and scripts, used rich event schemas to model the situations described in text. While the field has since focused on more shallow approaches, recent work on schema induction shows that event schemas might be learnable from raw text. This paper continues the trend, addressing the question, can event schemas be induced from raw text without prior knowledge? We present a new generative model for event schemas, and it produces state-of-the-art induction results, including a 7 F1 point gain over a different generative proposal developed in parallel with this work.

Event schemas are unique from most work in information extraction (IE). Current relation discovery (Banko et al., 2007a; Carlson et al., 2010b) focuses on atomic facts and relations. Event schemas build relations into coherent event structures, often called *templates* in IE. For instance, an election template jointly connects that *obama won a presidential election* with *romney was the defeated*, *the election occurred in 2012*, and *the popular vote was 50-48*. The entities in these relations fill specific semantic roles, as in this template schema:

Template Schema for Elections
(*events*: nominate, vote, elect, win, declare, concede)

Date:	Timestamp
Winner:	Person
Loser:	Person
Position:	Occupation
Vote:	Number

Traditionally, template extractors assume foreknowledge of the event schemas. They know a Winner exists, and research focuses on supervised learning to extract winners from text. This paper focuses on the other side of the supervision spectrum. The learner receives no human input, and it first induces a schema before extracting instances of it.

Our proposed model contributes to a growing line of research in schema induction. The majority of previous work relies on ad-hoc clustering algorithms (Filatova et al., 2006; Sekine, 2006; Chambers and Jurafsky, 2011). Chambers and Jurafsky is a pipelined approach, learning events first, and later learning syntactic patterns as fillers. It requires

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807,
Seattle, Washington, USA, 18-21 October 2013.

several ad-hoc metrics and parameters, and it lacks the benefits of a formal model. However, central to their algorithm is the use of coreferring entity mentions to knit events and entities together into an event schema. We adapt this entity-driven approach to a single model that requires fewer parameters and far less training data. Further, experiments show state-of-the-art performance.

Other research conducted at the time of this paper also proposes a generative model for schema induction (Cheung et al., 2013). Theirs is not entity-based, but instead uses a sequence model (HMM-based) of verb clauses. These two papers thus provide a unique opportunity to compare two very different views of document structure. One is entity-driven, modeling an entity's role by its coreference chain. The other is clause-driven, classifying individual clauses based on text sequence. Each model makes unique assumptions, providing an interesting contrast. Our entity model outperforms by 7 F1 points on a common extraction task.

The rest of the paper describes in detail our main contributions: (1) the first *entity-based* generative model for schema induction, (2) a direct pipeline/formal model comparison, (3) results improving state-of-the-art performance by 20%, and (4) schema induction from the smallest amount of training data to date.

2 Previous Work

Unsupervised learning for information extraction usually learns binary relations and atomic facts. Models can learn relations like *Person is married to Person* without labeled data (Banko et al., 2007b), or rely on seed examples for ontology induction (*dog is a mammal*) and attribute extraction (*dogs have tails*) (Carlson et al., 2010b; Carlson et al., 2010a; Huang and Riloff, 2010; Durme and Pasca, 2008). These do not typically capture the deeper connections modeled by event schemas.

Algorithms that do focus on event schema extraction typically require both the schemas and labeled corpora, such as rule-based approaches (Chinchor et al., 1993; Rau et al., 1992) and modern supervised classifiers (Freitag, 1998; Chieu et al., 2003; Bunescu and Mooney, 2004; Patwardhan and Riloff, 2009; Huang and Riloff, 2011). Classifiers rely on the labeled examples' surrounding context for features (Maslennikov and Chua, 2007). Weakly supervised learning removes some of the need for labeled data, but most still require the event schemas. One common approach is to begin with unlabeled, but clustered event-specific documents, and extract common word patterns as extractors (Riloff and Schmelzenbach, 1998; Sudo et al., 2003; Riloff et al., 2005; Filatova et al., 2006; Patwardhan and Riloff, 2007; Chen et al., 2011). Bootstrapping with seed examples of known slot fillers has been shown to be effective (Yangarber et al., 2000; Surdeanu et al., 2006).

Shinyama and Sekine (2006) presented *unrestricted relation discovery* to discover relations in unlabeled documents. Their algorithm used redundant documents (e.g., all describe Hurricane Ivan) to observe repeated proper nouns. The approach requires many documents about the exact same event instance, and relations are binary (not schemas) over repeated named entities. Our model instead learns *schemas* from documents with mixed topics that don't describe the same event, so repeated proper nouns are less helpful.

Chen et al. (2011) perform relation extraction with no supervision on earthquake and finance domains. Theirs is a generative model that represents relations as predicate/argument pairs. As with others, training data is pre-clustered by event type and there is no schema connection between relations.

This paper builds the most on Chambers and Jurafsky (2011). They learned event schemas with a three-stage clustering algorithm that included a requirement to retrieve extra training data. This paper removes many of these complexities. We present a formal model that uniquely models coreference chains. Advantages include a *joint* clustering of events and entities, and a formal probabilistic interpretation of the resulting schemas. We achieve better performance, and do so with far less training data.

Cheung et al. (2013) is most related as a generative formulation of schema induction. They propose an HMM-based model over latent event variables, where each variable generates the observed clauses. Latent schema variables generate the event variables (in the spirit of preliminary work by O'Connor (2012)). There is no notion of an entity, so learning uses text mentions and relies on the local HMM win-

message: id	dev-muc3-0112 (bellcore, mitre)
incident: date	10 mar 89
incident: location	peru: huanuco, ambo (town)
incident: type	bombing
incident: stage	accomplished
incident: instrument	explosive: "-"
perp: individual	"shining path members"
perp: organization	"shining path"

Figure 1: A subset of the slots in a MUC-4 template.

dow for event transitions. Their model was created in parallel with our work, and provides a nice contrast in both approach and results. Ours outperforms their model by 20% on a MUC-4 evaluation.

In summary, this paper extends most previous work on event schema induction by removing the supervision. Of the recent 'unsupervised' work, we present the first entity-driven generative model, and we experiment on a mixed-domain corpus.

3 Dataset: The MUC-4 Corpus

The corpus from the Message Understanding Conference (MUC-4) serves as the challenge text (Sundheim, 1991), and will ground discussion of our model. MUC-4 is also used by the closest previous work. It contains Latin American newswire about terrorism events, and it provides a set of hand-constructed event schemas that are traditionally called template schemas. It also maps labeled templates to the text, providing a dataset for template extraction evaluations. Until very recently, only extraction has been evaluated. We too evaluate our model through extraction, but we also compare our learned schemas to the hand-created template schemas. An example of a filled in MUC-4 template is given in Figure 1.

The MUC-4 corpus defines six template types: **Attack**, **Kidnapping**, **Bombing**, **Arson**, **Robbery**, and **Forced Work Stoppage**. Documents are often labeled with more than one template and type. Many include multiple events at different times in different locations. The corpus is particularly challenging because template schemas are inter-mixed and entities can play multiple roles across instances.

The training corpus contains 1300 documents, 733 of which are labeled with at least one schema. 567 documents are not labeled with any schemas. These unlabeled documents are articles that report on non-specific political events and speeches. They make the corpus particularly challenging. The development and test sets each contain 200 documents.

4 A Generative Model for Event Schemas

This paper's model is an entity-based approach, similar in motivation to Haghighi and Klein (2010) and the pipelined induction of Chambers and Jurafsky (2011). Coreference resolution guides the learning by providing a set of pre-resolved entities. Each entity receives a schema role label, so it allows *all mentions* of the entity to inform that role choice. This important constraint links coreferring mentions to the same schema role, and distinguishes our approach from others (Cheung et al., 2013).

4.1 Illustration

The model represents a document as a set of entities. An entity is a set of entity mentions clustered by coreference resolution. We will use the following two sentences for illustration:

> *A truck bomb exploded near the embassy.*
> *Three militia planted it, and then they fled.*

This text contains five entity mentions. A perfect coreference resolution system will resolve these five mentions into three entities:

Entity Mentions	**Entities**	**Roles**
a truck bomb	(a truck bomb, it)	Instrument
the embassy	(the embassy)	Target
three militia	(three militia, they)	Perpetrator
it		
they		

The schema roles, or template *slots*, are the type of target knowledge we want to learn. Each entity will be labeled with both a slot variable s and a template variable t (e.g., the *s=perpetrator* of a *t=bombing*). The lexical context of the entity mentions guides the learning model to this end.

4.2 Definitions

A document $d \in D$ is represented as a set of entities E_d. Each entity $e \in E_d$ is a triple: $e = (h, M, F)$

1. h_e is the canonical word for the entity (typically the first mention's head word)

Text

A truck bomb exploded near the embassy.
Three militia planted it, and then they fled.

Entity Representation

entity 1: h = *bomb*, F = {PHYS-OBJ},
M = { (p=explode, d=subject-explode)
(p=plant, d=object-plant) }
entity 2: h = *militia*, F = {PERSON, ORG},
M = { (p=plant, d=subject-plant),
(p=flee, subject-flee) }
entity 3: h = *embassy*, F = {PHYS-OBJ, ORG},
M = { (p=explode, d=prep_near-explode) }

Figure 2: Example text mapped to our entities.

2. M_e is a set of entity mentions $m \in M_e$. Each mention is a pair $m = (p, d)$: the predicate, and the typed dependency from the predicate to the mention (e.g., *push* and *subject-push*).

3. F_e is a set of binary entity features. This paper only uses named entity types as features, but generalizes to other features as well.

A document is thus reduced to its entities, their grammatical contexts, and entity features. Figure 2 continues our example using this formulation. h_e is chosen to be e's longest non-pronoun mention $m \in M_e$. Mentions are labeled with NER and WordNet synsets to create an entity's features $F_e \subseteq \{$*Person, Org, Loc, Event, Time, Object, Other*$\}$. We use the Stanford NLP toolkit to parse, extract typed dependencies, label with NER, and run coreference.

4.3 The Generative Models

Similar to topics in LDA, each document d in our model has a corresponding multinomial over *schema types* θ_d, drawn from a Dirichlet. For each entity in the document, a hidden variable t is drawn according to θ_d. These t variables represent the high level schema types, such as *bombing* or *kidnapping*. The predicates associated with each of the entity's mentions are then drawn from the schema's multinomial over predicates P_t. The variable t also generates a hidden variable s from its distribution over slots, such as *perpetrator* and *victim*. Finally, the entity's canonical head word is generated from β_s, all entity mentions' typed dependencies from δ_s, and named entity types from γ_s.

The most important characteristic of this model is the separation of event words from the lexical properties of specific entity mentions. The schema type variables t only model the distribution of event words (bomb, plant, defuse), but the slot variables s model the syntax (subject-bomb, subject-plant, object-arrest) and entity words (suspect, terrorist, man). This allows the high-level schemas to first select predicates, and then forces predicate arguments to prefer slots that are in the parent schema type.

Formally, a document d receives a labeling Z_d where each entity $e \in E_d$ is labeled $Z_{d,e} = (t, s)$ with a schema type t and a slot s. The joint distribution of a document and labeling is then as follows:

$$
\begin{aligned}
P(d, Z_d) = & \prod_{e \in E_d} P(t|\theta) \times P(s|t) \\
& \times \prod_{e \in E_d} P(h_e|s) \\
& \times \prod_{e \in E_d} \prod_{f \in F_e} P(f|s) \\
& \times \prod_{e \in E_d} \prod_{m \in M_e} P(d_m|s) * P(p_m|t) \qquad (1)
\end{aligned}
$$

The plate diagram for the model is given in Figure 3. The darker circles correspond to the observed entity components in Figure 2. We assume the following generative process for a document d:

Generate θ_d from Dir(α)
for each schema type $t = 1...m$ **do**
  Generate P_t from Dir(η)
  for each slot $s_t = 1...k$ **do**
    Generate β_s from Dir(μ)
    Generate γ_s from Dir(ν)
    Generate δ_s from Dir(φ)
for each entity $e \in E_d$ **do**
  Generate schema type t from Multinomial(θ_d)
  Generate slot s from UniformDist(k)
  Generate head word h from Multinomial(β_s)
  for each mention $m \in M_e$ **do**
    Generate predicate token p from Multinomial(P_t)
    Generate typed dependency d from Multinomial(δ_s)
  for each entity type $i = 1..|F_e|$ **do**
    Generate entity type f from Multinomial(γ_s)

The number of schema types m and the number of slots per schema k are chosen based on training set performance.

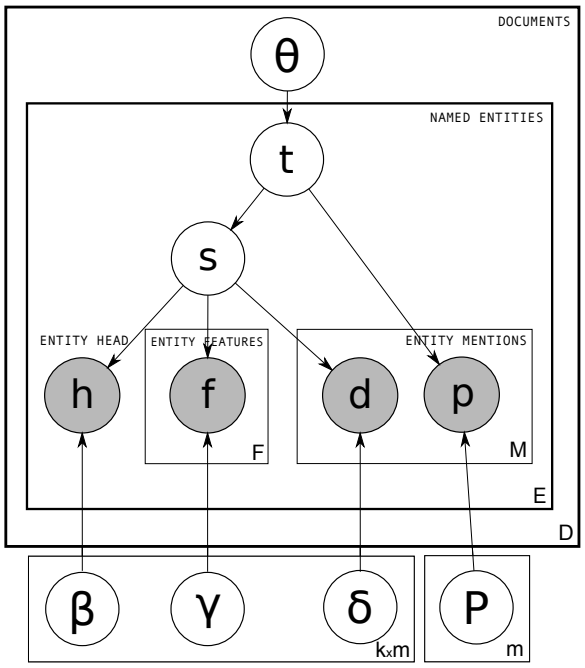


Figure 3: The full plate diagram for the event schema model. Hyper-parameters are omitted for readability.

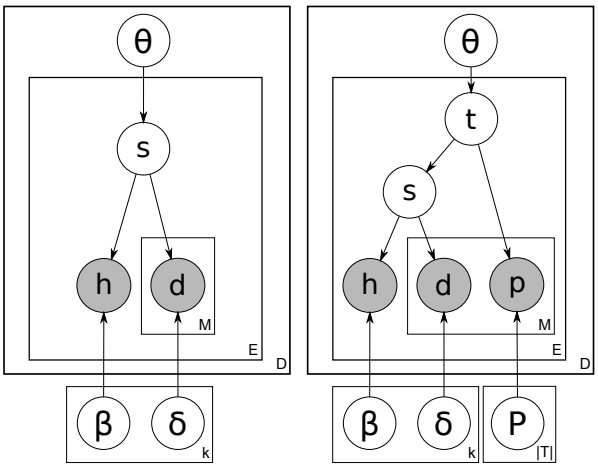


Figure 4: Simplified plate diagrams comparing the flat relation model to the full template model. The observed $f \in F$ variables are not included for clarity.

The Flat Relation Model

We also experiment with a *Flat Relation Model* that removes the hidden t variables, ignoring schema types. Figure 4 visually compares this flat model with the full model. We found that the predicate distribution P_t hurts performance in a flat model. Predicates are more informative at the higher level, but less so for slots where syntax is more important. We thus removed P_t from the model, and everything else remains the same. This flat model now learns a large set of k slots S that aren't connected by a high-level schema variable. Each slot $s \in S$ has a corresponding triple of multinomials (h, M, F) similar to above: (1) a multinomial over the head mentions β_s, (2) a multinomial over the grammatical relations of the entity mentions δ_s, and (3) a multinomial over the entity features γ_s. For each entity in a document, a hidden slot $s \in S$ is first drawn from Θ, and then the observed entity (h, M, F) is drawn according to the multinomials $(\beta_s, \gamma_s, \delta_s)$. We later evaluate this flat model to show the benefit of added schema structure.

4.4 Inference

We use collapsed Gibbs sampling for inference, sampling the latent variables $t_{e,d}$ and $s_{e,d}$ in sequence conditioned on a full setting of all the other variables (Griffiths and Steyvers, 2004). Initial parameter values are set by randomly setting t and s variables from the uniform distribution over schema types and slots, then computing the other parameter values based on these initial settings. The hyperparameters for the dirichlet distributions were chosen from a small grid search (see Experiments).

Beyond standard inference, we added one constraint to the model that favors grammatical distributions δ_s that do not contain conflicts. The subject and direct object of a verb should not both receive high probability mass under the same schema slot δ_s. For instance, the *victim* of a *kidnapping* should not favor both the subject and object of a single verb. Semantic roles should (typically) select one syntactic slot, so this constraint encourages that behavior. During sampling of $s_{e,d}$, we use a penalty factor λ to make conflicting relations less likely. Formally, $P(s_{e,d} = s|\theta, h_e, F_e, M_e) = \lambda$ iff there exists an $m \in M_e$ such that $P(m|\sigma_s) < P(inv(m)|\sigma_s)$ and $P(inv(m)|\sigma_s) > 0.1$, where $inv(m) = object$ if $m = subject$ and vice versa. Otherwise, the probability is computed as normal. We normalize the distributions after penalties are computed.

4.5 Entity Extraction for Template Filling

Inducing event schemas is only one benefit of the model. The learned model can also extract specific instances of the learned schemas without ad-

ditional complexity. To evaluate the effectiveness of the model, we apply the model to perform standard template extraction on MUC-4. Previous MUC-4 induction required an extraction algorithm separate from induction because induction created hard clusters (Chambers and Jurafsky, 2011). Cluster scores don't have a natural interpretation, so extraction required several parameters/thresholds to tune. Our model instead simply relies on model inference.

We run inference as described above and each entity receives a template label $t_{e,d}$ and a template slot label $s_{e,d}$. These labels are the extractions, and it requires no other parameters. The model thus requires far less machinery than a pipeline, and the experiments below further show that this simpler model outperforms the pipeline.

Beyond parameters, the question of "irrelevant" documents is a concern in MUC-4. Approximately half the corpus are documents that are not labeled with a template, so past algorithms required extra processing stages to filter out these irrelevant documents. Patwardhan and Riloff (2009) and Chambers and Jurafsky (2011) make initial decisions as to whether they should extract or not from a document. Huang and Riloff (2011) use a genre detector for this problem. Even the generative HMM-based model of Cheung et al. (Cheung et al., 2013) requires an extra filtering parameter. Our formal model is unique in not requiring additional effort. Ours is the only approach that doesn't require document filtering.

5 Evaluation Setup

Evaluating on MUC-4 has a diverse history that complicates comparison. The following balances comparison against previous work and enables future comparison to our results.

5.1 Template Schema Slots

Most systems do not evaluate performance on all MUC-4 template slots. They instead focus on four main slots, ignoring the parameterized slots that involve deeper reasoning (such as 'stage of execution' and 'effect of incident'). The four slots and example entity fillers are shown here:

Perpetrator:	Shining Path members
Victim:	Sergio Horna
Target:	public facilities
Instrument:	explosives

We also focus only on these four slots. We merged MUC's two perpetrator slots (individuals and orgs) into one gold Perpetrator. Previous work has both split the two and merged the two. We merge them because the distinction between an individual and an organization is often subtle and not practically important to analysts. This is also consistent with the most recent event schema induction in Chambers and Jurafsky (2011) and Cheung et al. (2013).

One peculiarity in MUC-4 is that some templates are labeled as optional (i.e., all its slots are optional), and some required templates contain optional slots (i.e., a subset of slots are optional). We ignore both optional templates and specific optional slots *when computing recall*, as in previous work (Patwardhan and Riloff, 2007; Patwardhan and Riloff, 2009; Chambers and Jurafsky, 2011).

Comparison between the extracted strings and the gold template strings uses head word scoring. We do not use gold parses for the text, so head words are defined simply as the rightmost word in the noun phrase. The exception is when the extracted phrase is of the form "A of B", then the rightmost word in "A" is used as the head. This is again consistent with previous work[1]. The standard evaluation metrics are precision, recall, and F1 score.

5.2 Mapping Learned Slots

Induced schemas need to map to gold schemas before evaluation. Which learned slots correspond to MUC-4 slots? There are two methods of mapping. The first ignores the schema type variables t, and simply finds the best performing s variable for each gold template slot[2]. We call this the *slot-only mapping* evaluation. The second approach is to map each template variable t to the best gold template type g, and limit the slot mapping so that only the slots under t can map to slots under g. We call this the *template mapping* evaluation. The slot-only mapping can result in higher scores since it is not constrained to preserve schema structure in the mapping.

Chambers and Jurafsky (2011) used template mapping in their evaluation. Cheung et al. (2013) used slot-only mapping. We run both evaluations in this paper and separately compare both.

[1] Personal communications with Patwardhan and Riloff

[2] bombing-victim is a template slot distinct from kidnap-victim. Both need to be mapped.

6 Experiments

We use the Stanford CoreNLP toolkit for text processing and parsing. We developed the models on the 1300 document MUC-4 training set. We then learned once on the entire 1700 training/dev/test set, and report extraction numbers from the inferred labels on the 200 document test set. Each experiment was repeated 10 times. Reported numbers are averaged across these runs.

There are two structure variables for the model: the number of schema types and the number of slots under each type. We searched for the optimal values on the training set before evaluating on test. The hyperparameters for all evaluations were set to $\alpha = \eta = \mu = \nu = 1$, $\varphi = .1$ based on a grid search.

6.1 Template Schema Induction

The first evaluation compares the learned schemas to the gold schemas in MUC-4.

Since most previous work assumes this knowledge ahead of time, we align our schemas with the main MUC-4 template types to measure quality. We inspected the learned event schemas that mapped to MUC-4 schemas based on the *template mapping* extraction evaluation.

Figure 5 shows some of the learned distributions for two mapped schemas: kidnappings and bombings. The predicate distribution for each event schema is shown, as well as the top 5 head words and grammatical relations for each slot. The words and events that were jointly learned in these examples appear quite accurate. The bombing and kidnap schemas learned all of the equivalent MUC-4 gold slots. Interestingly, our model also learned Locations and Times as important entities that appear in the text. These entities are not traditionally included in the MUC-4 extraction task.

Figure 6 lists the MUC-4 slots that we did and did not learn for the four most prevelant types. We report 71% recall, with almost all errors due to the model's failure to learn about arsons. Arson templates only occur in 40 articles, much less than the 200 bombing and over 400 attack. We show below that overall extraction performs well despite this. The learned distributions for Attack end up extracting Arson perpetrators and Arson victims in the actual extraction evaluation.

	Bomb	Kidnap	Attack	Arson
Perpetrator	✓	✓	✓	x
Victim	✓	✓	✓	✓
Target	✓	-	✓	x
Instrument	✓	-	x	x
Location	✓	✓	✓	✓
Date/Time	✓	✓	✓	x

Figure 6: The MUC-4 gold slots that were learned. The bottom two are not in the traditional evaluation, but were learned by our model nonetheless.

Evaluation: Template Mapping

	Prec	Recall	F1
C & J 2011	**.48**	.25	.33
Formal Template Model	.42	**.27**	.33

Table 1: MUC-4 extraction with template mapping. A learned schema first maps to a gold MUC template. Learned slots can then only map to slots in that template.

6.2 Extraction Experiments

We now present the full extraction experiment that is traditionally used for evaluating MUC-4 performance. Although our learned schemas closely match gold schemas, extraction depends on how well the model can extract from diverse lexical contexts. We ran inference on the full training and test sets, and used the inferred labels as schema labels. These labels were mapped and evaluated against the gold MUC-4 labels as discussed in Section 5.

Performance is compared to two state-of-the-art induction systems. Since these previous two models used different methods to map their learned schemas, we compare separately. Table 1 shows the *template mapping* evaluation with Chambers and Jurafsky (C&J). Table 2 shows the *slot-only mapping* evaluation with Cheung et al.

Our model achieves an F1 score comparable to C&J, and 20% higher than Cheung et al. Part of the greater increase over Cheung et al. is the mapping difference. For each MUC-4 type, such as *bombing*, any four learned slots can map to the four MUC-4 bombing slots. There is no constraint that the learned slots must come from the same schema type. The more strict *template mapping* (Table 1) ensures that entire schema types are mapped together, and it reduces our performance from .41 to .33.

Kidnapping Entities

Victim (Person 88%)		**Perpetrator** (Person 62%, Org 30%)		**Date** (TimeDate 89%)	
businessman	object-kidnap	guerrilla	subject-kidnap	TIME	tmod-kidnap
citizen	object-release	ELN	subject-hold	February	prep_on-kidnap
Soares	prep_of-kidnapping	group	subject-attack	hours	tmod-release
Kent	possessive-release	extraditables	subject-demand	morning	prep_on-release
hostage	object-found	man	subject-announce	night	tmod-take

Bombing Entities

Victim (Person 86%, Location 8%)		**Physical Target** (Object 65%, Event 42%)		**Instrument** (Event 56%, Object 39%)	
person	object-kill	building	object-destroy	bomb	subject-explode
guerrilla	object-wound	office	object-damage	explosion	subject-occur
soldier	subject-die	explosive	object-use	attack	object-cause
man	subject-blow_up	station	and-office	charge	object-place
civilian	subject-try	vehicle	prep_of-number	device	subject-destroy

Figure 5: Select distributions for two learned events. Left columns are head word distributions β, right columns are syntactic relation distributions δ, and entity types in parentheses are the learned γ. Most probable words are shown.

Evaluation: Slot-Only Mapping

	Prec	Recall	F1
Cheung et al. 2013	.32	.37	.34
Flat Relation Model	.26	**.45**	.33
Formal Template Model	**.41**	.41	**.41**

Table 2: MUC-4 extraction with slot-only mapping. Any learned slot is allowed to map to any gold slot.

Entity Role Performance

	Prec	Recall	F1
Perpetrator	.40	.20	.26
Victim	.42	.31	.34
Target	.38	.28	.31
Instrument	.57	.39	.45

Table 3: Results for each MUC-4 template slot using the template-mapping evaluation.

The macro-level F1 scores can be broken down into individual slot performance. Table 3 shows these results ranging from .26 to .45. The Instrument role proves easiest to learn, consistent with C&J.

A large portion of MUC-4 includes irrelevant documents. Cheung et al. (2013) evaluated their model without irrelevant documents in the test set that to see how performance is affected. We compare against their numbers in Table 4. Results are closer now with ours outperforming .46 to .43 F1. This suggests that the HMM-based approach stumbles more on spurious documents, but performs better on relevant ones.

Gold Document Evaluation

	Prec	Recall	F1
Cheung et al. 2013	.41	**.44**	.43
Formal Template Model	**.49**	.43	**.46**

Table 4: Full MUC-4 extraction with *gold document classification*. These results ignore false positives extracted from "irrelevant" documents in the test set.

6.3 Model Ablation

Table 2 shows that the flat relation model (no latent type variables t) is inferior to the full schema model. F1 drops 20% without the explicit modeling of both schema types t and their entity slots s. The entity features F_e are less important. Experiments without them show a slight drop in performance (2 F1 points), small enough that they could be removed for efficiency. However, it is extremely useful to learn slots with NER labels like Person or Location.

Finally, we experimented without the subject/object constraint (Section 4.4). Performance drops 5-10% depending on the number of schemas learned. Anecdotally, it merges too many schema slots that should be separate. We recommend using this constraint as it has little impact on CPU time.

6.4 Extension: Reduce Training Size

One of the main benefits of this generative model appears to be the reduction in training data. The pipelined approach in C&J required an information retrieval stage to bring in hundreds of other docu-

ments from an external corpus. This paper's generative model doesn't require such a stage.

We thus attempted to induce and extract event schemas from just the 200 test set documents, with no training or development data. We repeated this experiment 30 times and averaged the results, setting the number of templates $t = 20$ and slots $s = 10$ as in the main experiment. The resulting F1 score for the template-mapping evaluation fell to 0.27 from the full data experiment of 0.33 F1. Adding more training documents in another experiment did not significantly increase performance over 0.27 until all training and development documents were included. This could be explained by the development set being more similar to the test set than training. We did not investigate further to prevent over-experimentation on test.

7 Discussion

Our model is one of the first generative formulations of schema induction. It produces state-of-the-art performance on a traditional extraction task, and performs with less training data as well as a more complex pipelined approach. Further, our unique entity-driven approach outperforms an HMM-based model developed in parallel to this work.

Our entity-driven proposal is strongly influenced by the ideas in the pipeline model of Chambers and Jurafsky (2011). Coreference chains have been used in a variety of learning tasks, such as narrative learning and summarization. Here we are the first to show how it can be used for schema induction in a probabilistic model, connecting predicates across a document in a way that is otherwise difficult to represent. The models perform similarly, but ours also includes significant benefits like a reduction in complexity, reproducibility, and a large reduction in training data requirements.

This paper also implies that learning and extraction need not be independent algorithms. Our model's inference procedure to learn schemas is the same one that labels text for extraction. C&J required 3-4 separate pipelined steps. Cheung et al. (2013) required specific cutoffs for document classification before extraction. Not only does our model perform well, but it does so without these steps.

Highlighted here are key differences between this proposal and the HMM-based model of Cheung et al. (2013). One of the HMM strengths is the inclusion of sequence-based knowledge. Each slot label is influenced by the previous label in the text, encouraging syntactic arguments of a predicate to choose the same schema. This knowledge is only loosely present in our document distribution θ. Cheung et al. also include a hidden event variable between the template and slot variables. Our model collapses this event variable and makes fewer dependency assumptions. This difference requires further investigation as it is unclear if it provides valuable information, or too much complexity.

We also note a warning for future work on proper evaluation methodology. This task is particularly difficult to compare to other models due to its combination of both induction and then extraction. There are many ways to map induced schemas to gold answers, and this paper illustrates how extraction performance is significantly affected by the choice. We suggest the template-mapping evaluation to preserve learned structure.

Finally, these induced results are far behind supervised learning (Huang and Riloff, 2011). There is ample room for improvement and future research in event schema induction.

Acknowledgments

This work was partially supported by a grant from the Office of Naval Research. It was also supported, in part, by the Johns Hopkins Human Language Technology Center of Excellence. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author. Thanks to Eric Wang for his insights into Bayesian modeling, Brendan O'Connor for his efforts on normalizing MUC-4 evaluation details, Frank Ferraro and Benjamin Van Durme for helpful conversations, and to the reviewers for insightful feedback.

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007a. Learning relations from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007b. Open information extraction from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

Razvan Bunescu and Raymond Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 438–445.

Andrew Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. 2010a. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Andrew Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr., and T.M. Mitchell. 2010b. Coupled semi-supervised learning for information extraction. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the Association for Computational Linguistics*.

Harr Chen, Edward Benson, Tahira Naseem, and Regina Barzilay. 2011. In-domain relation discovery with meta-constraints via posterior regularization. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. 2003. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Nancy Chinchor, David Lewis, and Lynette Hirschman. 1993. Evaluating message understanding systems: an analysis of the third message understanding conference. *Computational Linguistics*, 19:3:409–449.

Benjamin Van Durme and Marius Pasca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd Annual Conference on Artificial Intelligence (AAAI-2008)*, pages 1243–1248.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Dayne Freitag. 1998. Toward general-purpose learning for information extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 404–408.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 5228–5235.

Aria Haghighi and Dan Klein. 2010. An entity-level approach to information extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Ruihong Huang and Ellen Riloff. 2010. Inducing domain-specific semantic class taggers from (almost) nothing. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: Detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mstislav Maslennikov and Tat-Seng Chua. 2007. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Brendan O'Connor. 2012. Learning frames from text with an unsupervised latent variable model. Technical report, Carnegie Mellon University.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective ie with semantic affinity patterns and relevant regions. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Lisa Rau, George Krupka, Paul Jacobs, Ira Sider, and Lois Childs. 1992. Ge nltoolset: Muc-4 test results and analysis. In *Proceedings of the Message Understanding Conference (MUC-4)*, pages 94–99.

Ellen Riloff and Mark Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Ellen Riloff, Janyce Wiebe, and William Phillips. 2005. Exploiting subjectivity classification to improve information extraction. In *Proceedings of AAAI-05*.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the Joint Conference of the*

International Committee on Computational Linguistics and the Association for Computational Linguistics, pages 731–738.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive ie using unrestricted relation discovery. In *Proceedings of NAACL*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 224–231.

Beth M. Sundheim. 1991. Third message understanding evaluation and conference (muc-3): Phase 1 status report. In *Proceedings of the Message Understanding Conference*.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. A hybrid approach for the acquisition of information extraction patterns. In *Proceedings of the EACL Workshop on Adaptive Text Extraction and Mining*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 940–946.

Using Soft Constraints in Joint Inference for Clinical Concept Recognition

Prateek Jindal and **Dan Roth**
Department of Computer Science, UIUC
201 N. Goodwin Ave, Urbana, IL 61801, USA
{jindal2, danr}@illinois.edu

Abstract

This paper introduces IQPs (Integer Quadratic Programs) as a way to model joint inference for the task of concept recognition in clinical domain. IQPs make it possible to easily incorporate soft constraints in the optimization framework and still support exact global inference. We show that soft constraints give statistically significant performance improvements when compared to hard constraints.

1 Introduction

In this paper, we study the problem of concept recognition in the clinical domain. State-of-the-art approaches (de Bruijn et al., 2011; Patrick et al., 2011; Torii et al., 2011; Minard et al., 2011; Jiang et al., 2011; Xu et al., 2012; Roberts and Harabagiu, 2011; Jindal and Roth, 2013) for concept recognition in clinical domain (Uzuner et al., 2011) use sequence-prediction models like CRF (Lafferty et al., 2001), MEMM (McCallum et al., 2000) etc. These approaches are limited by the fact that they can model only local dependencies (most often, first-order models like linear chain CRFs are used to allow tractable inference).

Clinical narratives, unlike newswire data, provide a domain with significant knowledge that can be exploited systematically to improve the accuracy of the prediction task. Knowledge in this domain can be thought of as belonging to two categories: (1) *Background Knowledge* captured in medical ontologies like UMLS (Url1, 2013), MeSH and SNOMED CT and (2) *Discourse Knowledge* driven by the fact that the narratives adhere to a specific writing style. While the former can be used by generating more expressive knowledge-rich features, the latter is more interesting from our current perspective, since it provides global constraints on what *output* structures are likely and what are not. We exploit this structural knowledge in our global inference formulation.

Integer Linear Programming (ILP) based approaches have been used for global inference in many works (Roth and Yih, 2004; Punyakanok et al., 2004; Punyakanok et al., 2008; Marciniak and Strube, 2005; Bramsen et al., 2006; Barzilay and Lapata, 2006; Riedel and Clarke, 2006; Clarke and Lapata, 2007; Clarke and Lapata, 2008; Denis et al., 2007; Chang et al., 2011). However, in most of these works, researchers have focussed only on hard constraints while formulating the inference problem.

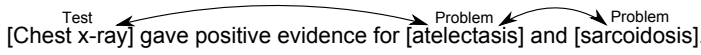
Formulating all the constraints as hard constraints is not always desirable because the constraints are not perfect in many cases. In this paper, we propose Integer Quadratic Programs (IQPs) as a way of formulating the inference problem. IQPs is a richer family of models than ILPs and it enables us to easily incorporate soft constraints into the inference procedure. Our experimental results show that soft constraints indeed give much better performance than hard constraints.

2 Identifying Medical Concepts

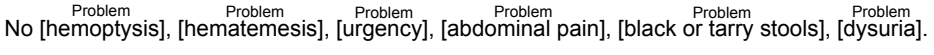
Task Description Our input consists of clinical reports in free-text (unstructured) format. The task is: (1) to identify the boundaries of medical concepts and (2) to assign types to such concepts. Each concept can have 3 possible types: (1) Test, (2) Treatment, and (3) Problem. We would refer to these three types by `TEST`, `TRE` and `PROB` in the following discussion.

Our Approach In the first step, we identify the concept boundaries using a CRF (with BIO encod-

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1808–1814,
Seattle, Washington, USA, 18-21 October 2013.



(a) Example 1



(b) Example 2

Figure 1: This figure motivates the global inference procedure we used. For discussion, please refer to §2.

ing). Features used by the CRF include the constituents given by MetaMap (Aronson and Lang, 2010; Url2, 2013), shallow parse constituents, surface form and part-of-speech (Url3, 2013) of words in a window of size 3. We also use conjunctions of the features.

After finding concept boundaries, we determine the probability distribution for each concept over 4 possible types (`TEST`, `TRE`, `PROB` or `NULL`). These probability distributions are found using a multi-class SVM classifier (Chang and Lin, 2011). Features used for training this classifier include concept tokens, full text of concept, bi-grams, headword, suffixes of headword, capitalization pattern, shallow parse constituent, Metamap type of concept, MetaMap type of headword, occurrence of concept in MeSH (Url4, 2013) and SNOMED CT (Url5, 2013), MeSH and SNOMED CT descriptors.

Inference Procedure: The final assignment of types to concepts is determined by an inference procedure. The basic principle behind our inference procedure is: "*Types of concepts which appear close to one another are often closely related. For some concepts, type can be determined with more confidence. And relations between concepts' types guide the inference procedure to determine the types of other concepts.*" We will now explain it in more detail with the help of examples. Figure 1 shows two sentences in which the concepts are shown in brackets and correct (gold) types of concepts are shown above them.

First, consider first and second concepts in Figure 1a. These concepts follow the pattern: *[Concept1] gave positive evidence for [Concept2].* In clinical narratives, such a pattern strongly suggests that *Concept1* is of type `TEST` and *Concept2* is of type `PROB`. Table 1 shows additional such patterns. Next, consider different concepts in Figure 1b. All these concepts are separated by commas and hence, form a list. It is highly likely that such concepts should have the same type.

	Pattern
1	using [TRE] for [PROB]
2	[TEST] showed [PROB]
3	Patient presents with [PROB] status post [TRE]
4	use [TRE] to correct [PROB]
5	[TEST] to rule out [PROB]
6	Unfortunately, [TRE] has caused [PROB]

Table 1: Some patterns that were used in constraints.

3 Modeling Global Inference

Inference is done at the level of sentences. Suppose there are m concepts in a sentence. Each of the m concepts has to be assigned one of the following types: `TEST`, `TRE`, `PROB` or `NULL`. To represent this as an inference problem, we define the indicator variables $x_{i,j}$ where i takes values from 1 to m (corresponding to concepts) and j takes values from 1 to 4 (corresponding to 4 possible types). $p_{i,j}$ refers to the probability that the i^{th} concept has type j.

We can now write the following optimization problem to find the optimal concept types:

$$\max_{x} \sum_{i=1}^{m} \sum_{j=1}^{4} x_{i,j} \cdot p_{i,j} \tag{1}$$

$$\text{subject to } \sum_{j=1}^{4} x_{i,j} = 1 \qquad \forall i \tag{2}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i, j \tag{3}$$

The objective function in Equation (1) expresses the fact that we want to maximize the expected number of correct predictions in each sentence. Equation (2) enforces the constraint that each concept has

a unique type. We would refer to these as **Type-1** constraints.

3.1 Constraints Used

In this subsection, we will describe two additional types of constraints (**Type-2** and **Type-3**) that were added to the optimization procedure described above. Whereas **Type-1** constraints described above were formulated as *hard constraints*, **Type-2** and **Type-3** constraints are formulated as *soft constraints*.

3.1.1 Type-2 Constraints

Certain constructs like comma, conjunction, etc. suggest that the 2 concepts appearing in them should have the same type. Figure 1b shows an example of such a constraint. Suppose that there are n_2 such constraints. Also, assume that the l^{th} constraint says that the concepts $\mathcal{R}_l$ and $\mathcal{S}_l$ should have the same type. To model this, we define a variable w_l as follows:

$$w_l = \sum_{m=1}^{4} (x_{\mathcal{R}_l,m} - x_{\mathcal{S}_l,m})^2 \tag{4}$$

Now, if the concepts $\mathcal{R}_l$ and $\mathcal{S}_l$ have the same type, then w_l would be equal to 0; otherwise, w_l would be equal to 2. So, the l^{th} constraint can be enforced by subtracting $(\rho_2 \cdot \frac{w_l}{2})$ from the objective function given by Equation (1). Thus, a penalty of ρ_2 would be enforced iff this constraint is violated.

3.1.2 Type-3 Constraints

Some short patterns suggest possible types for the concepts which appear in them. Each such pattern, thus, enforces a constraint on the types of corresponding concepts. Figure 1a shows an example of such a constraint. Suppose that there are n_3 such constraints. Also, assume that the k^{th} constraint says that the concept $\mathcal{A}_{1,k}$ should have the type $\mathcal{B}_{1,k}$ and that the concept $\mathcal{A}_{2,k}$ should have the type $\mathcal{B}_{2,k}$. Equivalently, the k^{th} constraint can be written as follows in boolean algebra notation: $(x_{\mathcal{A}_{1,k},\mathcal{B}_{1,k}} = 1) \wedge (x_{\mathcal{A}_{2,k},\mathcal{B}_{2,k}} = 1)$. For the k^{th} constraint, we introduce one more variable $z_k \in \{0, 1\}$ which satisfies the following condition:

$$z_k = 1 \Leftrightarrow x_{\mathcal{A}_{1,k},\mathcal{B}_{1,k}} \wedge x_{\mathcal{A}_{2,k},\mathcal{B}_{2,k}} \tag{5}$$

Using boolean algebra, it is easy to show that Equation (5) can be reduced to a set of linear inequalities. Thus, we can incorporate the k^{th} constraint in the optimization problem by adding to it the constraint given by Equation (5) and by subtracting $(\rho_3(1 - z_k))$ from the objective function given by Equation (1). Thus, a penalty of ρ_3 is imposed iff k^{th} constraint is not satisfied ($z_k = 0$).

$$\max_x \sum_{i=1}^{m}\sum_{j=1}^{4} x_{i,j} \cdot p_{i,j} - \sum_{k=1}^{n_3} \rho_3(1 - z_k) - \sum_{l=1}^{n_2} \left(\rho_2 \cdot \frac{\sum_{m=1}^{4} (x_{\mathcal{R}_l,m} - x_{\mathcal{S}_l,m})^2}{2} \right) \tag{6}$$

$$\text{subject to } \sum_{j=1}^{4} x_{i,j} = 1 \qquad \forall i \tag{7}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i, j \tag{8}$$

$$z_k = 1 \Leftrightarrow x_{\mathcal{A}_{1,k},\mathcal{B}_{1,k}} \wedge x_{\mathcal{A}_{2,k},\mathcal{B}_{2,k}} \forall k \in \{1...n_3\} \tag{9}$$

Figure 2: Final Optimization Problem (an IQP)

3.2 Final Optimization Problem - An IQP

After incorporating all the constraints mentioned above, the final optimization problem (an IQP) is shown in Figure 2. We used Gurobi toolkit (Url6, 2013) to solve such IQPs. In our case, it solves 76 IQPs per second on a quad-core server with Intel Xeon X5650 @ 2.67 GHz processors and 50 GB RAM.

4 Experiments and Results

4.1 Datasets and Evaluation Metrics

For our experiments, we used the datasets provided by i2b2/VA team as part of 2010 i2b2/VA shared task (Uzuner et al., 2011). The datasets used for this shared task contained de-identied clinical reports from three medical institutions: Partners Healthcare (`PH`), Beth-Israel Deaconess Medical Center (`BIDMC`) and the University of Pittsburgh Medical Center (UPMC). UPMC data was divided into 2 sections, namely discharge (`UPMCD`) and progress notes (`UPMCP`). A total of 349 training reports and 477 test reports were made available to the participants. However, data which came from UPMC (more than 50% data) was not made available for public use. As a result, we had only 170 clinical reports for training and 256 clinical reports for testing. Table 3 shows the number of clinical reports made available by different institutions. The

	B			BK			BC			BKC		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
TEST	92.4	79.4	85.4	91.9	80.2	85.7	92.7	79.6	85.7	92.1	80.4	**85.8**
TRE	92.1	73.6	81.8	92.0	79.5	85.3	92.3	76.8	83.8	92.0	80.2	**85.7**
PROB	83.6	83.6	83.6	88.9	83.7	86.3	85.9	83.8	84.8	89.6	83.9	**86.7**
OVERALL	88.4	79.4	83.6	90.7	81.4	85.8	89.6	80.5	84.8	91.0	81.7	**86.1**

Table 2: Our final system, **BKC**, consistently performed the best among all 4 systems (**B**, **BK**, **BC** and **BKC**).

	PH	BIDMC	~~UPMCD~~	~~UPMCP~~
Train	97	73	~~98~~	~~81~~
Test	133	123	~~102~~	~~119~~

Table 3: Dataset Characteristics

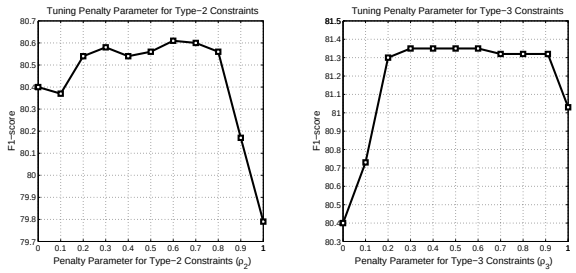
strikethrough text in this table indicates that the data was not made available for public use and hence, we couldn't use it. We used about 20% of the training data as a development set. For evaluation, we report precision, recall and F1 scores.

4.2 Results

In this section, we would refer to following 4 systems: (1) *Baseline* **(B)**, (2) *Baseline + Knowledge* **(BK)**, (3) *Baseline + Constraints* **(BC)** and (4) *Baseline + Knowledge + Constraints* **(BKC)**. Please note that the difference between **B** and **BK** is that **B** does not use the features derived from domain-specific knowledge sources (namely MetaMap, UMLS, MeSH and SNOMED CT) for training the classifiers. Both **B** and **BK** do not use the inference procedure. **BKC** uses all the features and also the inference procedure. In addition to these 4 systems, we would refer to another system, namely, **BKC-HARD**. This is similar to **BKC** system. However, it sets $\rho_2 = \rho_3 = 1$ which effectively turns **Type-2** and **Type-3** constraints into hard constraints by imposing very high penalty.

4.2.1 Importance of Soft Constraints

Figures 3a and 3b show the effect of varying the penalties (ρ_2 and ρ_3) for **Type-2** and **Type-3** constraints respectively. These figures show the F1-score of **BKC** on the development set. Penalty of 0 means that the constraint is not active. As we increase the penalty, the constraint becomes stronger. As the penalty becomes 1, the constraint becomes hard in the sense that final assignments must respect



(a) Type-2 Constraints **(b)** Type-3 Constraints

Figure 3: These figures show the result of tuning the penalty parameters (ρ_2 and ρ_3) for soft constraints.

	BKC-HARD	**BKC**
TEST	84.7	**85.8**
TRE	84.7	**85.7**
PROB	85.6	**86.7**
OVERALL	85.1	**86.1**

Table 4: Soft constraints **(BKC)** consistently perform much better than hard constraints **(BKC-HARD)**.

the constraint. We observe from Figures 3a and 3b that for **Type-2** and **Type-3** constraints, global maxima is attained at $\rho_2 = 0.6$ and $\rho_3 = 0.3$ respectively.

Hard vs Soft Constraints Table 4 compares the performance of **BKC-HARD** with that of **BKC**. First 3 rows in this table show the performance of both systems for the individual categories (TEST, TRE and PROB). The fourth row shows the overall score of both systems. **BKC** outperformed **BKC-HARD** on all the categories by statistically significant differences at $p = 0.05$ according to Bootstrap Resampling Test (Koehn, 2004). For the OVERALL category, **BKC** improved over **BKC-HARD** by 1.0 F1 points.

Figure 4: This figure shows the effect of training data size on performance of concept recognition.

4.2.2 Comparing with state-of-the-art baseline

In the 2010 i2b2/VA shared task, majority of top systems were CRF-based models, motivating the use of CRF as our baseline. Table 2 compares the performance of 4 systems: **B**, **BK**, **BC** and **BKC**. As pointed out before, our **BK** system uses CRF for boundary detection, employs all the knowledge-based features and is very similar to the top-performing systems in i2b2 challenge. We see from Table 2 that **BKC** consistently performed the best for individual as well as overall categories[1]. This result is statistically significant at $p = 0.05$ according to Bootstrap Resampling Test (Koehn, 2004). It should also be noted that **BC** performed significantly better than **B** for all the categories. Thus, the constraints are helpful even in the absence of knowledge-based features. Since we report results on publicly available datasets, future works would be able to compare their results with ours.

4.2.3 Effect of training data size

In Figure 4, we report the overall F1-score on a part of the development set as we vary the size of the training data from 40 documents to 130 documents. We notice that the performance increases steadily as more and more training data is provided. This suggests that if we could train on full training data as was made available in the challenge, the final scores could be much higher. We also notice from the figure that **BKC** consistently outperforms the state-of-the-art **BK** system as we vary the size of the training data, indicating the robustness of the joint inference procedure.

[1]Please note that the results reported in Table 2 can not be directly compared with those reported in the challenge because we only had a fraction of the original training and testing data.

5 Discussion and Related Work

In this paper, we chose to train a rather simple sequential model (using CRF), and focused on incorporating global constraints only at inference time[2]. While it is possible to jointly train the model with the global constraints (as illustrated by Chang et al. (2007), Mann and McCallum (2007), Mann and McCallum (2008), Ganchev et al. (2010) etc.), this process will be a lot less efficient, and prior work (Roth and Yih, 2005) has shown that it may not be beneficial.

Roth and Yih (2004, 2007) suggested the use of integer programs to model joint inference in a fully supervised setting. Our paper follows their conceptual approach. However, they used only hard constraints in their inference formulation. Chang et al. (2012) extended the ILP formulation and used soft constraints within the Constrained Conditional Model formulation (Chang, 2011). However, their implementation performed only approximate inference. In this paper, we extended the integer linear programming to a quadratic formulation, arguing that it simplifies the modeling step[3], and showed that it is possible to do exact inference efficiently.

Conclusion

This paper presented a global inference strategy (using IQP) for concept recognition which allows us to model structural knowledge of the clinical domain as soft constraints in the optimization framework. Our results showed that soft constraints are more effective than hard constraints.

Acknowledgments

This research was supported by Grant HHS 90TR0003/01 and by IARPA FUSE program via DoI/NBC contract #D11PC2015. Its contents are solely the responsibility of the authors and do not necessarily represent the official views, either expressed or implied, of the HHS, IARPA, DoI/NBC or the US government. The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

[2]In another experiment, we replaced the CRF with an MEMM. Surprisingly, MEMM performed as well as CRF.

[3]It should be noted that it is possible to reduce IQPs to ILPs using variable substitution. However, the resulting ILPs can be exponentially larger than original IQPs.

References

A.R. Aronson and F.M. Lang. 2010. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229.

R. Barzilay and M. Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366. Association for Computational Linguistics.

P. Bramsen, P. Deshpande, Y.K. Lee, and R. Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 189–198. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Association for Computational Linguistics*, pages 280–287, Prague, Czech Republic, 6. Association for Computational Linguistics.

K.-W. Chang, R. Samdani, A. Rozovskaya, N. Rizzolo, M. Sammons, and D. Roth. 2011. Inference protocols for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 40–44, Portland, Oregon, USA. Association for Computational Linguistics.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine learning*, pages 1–33.

M. Chang. 2011. *Structured Prediction with Indirect Supervision*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1–11.

J. Clarke and M. Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31(1):399–429.

B. de Bruijn, C. Cherry, S. Kiritchenko, J. Martin, and X. Zhu. 2011. Machine-learned solutions for three stages of clinical information extraction: the state of the art at i2b2 2010. *Journal of the American Medical Informatics Association*, 18(5):557–562.

P. Denis, J. Baldridge, et al. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 236–243.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

M. Jiang, Y. Chen, M. Liu, S.T. Rosenbloom, S. Mani, J.C. Denny, and H. Xu. 2011. A study of machine-learning-based approaches to extract clinical entities and their assertions from discharge summaries. *J Am Med Info Assoc*, 18(5):601–606.

P. Jindal and D. Roth. 2013. End-to-end coreference resolution for clinical narratives. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2106–2112, 8.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of Empirical Methods in Natural Language Processing*, volume 4, pages 388–395.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gideon S Mann and Andrew McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th international conference on Machine learning*, pages 593–600. ACM.

Gideon Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of Association for Computational Linguistics*, pages 870–878.

T. Marciniak and M. Strube. 2005. Beyond the pipeline: Discrete optimization in nlp. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 136–143. Association for Computational Linguistics.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598.

A.L. Minard, A.L. Ligozat, A.B. Abacha, D. Bernhard, B. Cartoni, L. Deléger, B. Grau, S. Rosset, P. Zweigenbaum, and C. Grouin. 2011. Hybrid methods for

improving information access in clinical documents: Concept, assertion, and relation identification. *J Am Med Info Assoc*, 18(5):588–593.

J.D. Patrick, D.H.M. Nguyen, Y. Wang, and M. Li. 2011. A knowledge discovery and reuse pipeline for information extraction in clinical notes. *Journal of the American Medical Informatics Association*, 18(5):574–579.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 129–137. Association for Computational Linguistics.

K. Roberts and S.M. Harabagiu. 2011. A flexible framework for deriving assertions from electronic medical records. *Journal of the American Medical Informatics Association*, 18(5):568–573.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 737–744.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to Statistical Relational Learning*, pages 553–580.

M. Torii, K. Wagholikar, and H. Liu. 2011. Using machine learning for concept extraction on clinical documents from multiple data sources. *Journal of the American Medical Informatics Association*, 18(5):580–587.

Url1. 2013. Umls: Unified medical language system (http://www.nlm.nih.gov/research/umls/) (accessed july 1, 2013).

Url2. 2013. Metamap (http://metamap.nlm.nih.gov/) (accessed july 1, 2013).

Url3. 2013. Illinois part-of-speech tagger (http://cogcomp.cs.illinois.edu/page/software_view/pos) (accessed july 1, 2013).

Url4. 2013. Mesh: Medical subject headings (http://www.nlm.nih.gov/mesh/meshhome.html) (accessed july 1, 2013).

Url5. 2013. Snomed ct: Snomed clinical terms (http://www.ihtsdo.org/snomed-ct/) (accessed july 1, 2013).

Url6. 2013. Gurobi optimization toolkit (http://www.gurobi.com/) (accessed july 1, 2013).

O. Uzuner, B.R. South, S. Shen, and S.L. DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of American Medical Informatics Association*.

Y. Xu, K. Hong, J. Tsujii, I. Eric, and C. Chang. 2012. Feature engineering combined with machine learning and rule-based methods for structured information extraction from narrative clinical discharge summaries. *Journal of the American Medical Informatics Association*, 19(5):824–832.

Exploring Demographic Language Variations to Improve Multilingual Sentiment Analysis in Social Media

Svitlana Volkova
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD
svitlana@jhu.edu

Theresa Wilson
Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD
taw@jhu.edu

David Yarowsky
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD
yarowsky@cs.jhu.edu

Abstract

Different demographics, e.g., gender or age, can demonstrate substantial variation in their language use, particularly in informal contexts such as social media. In this paper we focus on learning gender differences in the use of subjective language in English, Spanish, and Russian Twitter data, and explore cross-cultural differences in emoticon and hashtag use for male and female users. We show that gender differences in subjective language can effectively be used to improve sentiment analysis, and in particular, polarity classification for Spanish and Russian. Our results show statistically significant relative F-measure improvement over the gender-independent baseline 1.5% and 1% for Russian, 2% and 0.5% for Spanish, and 2.5% and 5% for English for polarity and subjectivity classification.

1 Introduction

Sociolinguistics and dialectology have been studying the relationships between language and speech at the phonological, lexical and morphosyntactic levels and social identity for decades (Picard, 1997; Gefen and Ridings, 2005; Holmes and Meyerhoff, 2004; Macaulay, 2006; Tagliamonte, 2006). Recent studies have focused on exploring demographic language variations in personal email communication, blog posts, and public discussions (Boneva et al., 2001; Mohammad and Yang, 2011; Eisenstein et al., 2010; O'Connor et al., 2010; Bamman et al., 2012). However, one area that remains largely unexplored is the effect of demographic language variation on subjective language use, and whether these differences may be exploited for automatic sentiment analysis. With the growing commercial importance of applications such as personalized recommender systems and targeted advertising (Fan and Chang, 2009), detecting helpful product review (Ott et al., 2011), tracking sentiment in real time (Resnik, 2013), and large-scale, low-cost, passive polling (O'Connor et al., 2010), we believe that sentiment analysis guided by user demographics is a very important direction for research.

In this paper, we focus on gender demographics and language in social media to investigate differences in the language used to express opinions in Twitter for three languages: English, Spanish, and Russian. We focus on Twitter data because of its volume, dynamic nature, and diverse population worldwide.[1] We find that some words are more or less likely to be positive or negative in context depending on the the gender of the author. For example, the word *weakness* is more likely to be used in a positive way by women (*Chocolate is my weakness!*) but in a negative way by men (*Clearly they know our weakness. Arggggg*). The Russian word достичь (achieve) is used in a positive way by male users and in a negative way by female users.

Our goals of this work are to (1) explore the gender bias in the use of subjective language in social media, and (2) incorporate this bias into models to improve sentiment analysis for English, Spanish, and Russian. Specifically, in this paper we:

- investigate multilingual lexical variations in the use of subjective language, and cross-cultural

[1] As of May 2013, Twitter has 500m users (140m of them in the US) from more than 100 countries.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1815–1827,
Seattle, Washington, USA, 18-21 October 2013.

emoticon and hashtag usage on a large scale in Twitter data;[2]

- show that gender bias in the use of subjective language can be used to improve sentiment analysis for multiple languages in Twitter.
- demonstrate that simple, binary features representing author gender are insufficient; rather, it is the combination of lexical features, together with set-count features representing gender-dependent sentiment terms that is needed for statistically significant improvements.

To the best of our knowledge, this work is the first to show that incorporating gender leads to significant improvements for sentiment analysis, particularly subjectivity and polarity classification, for multiple languages in social media.

2 Related Work

Numerous studies since the early 1970's have investigated gender-language differences in interaction, theme, and grammar among other topics (Schiffman, 2002; Sunderland et al., 2002). More recent research has studied gender differences in telephone speech (Cieri et al., 2004; Godfrey et al., 1992) and emails (Styler, 2011). Mohammad and Yang (2011) analyzed gender differences in the expression of sentiment in love letters, hate mail, and suicide notes, and emotional word usage across genders in email.

There has also been a considerable amount of work in subjectivity and sentiment analysis over the past decade, including, more recently, in microblogs (Barbosa and Feng, 2010; Bermingham and Smeaton, 2010; Pak and Paroubek, 2010; Bifet and Frank, 2010; Davidov et al., 2010; Li et al., 2010; Kouloumpis et al., 2011; Jiang et al., 2011; Agarwal et al., 2011; Wang et al., 2011; Calais Guerra et al., 2011; Tan et al., 2011; Chen et al., 2012; Li et al., 2012). In spite of the surge of research in both sentiment and social media, only a limited amount of work focusing on gender identification has looked at differences in subjective language across genders within social media. Thelwall (2010) found that men and women use emoticons to differing degrees on MySpace, e.g., female users express positive emoticons more than male users. Other researchers included subjective patterns as features for gender classification of Twitter users (Rao et al., 2010). They found that the majority of emotion-bearing features, e.g., emoticons, repeated letters, exasperation, are used more by female than male users, which is consistent with results reported in other recent work (Garera and Yarowsky, 2009; Burger et al., 2011; Goswami et al., 2009; Argamon et al., 2007). Other related work is that of Otterbacher (2010), who studied stylistic differences between male and female reviewers writing product reviews, and Mukherjee and Liu (2010), who applied positive, negative and emotional connotation features for gender classification in microblogs.

Although previous work has investigated gender differences in the use of subjective language, and features of sentiment have been used in gender identification, to the best of our knowledge no one has yet investigated whether gender differences in the use of subjective language can be exploited to improve sentiment classification in English or any other language. In this paper we seek to answer this question for the domain of social media.

3 Data

For the experiments in this paper, we use three sets of data for each language: a large pool of data (800K tweets) labeled for gender but *unlabeled* for sentiment, plus 2K development data and 2K test data labeled for both sentiment and gender. We use the unlabeled data to bootstrap Twitter-specific lexicons and investigate gender differences in the use of subjective language. We use the development data for parameter tuning while bootstrapping, and the test data for sentiment classification.

For English, we download tweets from the corpus created by Burger et al. (2011). This dataset contains 2,958,103 tweets from 184K users, excluding retweets. Retweets are omitted because our focus is on the sentiment of the person tweeting; in retweets, the words originate from a different user. All users in this corpus have gender labels, which Burger et al. automatically extracted from self-reported gender on Facebook or MySpace profiles linked to by the Twitter users. English tweets are identified using a compression-based language identification (LID)

[2]Gender-dependent and independent lexical resources of subjective terms in Twitter for Russian, Spanish and English can be found here: `http://www.cs.jhu.edu/~svitlana/`

tool (Bergsma et al., 2012). According to LID, there are 1,881,620 (63.6%) English tweets from which we select a random, gender-balanced sample of 0.8M tweets. Burger's corpus does not include Russian and Spanish data on the same scale as English. Therefore, for Russian and Spanish we construct a new Twitter corpus by downloading tweets from followers of region-specific news and media Twitter feeds. We use LID to identify Russian and Spanish tweets, and remove retweets as before. In this data, gender is labeled automatically based on user first and last name morphology with a precision above 0.98 for all languages.

Sentiment labels for tweets in the development and test sets are obtained using Amazon Mechanical Turk. For each tweet we collect annotations from five workers and use majority vote to determine the final label for the tweet. Snow et al. (2008) show that for a similar task, labeling emotion and valence, on average four non-expert labelers are needed to achieve an expert level of annotation. Below are the example Russian tweets labeled for sentiment:

- **Pos**: Как же приятно просто лечь в постель после тяжелого дня... (It is a great pleasure to go to bed after a long day at work...)
- **Neg**: Уважаемый господин Прохоров купите эти выборы! (Dear Mr. Prokhorov just buy the elections!)
- **Both**: Затолкали меня на местном рынке! но зато закупилась подарками для всей семьи :) (It was crowded at the local market! But I got presents for my family:-))
- **Neutral**: Киев очень старый город (Kiev is a very old city).

Table 1 gives the distribution of tweets over sentiment and gender labels for the development and test sets for English (EDEV, ETEST), Spanish (SDEV, STEST), and Russian (RDEV, RTEST).

Data	Pos	Neg	Both	Neut	♀	♂
EDEV	617	357	202	824	1,176	824
ETEST	596	347	195	862	1,194	806
SDEV	358	354	86	1,202	768	1,232
STEST	317	387	93	1203	700	1,300
RDEV	452	463	156	929	1,016	984
RTEST	488	380	149	983	910	1,090

Table 1: Gender and sentiment label distribution in the development and test sets for all languages.

4 Subjective Language and Gender

To study the intersection of subjective language and gender in social media, ideally we would have a large corpus labeled for both. Although our large corpus is labeled for gender, it is not labeled for sentiment. Only the 4K tweets for each language that compose the development and test sets are labeled for both gender and sentiment. Obtaining sentiment labels for all tweets would be both impractical and expensive. Instead we use large multilingual sentiment lexicons developed specifically for Twitter as described below. Using these lexicons we can begin to explore the relationship between subjective language and gender in the large pool of data labeled for gender but unlabeled for sentiment. We also look at the relationship between gender and the use of different hashtags and emoticons. These can be strong indicators of sentiment in social media, and in fact are sometimes used to create noisy training data for sentiment analysis in Twitter (Pak and Paroubek, 2010; Kouloumpis et al., 2011).

4.1 Bootstrapping Subjectivity Lexicons

Recent work by Banea et.al (2012) classifies methods for bootstrapping subjectivity lexicons into two types: corpus-based and dictionary-based. Corpus-based methods extract subjectivity lexicons from unlabeled data using different similarity metrics to measure the relatedness between words, e.g., Pointwise Mutual Information (PMI). Corpus-based methods have been used to bootstrap lexicons for ENGLISH (Turney, 2002) and other languages, including ROMANIAN (Banea et al., 2008) and JAPANESE (Kaji and Kitsuregawa, 2007).

Dictionary-based methods rely on relations between words in existing lexical resources. For example, Rao and Ravichandran (2009) construct HINDI and FRENCH sentiment lexicons using relations in WordNet (Miller, 1995), Rosas et. al. (2012) bootstrap a SPANISH lexicon using SentiWordNet (Baccianella et al., 2010) and OpinionFinder,[3] Clematide and Klenner (2010), Chetviorkin et al. (2012) and Abdul-Mageed et. al. (2011) automatically expand and evaluate GERMAN, RUSSIAN and ARABIC subjective lexicons.

[3] www.cs.pitt.edu/mpqa/opinionfinder

We use the corpus-based, language-independent approach proposed by Volkova et al. (2013) to bootstrap Twitter-specific subjectivity lexicons. To start, the new lexicon is seeded with terms from the initial lexicon L_I. On each iteration, tweets in the unlabeled data are labeled using the current lexicon. If a tweet contains one or more terms from the lexicon it is marked subjective, otherwise neutral. Tweet polarity is determined in a similar way, but takes into account negation. For every term not in the lexicon with a frequency threshold, the probability of that word appearing in a subjective sentence is calculated. The top k terms with a subjective probability are then added to the lexicon. Bootstrapping continues until there are no more new terms meeting the criteria to add to the lexicon. The parameters are optimized using a grid search on the development data using F-measure for subjectivity classification. In Table 2 we report size and term polarity from the initial L_I and the bootstrapped L_B lexicons. Although more sophisticated bootstrapping methods exist, this approach has been shown to be effective for atomically learning subjectivity lexicons in multiple languages on a large scale without any external, rich, lexical resources, e.g., WordNet, or advanced NLP tools, e.g., syntactic parsers (Wiebe, 2000) or information extraction tools (Riloff and Wiebe, 2003).

For English, seed terms for bootstrapping are the strongly subjective terms in the MPQA lexicon (Wilson et al., 2005). For Spanish and Russian, the seed terms are obtained by translating the English seed terms using a bi-lingual dictionary, collecting subjectivity judgments from MTurk on the translations, filtering out translations that are not strongly subjective, and expanding the resulting word lists with plurals and inflectional forms.

To verify that bootstrapping does provide a better resource than existing dictionary-expanded lexicons, we compare our Twitter-specific lexicons L_B to the corresponding initial lexicons L_I and the existing state-of-the-art subjective lexicons including:

- 8K strongly subjective English terms from SentiWordNet χ^E (Baccianella et al., 2010);
- 1.5K full strength terms from the Spanish sentiment lexicon χ^S (Perez-Rosas et al., 2012);
- 5K terms from the Russian sentiment lexicon χ^R (Chetviorkin and Loukachevitch, 2012).

For that we apply rule-based subjectivity classification on the test data.[4] This subjectivity classifier predicts that a tweet is subjective if it contains at least one, or at least two subjective terms from the lexicon. To make a fair comparison, we automatically expand χ^E with plurals and inflectional forms, χ^S with the inflectional forms for verbs, and χ^R with the inflectional forms for adverbs, adjectives and verbs. We report precision, recall and F-measure results in Table 3 and show that our bootstrapped lexicons outperform the corresponding initial lexicons and the external resources.

	English		**Spanish**		**Russian**	
	L_I^E	L_B^E	L_I^S	L_B^S	L_I^R	L_B^R
Pos	2.3	16.8	2.9	7.7	1.4	5.3
Neg	2.8	4.7	5.2	14.6	2.3	5.5
Total	**5.1**	**21.5**	**8.1**	**22.3**	**3.7**	**10.8**

Table 2: The initial L_I and the bootstrapped L_B (highlighted) lexicon term count ($L_I \subset L_B$) with polarity across languages (thousands).

	$Subj \geq 1$			$Subj \geq 2$		
	P	R	F	P	R	F
χ^E	0.67	0.49	0.57	0.76	0.16	0.27
L_I^E	**0.69**	0.73	0.71	**0.79**	0.34	0.48
L_B^E	0.64	**0.91**	**0.75**	0.7	**0.74**	**0.72**
χ^S	**0.52**	0.39	0.45	**0.62**	0.07	0.13
L_I^S	0.50	0.73	**0.59**	0.59	0.36	0.45
L_B^S	0.44	**0.91**	**0.59**	0.51	**0.71**	**0.59**
χ^R	0.61	0.49	0.55	0.74	0.17	0.29
L_I^R	**0.72**	0.34	0.46	**0.83**	0.07	0.13
L_B^R	0.64	**0.58**	**0.61**	0.74	**0.23**	**0.35**

Table 3: Precision, recall and F-measure results for subjectivity classification using the external χ, initial L_I and bootstrapped L_B lexicons for all languages.

4.2 Lexical Evaluation

With our Twitter-specific sentiment lexicons, we can now investigate how the *subjective use* of these terms differs depending on gender for our three languages. Figure 1 illustrates what we expect to find. $\{F\}$ and $\{M\}$ are the sets of subjective terms used by females and males, respectively. We expect that some terms will be used by males, but never by females, and vice-versa. The vast majority, however, will be used by both genders. Within this set of shared terms, many words will show little difference

[4] A similar rule-based approach using terms from the MPQA lexicon is suggested by (Riloff and Wiebe, 2003).

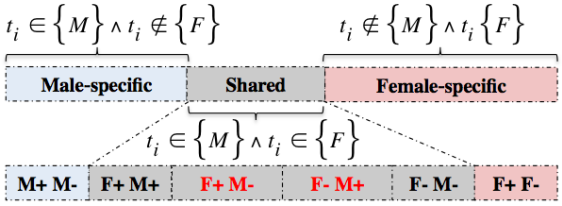


Figure 1: Gender-dependent vs. independent subjectivity terms (+ and - indicates term polarity).

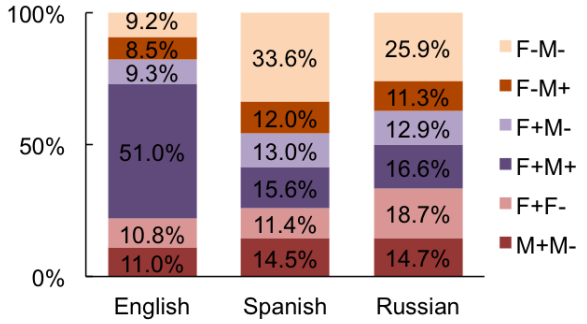


Figure 2: The distribution of gender-dependent $GDep$ and gender-independent $GInd$ sentiment terms.

in their subjective use when considering gender, but there will be some words for which gender will have an influence. Of particular interest for our work are words in which the polarity of a term as it is used in context is gender-influenced, the extreme case being terms that flip their polarity depending on the gender of the user. Polarity may be different because the concept represented by the term tends to be viewed in a different light depending on gender. There are also words like *weakness* in which a more positive or more negative word sense tends to be used by men or women. In Figure 2 we show the distribution of gender-specific and gender-independent terms from the L_B lexicons for all languages.

To identify gender-influenced terms in our lexicons, we start by randomly sampling 400K male and 400K female tweets for each language from the data. Next, for both genders we calculate the probability of term t_i appearing in a tweet with another subjective term (Eq.1), and the probability of it appearing with a positive or negative term (Eq.2-3) from L_B.

$$p_{t_i}(subj|g) = \frac{c(t_i, P, g) + c(t_i, N, g)}{c(t_i, g)}, \quad (1)$$

where $g \in F, M$ and P and N are positive and negative sets of terms from the *initial* lexicon L_I.

$$p_{t_i}(+|g) = \frac{c(t_i, P, g)}{c(t_i, P, g) + c(t_i, N, g)} \quad (2)$$

$$p_{t_i}(-|g) = \frac{c(t_i, N, g)}{c(t_i, P, g) + c(t_i, N, g)} \quad (3)$$

We introduce a novel metric $\Delta p^+_{t_i}$ to measure polarity change across genders. For every subjective term t_i we want to maximize the difference[5]:

$$\Delta p^+_{t_i} = |p_{t_i}(+|F) - p_{t_i}(+|M)|$$
$$s.t.$$
$$\left|1 - \frac{tf^{subj}_{t_i}(F)}{tf^{subj}_{t_i}(M)}\right| \leq \lambda, \; tf^{subj}_{t_i}(M) \neq 0, \quad (4)$$

where $p(+|F)$ and $p(+|M)$ are probabilities that term t_i is positive for females and males respectively; $tf^{subj}_{t_i}(F)$ and $tf^{subj}_{t_i}(M)$ are corresponding term frequencies (if $tf^{subj}_{t_i}(F) > tf^{subj}_{t_i}(M)$ the fraction is flipped); λ is a threshold that controls the level of term frequency similarity[6]. The terms in which polarity is most strongly gender-influenced are those with $\lambda \to 0$ and $\Delta p^+_{t_i} \to 1$.

Table 4 shows a sample of the most strongly gender-influenced terms from the initial L_I and the bootstrapped L_B lexicons for all languages. A plus (+) means that the term tends to be used positively by women and minus (−) means that the term tends to be used positively by men. For instance, in English we found that *perfecting* is used with negative polarity by male users but with positive polarity by female users; the term *dogfighting* has negative polarity for women but positive polarity for men.

4.3 Hashtags

People may also express positive or negative sentiment in their tweets using hashtags. From our balanced samples of 800K tweets for each language, we extracted 611, 879, and 71 unique hashtags for English, Spanish, and Russian, respectively. As we did for terms in the previous section, we evaluated the subjective use of the hashtags. Some of these are clearly expressing sentiment (*#horror*), while others seem to be topics that people are frequently opinionated about (*#baseball, #latingrammy, #spartak*).

[5] One can also maximize $\Delta p^-_{t_i} = |p_{t_i}(-|F) - p_{t_i}(-|M)|$.

[6] $\lambda = 0$ means term frequencies are identical for both genders; $\lambda \to 1$ indicates increasing gender divergence.

English Initial Terms L_I^E	Δp^+	λ	**English Bootstrapped Terms** L_B^E	Δp^+	λ
perfecting	+ 0.7	0.2	pleaseeeeee	+ 0.7	0.0
weakened	+ 0.1	0.0	adorably	+ 0.6	0.4
saddened	– 0.1	0.0	creatively	– 0.6	0.5
misbehaving	– 0.4	0.0	dogfighting	– 0.7	0.5
glorifying	– 0.7	0.5	overdressed	– 1.0	0.3
Spanish Initial Terms L_I^S			**Spanish Bootstrapped Terms** L_B^S		
fiasco (fiasco)	+ 0.7	0.3	cafeína (caffeine)	+ 0.7	0.5
triunfar (succeed)	+ 0.7	0.0	claro (clear)	+ 0.7	0.3
inconsciente (unconscious)	– 0.6	0.2	cancio (dog)	– 0.3	0.3
horroriza (horrifies)	– 0.7	0.3	llevara (take)	– 0.8	0.3
groseramente (rudely)	– 0.7	0.3	recomendarlo (recommend)	– 1.0	0.0
Russian Initial Terms L_I^R			**Russian Bootstrapped Terms** L_B^R		
магическая (magical)	+ 0.7	0.3	мечтайте (dream!)	+ 0.7	0.3
сенсационный (sensational)	+ 0.7	0.3	танцуете (dancing)	+ 0.7	0.3
обожаемый (adorable)	– 0.7	0.0	сложны (complicated)	– 1.0	0.0
искушение (temptation)	– 0.7	0.3	молоденькие (young)	– 1.0	0.0
заслуживать (deserve)	– 1.0	0.0	достичь (achieve)	– 1.0	0.0

Table 4: Sample of subjective terms sorted by Δp^+ to show lexical differences and polarity change across genders (module is not applied as defined in Eq.1 to demonstrate the polarity change direction).

English	Δp^+	λ	**Spanish**	Δp^+	λ	**Russian**	Δp^+	λ
#parenting	+ 0.7	0.0	#rafaelnarro (politician)	+ 1.0	0.0	#совет (advise)	+ 1.0	0.0
#vegas	– 0.2	0.8	#amores (loves)	+ 0.2	1.0	#ukrlaw	+ 1.0	1.0
#horror	– 0.6	0.7	#britneyspears	+ 0.1	0.3	#spartak (soccer team)	– 0.7	0.9
#baseball	– 0.6	0.9	#latingrammy	– 0.5	0.1	#сны (dreams)	– 1.0	0.0
#wolframalpha	– 0.7	1.0	#metallica (music band)	– 0.5	0.8	#iphones	– 1.0	1.0

Table 5: Hashtag examples with opposite polarity across genders for English, Spanish, and Russian.

Table 5 gives the hashtags, correlated with subjective language, that are most strongly gender-influenced. Analogously to Δp^+ values in Table 4, a plus (+) means the hashtag is more likely to be used positively by women, and a minus (–) means the hashtag is more likely to be used positively by men. For example, in English we found that male users tend to express positive sentiment in tweets mentioning $\#baseball$, while women tend to be negative about this hashtag. The opposite is true for the hashtag $\#parenting$.

4.4 Emoticons

We investigate how emoticons are used differently by men and women in social media following the work by (Bamman et al., 2012). For that we rely on the lists of emoticons from Wikipedia[7] and present the cross-cultural and gender emoticon differences in Figure 3. The frequency of each emoticon is given on the right of each language chart, with probability of use by a male user in that language given on the x-axis. The top 8 emoticons are the same across languages and sorted by English frequency.

We found that emoticons in English data are used more overall by female users, which is consistent with previous findings in Schnoebelen's work.[8] In addition, we found that some emoticons like :-) *(smile face)* and :-o *(surprised)* are used equally by both genders, at least in Twitter. When comparing English emoticon usage to other languages, there are some similarities, but also some clear differences. In Spanish data, several emoticons are more likely to be used by male than by female users, e.g., :-o *(surprised)* and :-& *(tongue-tied)*, and the difference in probability of use by males and females is greater for the emoticons, as compared to the same emoticons for English. Interestingly, in Russian Twitter

[7]List of emoticons from Wikipedia http://en.wikipedia.org/wiki/List_of_emoticons

[8]Language and emotion (talks, essays and reading notes) www.stanford.edu/~tylers/emotions.shtml

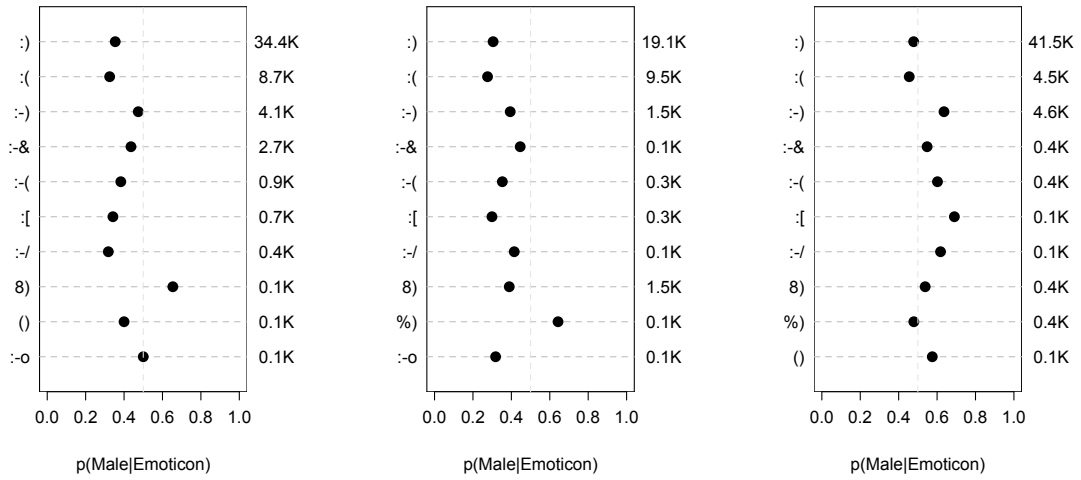


Figure 3: Probability of gender and emoticons for English, Spanish and Russian (from left to right).

data emoticons tend to be used more or equally by male users rather than female users.

5 Experiments

The previous section showed that there are gender differences in the use of subjective language, hashtags, and emoticons in Twitter. We aim leverage these differences to improve subjectivity and polarity classification for the informal, creative and dynamically changing multilingual Twitter data.[9] For that we conduct experiments using gender-independent $GInd$ and gender-dependent $GDep$ features and compare the results to evaluate the influence of gender on sentiment classification.

We experiment with two classification approaches: (I) rule-based classifier which uses only subjective terms from the lexicons designed to verify if the gender differences in subjective language create enough of a signal to influence sentiment classification; (II) state-of-the-art supervised models which rely on lexical features as well as lexicon set-count features.[10,11] Moreover, to show that the gender-sentiment signal can be learned by more than one classifier we apply a variety of classifiers implemented in Weka (Hall et al., 2009). For that we do 10-fold cross validation over English, Spanish, and Russian test data (ETEST, STEST and RTEST) labeled with subjectivity (pos, neg, both vs. neut) and polarity (pos vs. neg) as described in Section 3.

5.1 Models

For the rule-based $GInd^{RB}_{subj}$ classifier, tweets are labeled as subjective or neutral as follows:

$$GInd^{RB}_{subj} = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{f} \geq 0.5, \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $\vec{w} \cdot \vec{f}$ stands for weighted set features, e.g., terms from L_I only, emoticons E, or different part-of-speech tags (POS) from L_B weighted using $w = p(subj) = p(subj|M) + p(subj|F)$ subjectivity score as shown in Eq.1. We experiment with the POS tags to show the contribution of each POS to sentiment classification.

Similarly, for the rule-based $GInd^{RB}_{pol}$ classifier, tweets are labeled as positive or negative:

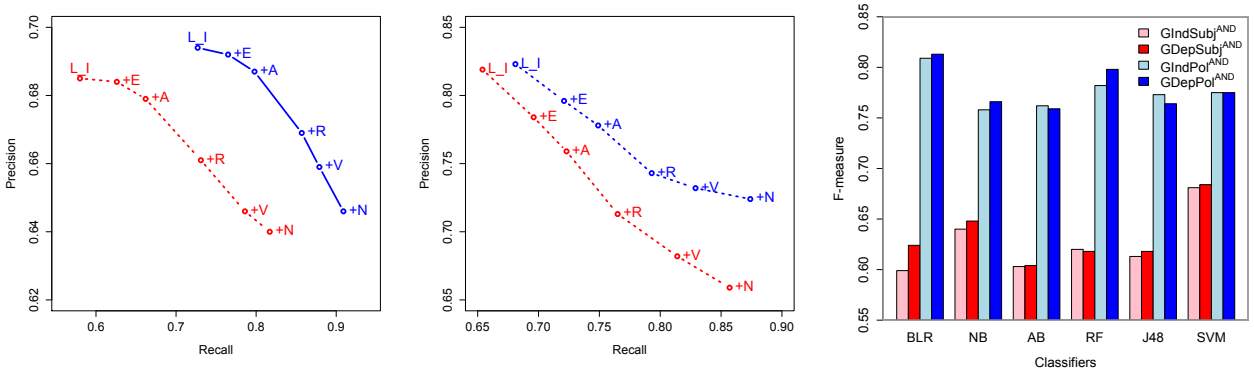
$$GInd^{RB}_{pol} = \begin{cases} 1 & \text{if } \vec{w^+} \cdot \vec{f^+} \geq \vec{w^-} \cdot \vec{f^-}, \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\vec{f}^+$, $\vec{f}^-$ are feature sets that include only positive and negative features from L_I or L_B; w^+ and w^-

[9] For polarity classification we distinguish between positive and negative instances, which is the approach typically reported in the literature for recognizing polarity (Velikovich et al., 2010; Yessenalina and Cardie, 2011; Taboada et al., 2011)

[10] A set-count feature is a count of the number of instances from a set of terms that appears in a tweet.

[11] We also experimented with repeated punctuation (*!!, ??*) and letters ($nooo, reealy$), which are often used in sentiment classification in social media. However, we found these features to be noisy and adding them decreased performance.



(a) Rule-based subjectivity (b) Rule-based polarity (c) SL subjectivity and polarity

Figure 4: Rule-based (RB) and Supervised Learning (SL) sentiment classification results for English. L_I - the initial lexicon, E - emoticons, A, R, V, N are adjectives, adverbs, verbs, nouns from L_B.

are positive and negative polarity scores estimated using Eq.2 - 3 such as: $w^+ = p(+|M) + p(+|F)$ and $w^- = p(-|M) + p(-|F)$.

The *gender-dependent* rule-based classifiers are defined in a similar way. Specifically, $\vec{f}$ is replaced by $\vec{f}^M$ and $\vec{f}^F$ in Eq.5 and $\vec{f}^-, \vec{f}^+$ are replaced by $\vec{f}^{M-}, \vec{f}^{F-}$ and $\vec{f}^{M+}, \vec{f}^{F+}$ respectively in Eq.6. We learn subjectivity $\vec{s}$ and polarity $\vec{p}$ score vectors using Eq.1-3. The difference between $GInd$ and $GDep$ models is that $GInd$ scores $\vec{w}$, $\vec{w^+}$ and $\vec{w^-}$ are not conditioned on gender.

For *gender-independent* classification using supervised models, we build feature vectors using lexical features V represented as term frequencies, together with set-count features from the lexicons:

$$\vec{f}^{GInd}_{subj} = [L_I, L_B, E, V];$$
$$\vec{f}^{GInd}_{pol} = [L^+_I, L^+_B, E^+, L^-_I, L^-_B, E^-, V].$$

Finally, for *gender-dependent* supervised models, we try different feature combinations. (A) We extract set-count features for gender-dependent subjective terms from L_I, L_B, and E jointly:

$$\vec{f}^{GDep-J}_{subj} = [L^M_I, L^M_B, E^M, L^F_I, L^F_B, E^F, V];$$
$$\vec{f}^{Dep-J}_{pol} = [L^{M+}_I, L^{M+}_B, E^{M+}, L^{F+}_I, L^{F+}_B, E^{F+}$$
$$L^{M-}_I, L^{M-}_B, E^{M-}, L^{F-}_I, L^{F-}_B, E^{F-}, V].$$

(B) We extract disjoint (prefixed) gender-specific features (in addition to lexical features V) by relying only on female set-count features when classifying female tweets; and only male set-count features for male tweets. We refer to the joint features as $GInd-J$ and $GDep-J$, and to the disjoint features $GInd-D$ and $GDep-D$.

5.2 Results

Figures 4a and 4b show performance improvements for subjectivity and polarity classification under the rule-based approach when taking into account gender. The left figure shows precision-recall curves for subjective *vs.* neutral classification, and the middle figure shows precision-recall curves for positive *vs.* negative classification. We measure performance starting with features from L_I, and then incrementally add emoticon features E and features from L_B one part of speech at a time to show the contribution of each part of speech for sentiment classification.[12] This experiment shows that there is a clear improvement for the models parameterized with gender, at least for the simple, rule-based model.

For the supervised models we experiment with a variety of learners for English to show that gender differences in subjective language improve sentiment classification for many learning algorithms. We present the results in Figure 4c. For subjectivity classification, Support Vector Machines (SVM), Naive Bayes (NB) and Bayesian Logistic Regression (BLR) achieve the best results, with improvements in F-measure ranging from 0.5 - 5%. The polarity classifiers overall achieve much higher scores, with improvements for $GDep$ features ranging from 1-2%. BLR with Gaussian prior is the top scorer

[12] POS from the Twitter POSTagger (Gimpel et al., 2011).

	P	R	F	A	A^{rand}	P	R	F	A	A^{rand}
	English subj vs. neutral p(subj)=0.57					**English pos vs. neg p(pos)=0.63**				
$GInd_{LR}$	0.62	0.58	0.60	0.66	–	0.78	0.83	0.80	0.71	–
$GDep-J$	0.64	0.62	0.63	0.68	*0.66*	0.80	0.83	0.82	0.73	*0.70*
$\Delta R,\%$	+3.23	+6.90	+5.00	+3.03	*3.03*↓	+2.56	0.00	+2.50	+2.82	*4.29*↓
$GInd_{SVM}$	0.66	0.70	0.68	0.72	–	0.79	0.86	0.82	0.77	–
$GDep-D$	0.66	0.71	0.68	0.72	*0.70*	0.80	0.87	0.83	0.78	*0.76*
$\Delta R,\%$	–0.45	+0.71	0.00	–0.14	*2.85*↓	+0.38	+0.23	+0.24	+0.41	*2.63*↓
	Spanish subj vs. neutral p(subj)=0.40					**Spanish pos vs. neg p(pos)=0.45**				
$GInd_{LL}$	0.67	0.71	0.68	0.61	–	0.71	0.63	0.67	0.71	–
$GDep-J$	0.67	0.72	0.69	0.62	*0.61*	0.72	0.65	0.68	0.71	*0.68*
$\Delta R,\%$	0.00	+1.40	+0.58	+0.73	*1.64*↓	+2.53	+3.17	+1.49	0.00	*4.41*↓
$GInd_{SVM}$	0.68	0.79	0.73	0.65	–	0.66	0.65	0.65	0.69	–
$GDep-D$	0.68	0.79	0.73	0.66	*0.65*	0.68	0.67	0.67	0.71	*0.68*
$\Delta R,\%$	+0.35	+0.21	+0.26	+0.54	*1.54*↓	+2.43	+2.44	+2.51	+2.08	*4.41*↓
	Russian subj vs. neutral p(subj)=0.51					**Russian pos vs. neg p(pos)=0.58**				
$GInd_{LR}$	0.66	0.68	0.67	0.67	–	0.66	0.72	0.69	0.62	–
$GDep-J$	0.66	0.69	0.68	0.67	*0.66*	0.68	0.73	0.70	0.64	*0.63*
$\Delta R,\%$	0.00	+1.47	+0.75	0.00	*1.51*↓	+3.03	+1.39	+1.45	+3.23	*1.58*↓
$GInd_{SVM}$	0.67	0.75	0.71	0.70	–	0.64	0.73	0.68	0.62	–
$GDep-D$	0.67	0.76	0.71	0.70	*0.69*	0.65	0.74	0.69	0.63	*0.62*
$\Delta R,\%$	–0.30	+1.46	+0.56	+0.14	*1.44*↓	+0.93	+1.92	+1.46	+1.49	*1.61*↓

Table 6: Sentiment classification results obtained using gender-dependent and gender-independent joint and disjoint features for Logistic Regression (LR) and SVM models.

for polarity classification with an F-measure of 82%. We test our results for statistical significance using McNemar's Chi-squared test (p-value < 0.01) as suggested by Dietterich (1998). Only three classifiers, J48, AdaBoostM1 (AB) and Random Forest (RF) do not always show significant improvements for $GDep$ features over $GInd$ features. However, for the majority of classifiers, $GDep$ models outperform $GInd$ models for both tasks, demonstrating the robustness of $GDep$ features for sentiment analysis.

In Table 6 we report results for subjectivity and polarity classification using the best performing classifiers (as shown in Figure 4c) :

- Logistic Regression (LR) (Genkin et al., 2007) for $GInd-J$ and $GDep-J$ models.
- SVM model with radial-based kernel for $GInd-D$ and $GDep-D$ models. We use LibSVM implementation (EL-Manzalawy and Honavar, 2005).

Each $\Delta R(\%)$ row shows the relative percent improvements in terms of precision P, recall R, F-measure F and accuracy A for $GDep$ compared to $GInd$ models. Our results show that differences in subjective language across genders can be exploited to improve sentiment analysis, not only for English but for multiple languages. For Spanish and Russian results are lower for subjectivity classification, we suspect, because lexical features V are already inflected for gender and set-count features are down-weighted by the classifier. For polarity classification, on the other hand, gender-dependent features provide consistent, significant improvements (1.5-2.5%) across all languages.

As a reality check, Table 6 also reports accuracies (in A^{rand} columns) for experiments that use random permutations of male and female subjective terms, which are then encoded as gender-dependent set-count features as before. We found that all gender-dependent models, $GDep-J$ and $GDep-D$, outperformed their random equivalents for both subjectivity and polarity classification (as reflected by relative accuracy decrease ↓ for A^{rand} compared to A). These results further confirm the existence of gender bias in subjective language for any of our three languages and its importance for sentiment analysis.

Finally, we check whether encoding gender as a binary feature would be sufficient to improve sentiment classification. For that we encode fea-

	English		Spanish		Russian	
	P	R	P	R	P	R
(a)	0.73	0.93	0.68	0.63	0.66	0.74
(b)	0.72	0.94	0.69	0.64	0.66	0.74
(c)	0.78	0.83	0.71	0.63	0.66	0.72
(d)	0.69	0.93	0.71	0.62	0.65	0.76
(e)	**0.80**	**0.83**	**0.72**	**0.65**	**0.68**	**0.73**

Table 7: Precision and recall results for polarity classification: encoding gender as a binary feature vs. gender-dependent features $GDep - J$.

tures such as: (a) unigram term frequencies V, (b) term frequencies and gender binary $V + GBin$, (c) gender-independent $GInd$, (d) gender-independent and gender binary $GBin + GInd$, and (e) gender-dependent $GDep - J$. We train logistic-regression model for polarity classification and report precision and recall results in Table 7. We observe that including gender as a binary feature does not yield significant improvements compared to $GDep - J$ for all three languages.

6 Conclusions

We presented a qualitative and empirical study that analyses substantial and interesting differences in subjective language between male and female users in Twitter, including hashtag and emoticon usage across cultures. We showed that incorporating author gender as a model component can significantly improve subjectivity and polarity classification for English (2.5% and 5%), Spanish (1.5% and 1%) and Russian (1.5% and 1%). In future work we plan to develop new models for joint modeling of personalized sentiment, user demographics e.g., age and user preferences e.g., political favorites in social media.

Acknowledgments

The authors thank the anonymous reviewers for helpful comments and suggestions.

References

Muhammad Abdul-Mageed, Mona T. Diab, and Mohammed Korayem. 2011. Subjectivity and sentiment analysis of modern standard Arabic. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 587–591.

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media (LSM'11)*, pages 30–38.

Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. 2007. Mining the blogosphere: Age, gender and the varieties of self-expression. *First Monday*, 12(9). http://www.firstmonday.org/ojs/index.php/fm/article/view/2003/1878.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 2200–2204.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2012. Gender in Twitter: styles, stances, and social networks. *Computing Research Repository*.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2008. A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 2764–2767.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on Twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 36–44.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific Twitter collections. In *Proceedings of the Second Workshop on Language in Social Media (LSM'12)*, pages 65–74.

Adam Bermingham and Alan F. Smeaton. 2010. Classifying sentiment in microblogs: Is brevity an advantage? In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*, pages 1833–1836.

Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in Twitter streaming data. In *Proceedings of the 13th International Conference on Discovery Science (DS'10)*, pages 1–15.

Bonka Boneva, Robert Kraut, and David Frohlich. 2001. Using email for personal relationships: The difference gender makes. *American Behavioral Scientist*, 45(3):530–549.

John D. Burger, John C. Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309.

Pedro Henrique Calais Guerra, Adriano Veloso, Wagner Meira Jr, and Virgílio Almeida. 2011. From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the17th International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pages 150–158.

Lu Chen, Wenbo Wang, Meenakshi Nagarajan, Shaojun Wang, and Amit P. Sheth. 2012. Extracting diverse sentiment expressions with target-dependent polarity from Twitter. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media (ICWSM'12)*, pages 50–57.

Ilia Chetviorkin and Natalia V. Loukachevitch. 2012. Extraction of Russian sentiment lexicon for product meta-domain. In *Proceedings of the 25rd International Conference on Computational Linguistics (COLING'12)*, pages 593–610.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The Fisher corpus: a resource for the next generations of speech-to-text. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, pages 69–71.

Simon Clematide and Manfred Klenner. 2010. Evaluation and extension of a polarity lexicon for German. In *Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA'10)*, pages 7–13.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using Twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 241–249.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'10)*, pages 1277–1287.

Yasser EL-Manzalawy and Vasant Honavar, 2005. *WLSVM: Integrating LibSVM into Weka Environment*. http://www.cs.iastate.edu/ yasser/wlsvm.

Teng-Kai Fan and Chia-Hui Chang. 2009. Sentiment-oriented contextual advertising. *Advances in Information Retrieval*, 5478:202–215.

Nikesh Garera and David Yarowsky. 2009. Modeling latent biographic attributes in conversational genres. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 710–718.

David Gefen and Catherine M. Ridings. 2005. If you spoke as she does, sir, instead of the way you do: a sociolinguistics perspective of gender differences in virtual communities. *SIGMIS Database*, 36(2):78–92.

Alexander Genkin, David D. Lewis, and David Madigan. 2007. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49:291–304.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 42–47.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: telephone speech corpus for research and development. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'92)*, pages 517–520.

Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. 2009. Stylometric analysis of bloggers age and gender. In *Proceedings of AAAI Conference on Weblogs and Social Media*, pages 214–217.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Exploratory Newsletter*, 11(1):10–18.

Janet Holmes and Miriam Meyerhoff. 2004. *The Handbook of Language and Gender*. Blackwell Publishing.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent Twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'07)*, pages 1075–1083.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM'11)*, pages 538–541.

Guangxia Li, Steven Hoi, Kuiyu Chang, and Ramesh Jain. 2010. Micro-blogging sentiment detection by collaborative online learning. In *Proceedings of IEEE 10th International Conference on Data Mining (ICDM'10)*, pages 893–898.

Hao Li, Yu Chen, Heng Ji, Smaranda Muresan, and Dequan Zheng. 2012. Combining social cognitive theories with linguistic features for multi-genre sentiment analysis. In *Proceedings of the 26th Pacific Asia*

Conference on Language,Information and Computation (PACLIC'12), pages 27–136.

Ronald Macaulay. 2006. Pure grammaticalization: The development of a teenage intensifier. *Language Variation and Change*, 18(03):267–283.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2012. Multilingual subjectivity and sentiment analysis. In *Proceedings of the Association for Computational Linguistics (ACL'12)*.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saif Mohammad and Tony Yang. 2011. Tracking sentiment in mail: How genders differ on emotional axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA'11)*, pages 70–79.

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'10)*, pages 207–217.

Brendan O'Connor, Jacob Eisenstein, Eric P. Xing, and Noah A. Smith. 2010. A mixture model of demographic lexical variation. In *Proceedings of NIPS Workshop on Machine Learning in Computational Social Science*, pages 1–7.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319.

Jahna Otterbacher. 2010. Inferring gender of movie reviewers: exploiting writing style, content and metadata. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*, pages 369–378.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 1320–1326.

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in Spanish. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 3077–3081.

Rosalind W. Picard. 1997. *Affective computing*. MIT Press.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL'09)*, pages 675–682.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in Twitter. In *Proceedings of the Workshop on Search and Mining User-generated Contents (SMUC'10)*, pages 37–44.

Philip Resnik. 2013. Getting real(-time) with live polling. http://vimeo.com/68210812.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'03)*, pages 105–112.

Harold Schiffman. 2002. Bibliography of gender and language. http://ccat.sas.upenn.edu/ haroldfs/popcult/bibliogs/gender/genbib.htm.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*, pages 254–263.

Will Styler. 2011. The EnronSent Corpus. Technical report, University of Colorado at Boulder Institute of Cognitive Science. http://verbs.colorado.edu/enronsent/.

Jane Sunderland, Ren-Feng Duann, and Paul Bake. 2002. Gender and genre bibliography. www.ling.lancs.ac.uk/pubs/clsl/clsl122.pdf.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Sali A. Tagliamonte. 2006. *Analysing Sociolinguistic Variation*. Cambridge University Press, 1st. Edition.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pages 1397–1405.

Mike Thelwall, David Wilkinson, and Sukhvinder Uppal. 2010. Data mining emotion in social network communication: Gender differences in MySpace. *Journal of the American Society for Information Science and Technology*, 61(1):190–199.

Peter D. Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*, pages 417–424.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Proceedings of*

the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'10), pages 777–785.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring sentiment in social media: Bootstrapping subjectivity clues from multilingual Twitter streams. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, pages 505–510.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in Twitter: A graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM'11)*, pages 1031–1040.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (AAAI'00*, pages 735–740.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'05)*, pages 347–354.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*, pages 172–182.

Opinion Mining in Newspaper Articles by Entropy-based Word Connections

Thomas Scholz and **Stefan Conrad**
Heinrich-Heine-University
Institute of Computer Science
D-40225 Düsseldorf, Germany
{scholz,conrad}@cs.uni-duesseldorf.de

Abstract

A very valuable piece of information in newspaper articles is the tonality of extracted statements. For the analysis of tonality of newspaper articles either a big human effort is needed, when it is carried out by media analysts, or an automated approach which has to be as accurate as possible for a Media Response Analysis (MRA). To this end, we will compare several state-of-the-art approaches for Opinion Mining in newspaper articles in this paper. Furthermore, we will introduce a new technique to extract entropy-based word connections which identifies the word combinations which create a tonality. In the evaluation, we use two different corpora consisting of news articles, by which we show that the new approach achieves better results than the four state-of-the-art methods.

1 Introduction

The Web keeps many potentially valuable opinions in news articles which are partly new online articles or uploaded print media articles. Many companies or organisations such as political parties or even distinguished public figures perform a Media Response Analysis (MRA) (Watson and Noble, 2007) in order to analyse the output of their effort in public relations. So, an opinion-oriented analysis of news articles is important, because the tonality (Watson and Noble, 2007; Scholz et al., 2012a) is the key indicator of a MRA. A purely manual solution implies a big human effort for so-called media analysts, because they have to read and rate approx. 200 to 800 news articles each week.

As a consequence, an automated Opinion Mining solution is very attractive. At the same time, Opinion Mining in newspaper articles appears to be difficult, because not all parts of news articles are as subjective (Balahur et al., 2010) as reviews, for example. Also, different parts of one article can contain different opinions (Watson and Noble, 2007). Therefore, we work with extracted statements of news articles, in which a sequence of consecutive sentences has the same tonality value. At the same time, some approaches focus more on differentiating only between positive and negative news and leave out neutral examples (Taboada et al., 2011; Scholz et al., 2012b). Conversely, we have noticed that even if the used words in the news domain are quite similar, the tonality which the words express can be different, especially if neutral examples are involved (cf. section 3.1). We propose this task formulation:

Problem definition: Let $s \subseteq d$ be a statement and document d represents a newspaper article. The task is to determine the tonality y for a given statement s, consisting of k words:

$$t : s = (w_1, w_2, ..., w_k) \mapsto \\ y \in \{\text{positive,neutral,negative}\} \quad (1)$$

Normally, a statement consists of one up to four sentences. But also longer statements are possible, but they appear less frequently in a MRA. An automated approach (Scholz and Conrad, 2013) for the extraction of statements already exists. The approach applies machine learning to extract relevant sentences from a collection of news articles and combine them to statements. So, we concentrate on the tonality classification, which is not provided by

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1828–1839,
Seattle, Washington, USA, 18-21 October 2013.

the approach for the statements extraction (Scholz and Conrad, 2013). Furthermore, we define the polarity of sentiment as the distinction between positive and negative sentiment and the subjectivity as the distinction between subjective (positive and negative) statements and neutral statements.

The following example is a positive statement from an article in The Telegraph (8th Aug 2012) which deals with the prospects of British companies in Africa:

- **Example statement (positive):** *There are structural factors behind the African growth story: a growing and sizeable population which is increasingly urbanised with disposable income; growing political stability; and a financial services industry that is still in its infancy.*

The so-called pressrelations dataset (Scholz et al., 2012a), which represents a publicly available corpus[1] of a MRA on German news articles, contains 1,521 annotated statements. Since this is the only publicly available corpus of a MRA as far as we know, we perform our experiments in German. We are aware of the fact that viewpoints play a significant role in a newspaper, but since we concentrate on the determination of the tonality, the extraction of viewpoints can be solved in a separate step (Scholz and Conrad, 2012). This is possible, because the tonality of a statement can be determined without knowledge of the viewpoint in almost all cases. The only exception is a statement with multiple viewpoints and different tonalities, but these statements are very rare (cf. also section 4.1).

Our approach learns a graph from an annotated collection of statements, in which nodes and edges model tonality-bearing word connections. For unseen statements, we recognize subgraphs of the learned graph, compare two weighting methods for extracting different tonality features, and classify the statements by a support vector machine.

In this paper, we describe four state-of-the-art techniques for Opinion Mining in the next section about related work. In the third section, we introduce our graph-based and entropy-based approach to calculate the tonality features T. We will evaluate our approach against the state-of-the-art methods in section 4, before we conclude in the last section.

[1] http://www.pressrelations.de/research/

2 Related Work

Opinion Mining and Sentiment Analysis represent a broad subject area (Pang and Lee, 2008).

The different contributions reach from applying Opinion Mining in reviews and recommending new multimedia products for individuals (Qumsiyeh and Ng, 2012) to sentiment analyses for different topics in social media (Wang et al., 2011) or the creation of sentiment dictionaries (Baccianella et al., 2009). In this paper, we focus on state-of-the-art methods for Opinion Mining which differ from each other in their methodology.

In the news domain, Wilson et al. (2009) developed a word-based classification approach which can extract contextual polarity. This method (denoted as **Wilson**) uses a lexicon and POS-tagging to generate word features and sentence features. Moreover, it also uses deep natural language analyses with dependency parse trees in order to calculate (general and polarity) modification features and structure features. Finally, they compute 32 features for neutral-polar classification and 10 features for the polarity classification. These features can be used by different kinds of machine learning techniques such as Ripper (Cohen, 1996) or BoosTexter (Schapire and Singer, 2000).

Based on a sentiment lexicon, Taboada et al. (2011) calculate the semantic orientation of opinion-bearing words (**SO-CAL**). They begin with a fine-grained dictionary of adjectives, adverbs, verbs, and nouns which have a score from -5 to +5. "Masterpiece" has a score of +5 and "monstrosity" of -5, for example. In addition, the approach takes intensifiers, negations, and irrealis (Taboada et al., 2011) into account and thereby modifies the score of the words through rules and formulas. SO-CAL identifies some special expressions and constructions, which tell the reader, that this text part does not really contain an actual opinion or sentiment. The linguistic term for this situation is called irrealis. Also, text-level features weight the final score by mere presence of the words.

In the field of customer reviews, Ding et al. (2008) also work with a dictionary, which even includes context-dependent words (positive, neutral, and negative words) as well as rules to identify the sentiment orientation of words (**Opinion**

Observer). The rules deal with negations, inter-sentence conjunctions, but-clauses, and the modifier "too". Furthermore, they extract relations between opinion words and corresponding product features. Thereby, a detailed analysis of product reviews is possible.

Sarvabhotla et al. (2011) propose to extract the subjective excerpt of a text (**RSUMM**). They construct two word-vectors: An average document frequency vector represents the most important and most specific word features for the given domain. Subsequently, an average subjective measure vector selects the most subjective terms. As a result, they require hardly any natural language preprocessing except a sentence splitter and a tokenizer. The final classification is accomplished by a SVM (SVM-Light (Joachims, 1999)).

For product reviews, graph-based approaches (Goldberg and Zhu, 2006; Wu et al., 2009) can increase the performance in cross-domain tasks (Ponomareva and Thelwall, 2012). By contrast, our graph nodes do not represent documents, but words to overcome the problem of similar bag-of-words representations (cf. next section).

One important resource for Opinion Mining in news is the MPQA corpus (Wiebe et al., 2005) which contains word and phrase-based annotation for 523 news articles. Unfortunately, since the corpus does not have statements and a statement-based tonality, it is not designed as a MRA. A slightly larger corpus, the pressrelations dataset (Scholz et al., 2012a) with 617 articles, is the result of a MRA in German. We use this corpus as one part of our evaluation.

3 Learning Tonality with Entropy-based Word Connections

3.1 Graph Model for Word Connections

To solve the Opinion Mining task for a MRA, we propose a graph-based approach to capture the opinion-bearing words and modifiers such as negations. In this way, our approach is able to recognize tonality-indicating structures (subgraphs) which provide precise information about the tonality, even if statements have a very similar bag-of-words representation and at the same time different tonalities. One could also say that we create a graph instead of a sentiment dictionary from training examples, as other approaches (Kaji and Kitsuregawa, 2007; Du et al., 2010) proceed.

In figure 1, simple examples are shown with a possible graph (the nodes and edges are taken from the given statements; of course, the graphs and weights become larger in practice). These simple examples are concentrated on nouns, verbs, and adverbs, but also examples with combinations of other categories are possible, such as, for example, different combinations of adjectives, nouns, and verbs: "This is a black day for the company", "The company is in the black", "The company is in the red" and "The company prevents to be in the red". Thus, even though the word representation is quite similar, the tonality can be different.

For opinion-bearing words, we use adjectives, nouns, verbs, and adverbs, which are widely acknowledged as opinion-bearing word categories (Bollegala et al., 2011; Remus et al., 2010; Taboada et al., 2011). Furthermore, we also include negation particles. Therefore, the vocabulary V is the set of words in lemma for one set of statements S. Thus, for every lemma $w \in V$, the approach creates one node v in the graph. A node v also contains the type information (adjective, noun, verb, adverb, or negation).

The edge e_{ij} shows the appearance of node v_i and v_j in combination with tonality y by means of a weight $\varepsilon_{i,j}$ (the sequence of the values in equation 2 is also used in figure 1 and 2).

$$\varepsilon_{ij} = (y_{ij\pi}, y_{ijo}, y_{ij\nu}) \tag{2}$$

$y_{ij\pi}$ is the number of co-occurrences of node v_i and v_j in positive statements within the same sentence. In analogy, y_{ijo} belongs to sentences of neutral statements and $y_{ij\nu}$ to sentences of negative statements. Figure 1 shows a small example for this calculation, too.

3.2 Generating Features for Learning

From a learned graph, we can combine different edges to calculate tonality features for an unseen statement s. An unseen statement is a statement, which is of course not used to learn the graph. We use all edges of the subgraph G_{sl} which contains the nodes for every lemma w_i in the l-th sentence of s.

1)	This solves the crisis.	(positive)
2)	This solves the crisis slowly.	(neutral)
3)	This intensifies the crisis.	(negative)

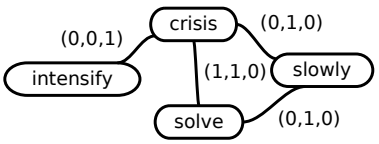


Figure 1: An example for different statements and a graph: The weights base on the three examples and their notation is (positive,neutral,negative).

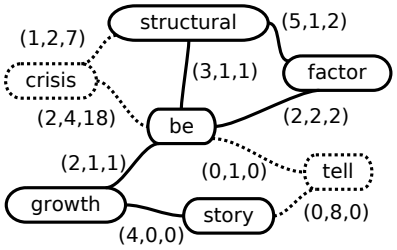


Figure 2: An example of a learned graph: The nodes and edges, which are drawn in solid lines, represent the recognized subgraph G_{sl} for the sentence "There are structural factors behind the African growth story.".

We explain this with an example. Assuming that our learned graph is shown in figure 2. It contains seven nodes and nine edges (also the nodes and edges in dashed lines). If we further assume that an unseen statement is the example of section 1. To keep this example short, we take the part until the colon as the first sentence of the statement: "There are structural factors behind the African growth story."

Our approach recognizes the nodes for "be", "structural", "factors", "growth", and "story". Thus, the subgraph G_{sl} for the first sentence ($l = 0$) would be the graph which is drawn in solid lines in figure 2. In this example, it is a connected graph, but it does not have to be.

We could also look for complete or connected graphs in the statement instead of using all edges. The largest complete graph would consist of the nodes "structural", "factor", and "be" in our example. But using all edges achieves better results, because this method provides all information. In addition, this method is quicker (search for largest complete or connected graph can be omitted, which would be an additional check).

If we have found our subgraphs G_{sl}, we can then compute the vectorial sum of all edges for one node v_i and we get the probability for a tonality y, if we observe v_i in the l-th sentence:

$$P(pos|v_i) = \frac{\sum\limits_{e_{ij} \in Gsl} y_{ij\pi}}{\sum\limits_{e_{ij} \in Gsl} y_{ij\pi} + y_{ij\nu}} \quad (3)$$

$$P(neg|v_i) = \frac{\sum\limits_{e_{ij} \in Gsl} y_{ij\nu}}{\sum\limits_{e_{ij} \in Gsl} y_{ij\pi} + y_{ij\nu}} \quad (4)$$

$$P(sub|v_i) = \frac{\sum\limits_{e_{ij} \in Gsl} y_{ij\pi} + y_{ij\nu}}{\sum\limits_{e_{ij} \in Gsl} y_{ij\pi} + y_{ijo} + y_{ij\nu}} \quad (5)$$

$$P(neu|v_i) = \frac{\sum\limits_{e_{ij} \in Gsl} y_{ijo}}{\sum\limits_{e_{ij} \in Gsl} y_{ij\pi} + y_{ijo} + y_{ij\nu}} \quad (6)$$

For the subjective class (sub), we add the appearance in positive statements ($y_{ij\pi}$) and negative statements ($y_{ij\nu}$). Otherwise we take the appearances in statements of the same class. The denominators of the polarity refer only to positive and negative appearances, while the denominators for the subjectivity refer to every tonality.

By calculating the vectorial sum, we combine several edges in order to estimate precise tonality scores. In this way, we can get the correct tonality score for the noun "crisis", if a sentence contains also "solve" and "slowly" ($\rightarrow$ more neutral) or "intensify" ($\rightarrow$ more negative) (cf. figure 1). And we get the correct tonality score for the adjective "structural", if a sentence includes also "crisis" ($\rightarrow$ negative) or the nodes "factor", "be", "growth", and "story" ($\rightarrow$ positive) (cf. figure 2).

We distinguish between different word categories (we have noticed that this creates better results than

just having a single feature for one statement). Thus, every category gets its own feature and every node only has a tonality value, if it belongs to the category of the feature. This does not mean that we only consider edges which connect two nodes with the same category; we divide the influence of different categories into different features:

$$T_{cat,z}(v_i) = \begin{cases} f_z(v_i) & \text{if } v_i \in cat \\ 0 & \text{if } v_i \notin cat \end{cases} \quad (7)$$

$cat \in \{adj, adv, n, v\}$ indicates the category of the node (adjectives, adverbs, nouns, or verbs) and z specifies the type of feature. One type shows the difference between positive and negative polarity ($z = pol$), for the other type we replace the positive class by the subjective one (the sum of positive and negative) and the negative by a neutral one in order to differentiate between neutral and non-neutral examples ($z = sub$). As a result, we calculate eight features (see table 1) for the tonality, two for each important word category. For the weighting, we apply and compare two methods, presented in the next sections.

3.2.1 Kullback-Leibler Weighting

For the final score, we can use the Kullback-Leibler divergence (relative entropy) (Kullback and Leibler, 1951) of P_2 from P_1:

$$D_{KL}(P_1||P_2) = \sum_{x \in X} P_1(x) \log \frac{P_1(x)}{P_2(x)} \quad (8)$$

To measure the information about tonality, we can define our tonality scores based on the divergence between the two category pairs:

$$f_{pol}(v_i) = D_{KL}(P(pos|v_i)||P(neg|v_i)) \quad (9)$$

$$f_{sub}(v_i) = D_{KL}(P(sub|v_i)||P(neu|v_i)) \quad (10)$$

Here, we measure the information lost, if $P(neg|v_i)$ approximates $P(pos|v_i)$, for example. The Kullback-Leibler is an asymmetric measure, so a switch of the distributions would give a different result. This is one reason why we prefer our second method, but we evaluate both in order to find out how important the choice of the weighting method is.

3.2.2 Entropy-summand Weighting

Also, the basic idea of the entropy (Shannon, 1948) can be applied to extract the importance of the edges for the tonality.

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) \quad (11)$$

Here, the $p(x_i)$ refer to the probabilities in the equations 3 to 6. We add or subtract the entropy-summand of the assumed tonality class for one node v_i to/from a perfect state (normalized to 1 and -1):

$$f_{pol}(v_i) = \begin{cases} 1 + P(pos|v_i) * \log_2(P(pos|v_i)) \\ \text{if } P(neg|v_i) \leq P(pos|v_i) \\ -1 - P(neg|v_i) * \log_2(P(neg|v_i)) \\ \text{otherwise} \end{cases} \quad (12)$$

$$f_{sub}(v_i) = \begin{cases} 1 + P(sub|v_i) * \log_2(P(sub|v_i)) \\ \text{if } P(neu|v_i) \leq P(sub|v_i) \\ -1 - P(neu|v_i) * \log_2(P(neu|v_i)) \\ \text{otherwise} \end{cases} \quad (13)$$

In this way, we measure how much disorder one node v_i provides for a certain tonality class. For a clearly positive node (appears only in positive statements), e.g., the disorder will be 0 and so $f_{pol}(v_i) = 1$ and also $f_{sub}(v_i) = 1$.

3.3 Final Scores and Classification

To compute the eight final features values (four for each z-class), we calculate the average scores of all nodes, which share the same category, over all sentences of the statement. If no nodes/edges could be recognized in an unseen statement, all features would be zero. We use a SVM[2] to classify the statements by the extracted features. This works according to the one-versus-all strategy for a non-binary classification, which achieved slightly better results than a one-versus-one strategy or a subjective-objective classification first and then a positive-negative classification. Linear kernels are used and the parameters are the default ones. This

[2]Rapidminer standard implementation (http://rapid-i.com/)

Polarity Features	Subjectivity Features
$T_{v,pol}$: polarity for edges with verbs	$T_{v,sub}$: subjectivity for edges with verbs
$T_{n,pol}$: polarity for edges with nouns	$T_{n,sub}$: subjectivity for edges with nouns
$T_{adv,pol}$: polarity for edges with adverbs	$T_{adv,sub}$: subjectivity for edges with adverbs
$T_{adj,pol}$: polarity for edges w. adjectives	$T_{adj,sub}$: subjectivity for edges w. adjectives

Table 1: Polarity and subjectivity features based on word connections

means that every class has the same priority, for instance.

By using only 8 features, we actually achieve better results if compared with the use of one edge as a feature, because we abstract from individual word combinations in order to prevent overfitting. We will demonstrate that in section 4, where this method of using all edges as features is denoted as the **graph edges** method. Another positive aspect of restricting the number of features to a constant limit is that we save computing time (for the calculation of distances within machine learning, e.g.), because the graphs can be large (cf. section 4).

4 Experiments

4.1 Data and Experimental Setup

We use two different datasets for our evaluation: The pressrelations dataset[3] (called **PDS**) contains 1,521 statements (446 positive, 492 neutral, 583 negative), and a real world dataset contains 8,500 statements (2,125 positive, 2,125 negative, 4,250 neutral) from 5,352 news items about a financial service provider, the so-called **Finance** dataset. Up to ten media analysts (professional experts in the field of MRA) annotate the extracted statements with a tonality. We have investigated their inter-annotator agreement. So, four analysts annotate the same statements from a small part of the statements. They achieve an agreement of 81.8% by using the simple accuracy metric. The PDS has an inter-annotator agreement of 88.06% (Cohen's kappa) (Scholz et al., 2012a). We do not use the viewpoint information contained in the PDS. This is not a problem, because the tonality of statements can be estimated without knowledge of the viewpoint in the most cases.

Nevertheless, a statement can have two different viewpoints in a MRA. This is the case for 116 statements (approx. 7.62%) of the pressrelations dataset and 279 statements of the Finance dataset (approx. 3.28%). Statements can have two different tonalities for different viewpoints, but this is rarely the case (for less than 3.56% of the pressrelations statements and less than 0.17% of the statements of the Finance dataset). One of these examples is the following statement, which is a translated statement of the PDS:

- **Example:** *The logical consequence would be a substantial increase of the subsidies, which the SPD fraction has demanded several times. But the government has limited the funding for 2011 and a too slight rise is planned for 2012.* (Code A: positive, SPD; Code B: negative, CDU)

At the time of the creation of this dataset, the SPD is the biggest opposing party of the CDU in Germany. The CDU is the governing party under its chairwoman Chancellor Merkel. We keep these statements within the dataset, because this case can occur in a MRA. However, we will show that this situation does not irritate our approach too much.

We use approx. 30% of the statements, that is 420 statements (the first 140 positive, neutral, or negative statements) or 2,500 statements (the first 625 positive or negative and the first 1,250 neutral statements) in order to create our graph (the graph has 41,470 or 154,001 edges, resp.). For POS-tagging, identification of negations, and lemmatisation, we apply the TreeTagger (Schmid, 1995). Unless otherwise stated, 20% of the remaining statements (220 and 1,200 statements) are the training set for the SVM and the rest is test set. The size of the test is so large, because we are aiming at a real significance of the solution which can actually be operated in practice.

[3] http://www.pressrelations.de/research/

4.2 Adapting the State-of-the-Art Approaches for a German MRA

For the approaches of Ding et al. (2008), Wilson et al. (2009), and Taboada et al. (2011) we need a sentiment dictionary. Thus, we use the same statements which we use for the creation of our graphs for the creation of a dictionary as one variant.

To create the lexicon of subjectivity clues for the method of **Wilson** et al. (2009), all words which appear more often in neutral statements get the *prior polarity* neutral. For all other words, we calculate the number of appearances in positive statements minus the appearances in negative statements divided by all appearances. A positive word has a value of over 0.2, a negative word has a value of less than -0.2 and the rest has the prior polarity *both*. A positive word with a value above 0.6 belongs to the reliability class *strongsubj*, the other positive words are *weaksubj*. We treat the negative words analogously. We use the Stanford Parser for German (Rafferty and Manning, 2008) to calculate the dependency trees for the sentences (Wilson et al., 2009), in order to extract the General Modification Features, the Polarity Modification Features and the Structure Features. The lists of intensifiers, copular verbs, modals, negations, and polarity shifters are translated by a domain expert, who also added such elements which are not direct translations, but have the same function. The result of this method is a classification of words and phrases. Thus, for a statement classification, we classify the words of the statements and the class of the most frequently used words is the class of the statement (ambiguous statements are classified as the most frequent class). According to the authors, we apply the best machine learning techniques for the word classification (BoosTexter for tonality classification and Ripper for Subjectivity Analysis with parameters as in (Wilson et al., 2009)).

For **Opinion Observer** (Ding et al., 2008), we also identify neutral words if they appear more often in neutral than in subjective statements and subjective words are positive if they appear more often in positive than in negative statements and vice versa for negative words. In contrast to Opinion Mining in customer reviews, we exchange product features through statements and calculate the orientation of opinions for all statements with their opinion orientation algorithm. For this purpose, we adapt the negation rules, the but-clause rule, the inter-sentence conjunction rule, and the "too" rules for German (by translating important words such as "but" or the negations).

SO-CAL (Taboada et al., 2011) needs dictionaries with sentiment values from -5 to +5 with intervals of one. Thus, we use the same scores as the Wilson method and a word with a value above 0.818 to 1 gets a sentiment score of +5 and so on. This means, that neutral words also exist. Our domain expert translated the list of intensifiers (amplifiers and downtoners) and negations, as well as the expert also added missing elements. The authors propose two approaches for the negation search. We use the second, more conservative approach, because this approach works better according to the authors. Also, we use the value 4 for the negation shift. Furthermore, we implement the algorithm of irrealis blocking and translate the list of irrealis markers (modal verbs, conditional markers, negative polarity items, private-state verbs (Taboada et al., 2011)).

For all dictionary-based methods (Wilson, Opinion Observer, SO-CAL), we also evaluate an additional variant which use a sentiment dictionary and not the statements which we use to construct the graphs on each fold. We apply the SentiWS (Remus et al., 2010) for this purpose. As the SentiWS has sentiment values between -1 and 1, we apply similar procedures to construct the method-specific dictionaries as described above: For SO-CAL, it is the same procedure by using the SentiWS values, positive words has a score above 0.33 for Wilson and Opinion Observer, *strongsubj* words have an absolute value above 0.66 and so on. The methods are denoted as *method* (dictionary).

RSUMM (Sarvabhotla et al., 2011) needs less specific adaptation, because only a sentence splitter and a tokenizer are needed. So, RSUMM is very language-independent. We test two versions of this method: one includes the optimization step to estimate the best values for X and Y (notated as RSUMM(X%, Y%)) and the other version (RSUMM(100%)) does without this step, because we believe that every sentence is important in the statements and also because more words mean more information about the tonality in our domain.

We use the sets for the creation of the graphs and lexicons as the validation dataset (VDS) (Sarvabhotla et al., 2011) and the subjectivity dataset (SDS) (Sarvabhotla et al., 2011). As in (Sarvabhotla et al., 2011), we apply the SVMLight package[4] for classification.

Opinion Observer (Ding et al., 2008) and SO-CAL (Taboada et al., 2011) do not use supervised learning. Therefore, we have also added our SVM in order to classify the statements based on the scores of Opinion Observer and SO-CAL (as shown in tables with (+ SVM)).

4.3 Results

Table 2 and 4 (left side) show the results on the pressrelations dataset (PDS) and table 3 and 4 (right side) show the results on Finance. Table 2 and 3 present the tonality classification (positive, neutral, negative) and table 4 displays the Subjectivity Analysis (subjective, neutral).

Word connections (Entropy-summand) achieve the best results with 63.45% accuracy on PDS (more than 15% better than Wilson, which is the best of the 'classical' state-of-the-art methods) and best results on Finance with 65.17% (more than 4% better than RSUMM, which comes in second). The weighting of the edges through the Entropy-summand performs better than the Kullback-Leibler weighting on both datasets, so we use the Entropy-summand weighting for all further experiments.

Also, the improved methods (RSUMM(100%), Opinion Observer (+ SVM), and SO-CAL(+ SVM)) get better results in the majority of cases (the improvement of SO-CAL is more than 13% on PDS and more than 4% on Finance, e.g.). Furthermore, the variants of the methods, which are expanded by a general sentiment dictionary, perform rather worse. The 'classical' Opinion Observer performs better with a general sentiment dictionary, while Wilson tends to achieve worse results in this variant.

Wilson (without an additional dictionary) achieves an accuracy of 42.91% on PDS (Subjectivity Analysis 69.36%) and 48.67% on Finance (Subjectivity Analysis 60.96%) for their word classification. The accuracy of the dictionary variant is 43.44% on PDS and 40.12% on Finance. Therefore, the tonality classification by the most frequent word class seems appropriate for this task and method, because this method achieves better results in the classification of statements than on the word level.

The findings of RSUMM are ambiguous. The 'classical' RSUMM with parameter optimization does not perform very well on PDS, but it performs well on Finance with a high proportion of sentences and words (RSUMM(90%,95%)). Also, if we use all sentences and all features (RSUMM(100%)) we obtain better results on Finance and PDS. This fits in with our assumption that every sentence of a statement is important and that more words lead to more tonality information. The number of word features for RSUMM(100%) is 4,985 features for one statement on PDS and 13,608 features on Finance. After the parameter optimization the size is 974 word features on PDS (RSUMM(80%,20%)) and 12,248 features on Finance (RSUMM(90%,95%)).

The outcomes of this study suggest that methods which include machine learning techniques tend to perform better than unsupervised techniques. The results of the approaches which we expand with a SVM support this conclusion. As mentioned before, only the graph edges obtain a not so high accuracy. This shows the importance of the aggregation of the edges and entropy-based weighting.

We evaluate the influence of the different input sizes and so we performed experiments with 5%, 10%, 40%, and 80% training for machine learning as well as 210 and 840 statements for the creation of dictionaries/graphs on PSD (0.17% training for 210 statements and 0.32% training for 840 statements in order to create the same size of training according to the results of 420 statements). The results are shown in table 5. Opinion Observer and SO-CAL are written in italics, because the results on the left side (size of the training set) belongs to their (+ SVM) variants and the results on the right side are the 'classical' methods with no supervised learning. These experiments show that our word connections remain very stable if the training set is decreased. However, it does not benefit from more training, especially when the training set is very large (80%). Opinion Observer and RSUMM(80%,20%) has the same problem. Nevertheless, it still receives the second-best results, even if another method gets a higher accuracy. However, in our opinion, it is more important

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

Method	Accuracy	Positive		Neutral		Negative	
		prec	rec	prec	rec	prec	rec
Wilson	0.4784	0.358	0.5	0.5423	0.5054	0.5540	0.4444
Wilson (dictionary)	0.4609	0.377	0.3366	0.3664	0.2963	0.5346	0.6223
Opinion Observer	0.3806	0.3732	0.1732	0.3481	0.8267	0.6098	0.1693
Opinion Observer (dictionary)	0.4468	0.5083	0.1993	0.4005	0.8693	0.576	0.2822
RSUMM(80%,20%)	0.403	-	0.0	-	0.0	0.403	1.0
SO-CAL	0.3279	0.3676	0.7353	0.2626	0.3551	0.8461	0.0248
SO-CAL (dictionary)	0.2852	0.2987	0.8464	0.2072	0.1307	0.0075	0.0002
Opinion Observer (+ SVM)	0.3825	-	0.0	0.252	0.1084	0.4037	0.8743
Opinion Observer (dictionary + SVM)	0.3235	0.52	0.2122	0.1322	0.0804	0.346	0.6
RSUMM(100%)	0.4801	0.4586	0.3025	0.8298	0.1354	0.4609	0.8789
SO-CAL (+ SVM)	0.4608	0.463	0.3061	0.3543	0.5699	0.6486	0.48
SO-CAL (dictionary + SVM)	0.3995	0.8235	0.0571	0.3559	0.9371	0.6306	0.2
graph edges	0.5482	0.4313	0.551	0.6578	0.5175	0.5831	0.5714
our approach (Kullback-Leibler)	0.5778	0.5	0.302	0.6642	0.6154	0.5534	0.74
our approach (Entropy-summand)	**0.6345**	0.5346	0.4735	0.6989	0.6818	0.6442	0.7086

Table 2: Results of the experiments on the **PDS**

Method	Accuracy	Positive		Neutral		Negative	
		prec	rec	prec	rec	prec	rec
Wilson	0.5602	0.4206	0.188	0.6358	0.7329	0.4706	0.5872
Wilson (dictionary)	0.4088	0.3678	0.3291	0.5618	0.339	0.3367	0.6132
Opinion Observer	0.4357	0.3641	0.0947	0.5033	0.713	0.2449	0.222
Opinion Observer (dictionary)	0.4583	0.3275	0.186	0.5325	0.664	0.3404	0.3193
RSUMM(90%,95%)	0.6092	0.4433	0.4840	0.731	0.6145	0.5866	0.7233
SO-CAL	0.3478	0.2992	0.5993	0.384	0.373	0.8519	0.046
SO-CAL (dictionary)	0.2905	0.2669	0.9207	0.4429	0.1203	0.001	0.0007
Opinion Observer (+ SVM)	0.4852	0.3384	0.0914	0.496	0.9269	-	0.0
Opinion Observer (dictionary + SVM)	0.4577	0.3299	0.187	0.5118	0.6649	0.3384	0.3177
RSUMM(100%)	0.6088	0.4428	0.4823	0.731	0.6145	0.5854	0.7233
SO-CAL (+ SVM)	0.3921	0.2986	0.7479	0.4573	0.1074	0.599	0.601
SO-CAL (dictionary + SVM)	0.4762	0.3862	0.341	0.544	0.6206	0.3878	0.3244
graph edges	0.5875	0.4437	0.3633	0.6444	0.7096	0.5816	0.5708
our approach (Kullback-Leibler)	0.561	0.3868	0.5445	0.7659	0.5524	0.5201	0.5951
our approach (Entropy-summand)	**0.6517**	0.53	0.5675	0.7714	0.6527	0.5946	0.7351

Table 3: Results of the experiments on **Finance**

Method	Accuracy	Subjective		Objective		Accuracy	Subjective		Objective	
		prec	rec	prec	rec		prec	rec	prec	rec
Wilson	0.6818	0.7251	0.8602	0.4970	0.2975	0.6307	0.6228	0.6649	0.6399	0.5966
Wilson (dictionary)	0.7029	0.7742	0.8636	0.2871	0.179	0.5247	0.5296	0.7944	0.5069	0.2305
Opinion Observer	0.4496	0.7698	0.2724	0.3481	0.8267	0.5047	0.508	0.2963	0.5033	0.713
Opinion Observer (dictionary)	0.5422	0.8635	0.3885	0.4005	0.8693	0.5405	0.5538	0.417	0.5325	0.664
RSUMM(80%,20%)/(90%,95%)	0.3269	-	0.0	0.3269	1.0	0.6919	0.7307	0.6170	0.6630	0.7682
SO-CAL	0.5250	0.7373	0.4686	0.3632	0.6449	0.6127	0.616	0.5983	0.6095	0.627
SO-CAL (dictionary)	0.4378	0.7928	0.235	0.3481	0.8693	0.5155	0.5571	0.1513	0.509	0.8797
Opinion Observer (+ SVM)	0.6061	0.6636	0.8454	0.252	0.1084	0.494	0.4665	0.0636	0.496	0.9269
Opinion Observer (dictionary + SVM)	0.4109	0.88	0.1479	0.3508	0.958	0.5327	0.5732	0.2667	0.5204	0.8003
RSUMM(100%)	0.7083	0.7014	0.9865	0.8298	0.1354	0.6975	0.7424	0.6137	0.6654	0.7829
SO-CAL (+ SVM)	0.5153	0.7485	0.4252	0.3702	0.7028	0.6231	0.7415	0.3814	0.582	0.8663
SO-CAL (dictionary + SVM)	0.3598	0.878	0.0605	0.3345	0.9825	0.511	0.5481	0.1421	0.5055	0.8822
graph edges	0.7037	0.6983	0.9882	0.8205	0.1119	0.6302	0.7821	0.3639	0.5840	0.898
our approach (Kullback-Leibler)	0.7662	0.8215	0.8353	0.6449	0.6224	0.7006	0.6753	0.7761	0.735	0.6247
our approach (Entropy-summand)	**0.7707**	0.8478	0.8050	0.6329	0.6993	**0.739**	0.7179	0.7898	0.7649	0.6878

Table 4: Subjectivity Analysis on **PDS** (left side) and on **Finance** (right side)

Method	0.05	0.1	0.2	0.4	0.8	210	420	840
Wilson	0.4388	0.4743	0.4784	0.5514	0.5795	**0.5275**	0.4784	0.5553
Opinion Observer	0.3403	0.3683	0.3825	0.3979	0.3591	0.3585	0.3806	0.3822
SO-CAL	0.4579	0.439	0.4608	0.4402	0.4818	0.3509	0.3279	0.2702
RSUMM(80%,20%)	0.4063	0.4046	0.403	0.3949	0.3636	0.3226	0.403	0.4557
RSUMM(100%)	0.2964	0.448	0.4801	0.5265	**0.6318**	0.489	0.4801	0.5529
our approach (Entropy-summand)	**0.5717**	**0.5883**	**0.6345**	**0.6278**	0.5818	0.5224	**0.6345**	**0.6452**

Table 5: Different sizes of the training set and the dictionaries/graphs

Features	Level(Wilson)	Level(SO-CAL)	Features	Level(Wilson)	Level(SO-CAL)
$T_{v,pol}$	$-----$	nsc	$T_{v,sub}$	nsc	$+++++$
$T_{n,pol}$	$-----$	$---$	$T_{n,sub}$	$-----$	$++$
$T_{adv,pol}$	$-----$	$-$	$T_{adv,sub}$	$-----$	nsc
$T_{adj,pol}$	$-----$	$-----$	$T_{adj,sub}$	$-----$	nsc
$T_{cat,pol}$	$-----$	nsc	$T_{cat,sub}$	$--$	$+++++$
$T_{cat,z}$(all)	$+++++$	$+++++$			

Table 6: Significance of the tonality features T to the baselines Wilson and SO-CAL

to obtain good results on small training sizes, because over 75% for training would mean that a possible practical implementation would not save much human effort.

4.4 Statistical Significance of the Features

We perform a 10-fold cross validation with our method, Wilson (as the best 'classical' state-of-the-art-method) and SO-CAL (+ SVM) on the pressrelations dataset in order to evaluate the contribution of single tonality features. Our approach (Entropy-summand with all features) achieves an accuracy of 61.94%, while Wilson gets 56.36% and SO-CAL 46.68%. As an analogy to Wilson et al. (2009), we carry out a two-sided t-test with Wilson and SO-CAL (+ SVM) as baselines. The results are shown in table 6. The pluses indicate a significant increase to the baseline, the minuses show a significant decrease. For one sign, changes are significant at the level $p \leq 0.1$, two signs mean $p \leq 0.05$, three signs $p \leq 0.025$, four signs $p \leq 0.01$ and five signs indicate $p \leq 0.005$. "nsc" stands for no significant change.

As shown in table 6, the features with type $z = sub$ are more important than the polarity features. In the categories, the nouns and verbs are more significant than adjectives and adverbs (adverbs are a little stronger in the polarity difference). Combining all features produces a very significant increase against both baselines.

5 Conclusion

We have shown that the word connections outperform state-of-the-art-methods in most cases of tonality classification for a MRA. As a major advantage, our approach does not need much training data. The combination of all tonality features is a significant increase against both baselines, too. The findings show that the word connections in combination with the entropy weighting allow to learn the tonality structure of different word combinations accurately, even though the training size is small. This is a major advantage for a solution, which operates in practice for media analysts, which have to analyse articles for a MRA.

So, this approach in combination with an extraction of statements (Scholz and Conrad, 2013) and the determination of viewpoints (Scholz and Conrad, 2012) represents a fully automated solution in order to perform Opinion Mining for a MRA.

Acknowledgments.

This work is funded by the German Federal Ministry of Economics and Technology under the ZIM-program (Grant No. KF2846501ED1). The authors want to thank the anonymous reviewers for their very helpful comments.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2009. Multi-facet rating of product reviews. In

Proc. of the 31th European Conf. on IR Research on Advances in Information Retrieval, ECIR '09, pages 461–472.

Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik van der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. 2010. Sentiment analysis in the news. In *Proc. of the 7th intl. conf. on Language Resources and Evaluation (LREC'10)*, pages 2216–2220.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 132–141.

William W. Cohen. 1996. Learning trees and rules with set-valued features. In *Proc. of the 13th national conference on Artificial intelligence - Volume 1*, AAAI'96, pages 709–716. AAAI Press.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proc. of the Intl. Conf. on Web search and web data mining*, WSDM '08, pages 231–240.

Weifu Du, Songbo Tan, Xueqi Cheng, and Xiaochun Yun. 2010. Adapting information bottleneck method for automatic construction of domain-oriented sentiment lexicon. In *Proc. of the 3rd ACM intl. conf. on Web search and data mining*, WSDM '10, pages 111–120.

Andrew B. Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *Proc. of the 1st Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 45–52.

Thorsten Joachims. 1999. Advances in kernel methods. chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In *Proc. of the 2007 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1075–1083.

Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Natalia Ponomareva and Mike Thelwall. 2012. Do neighbours help?: an exploration of graph-based algorithms for cross-domain sentiment classification. In *Proc. of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 655–665.

Rani Qumsiyeh and Yiu-Kai Ng. 2012. Predicting the ratings of multimedia items for making personalized recommendations. In *Proc. of the 35th intl. ACM SIGIR conf. on Research and development in information retrieval*, SIGIR '12, pages 475–484.

Anna N. Rafferty and Christopher D. Manning. 2008. Parsing three german treebanks: lexicalized and unlexicalized baselines. In *Proc. of the Workshop on Parsing German*, PaGe '08, pages 40–46.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. Sentiws – a publicly available german-language resource for sentiment analysis. In *Proc. of the 7th Intl. Conf. on Language Resources and Evaluation (LREC'10)*, pages 1168–1171.

Kiran Sarvabhotla, Prasad Pingali, and Vasudeva Varma. 2011. Sentiment classification: a lexical similarity based approach for extracting subjectivity in documents. *Inf. Retr.*, 14(3):337–353.

Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based systemfor text categorization. *Mach. Learn.*, 39(2-3):135–168.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proc. of the ACL SIGDAT-Workshop*, pages 47–50.

Thomas Scholz and Stefan Conrad. 2012. Integrating viewpoints into newspaper opinion mining for a media response analysis. In *Proc. of the 11th conf. on Natural Language Processing (KONVENS 2012)*, pages 30–38.

Thomas Scholz and Stefan Conrad. 2013. Extraction of statements in news for a media response analysis. In *Proc. of the 18th Intl. conf. on Applications of Natural Language Processing to Information Systems 2013 (NLDB 2013)*, pages 1–12.

Thomas Scholz, Stefan Conrad, and Lutz Hillekamps. 2012a. Opinion mining on a german corpus of a media response analysis. In *Proc. of the 15th International Conference on Text, Speech and Dialogue (TSD 2012)*, pages 39–46.

Thomas Scholz, Stefan Conrad, and Isabel Wolters. 2012b. Comparing different methods for opinion mining in newspaper articles. In *Proc. of the 17th Intl. conf. on Applications of Natural Language Processing to Information Systems 2012 (NLDB 2012)*, pages 259–264.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27:379–423.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based

methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proc. of the 20th ACM intl. conf. on Information and knowledge management*, CIKM '11, pages 1031–1040.

Tom Watson and Paul Noble, 2007. *Evaluating public relations: a best practice guide to public relations planning, research & evaluation*, chapter 6, pages 107–138. PR in practice series. Kogan Page.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433.

Qiong Wu, Songbo Tan, and Xueqi Cheng. 2009. Graph ranking for sentiment transfer. In *Proc. of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 317–320.

Collective Opinion Target Extraction in Chinese Microblogs

Xinjie Zhou, Xiaojun Wan* and Jianguo Xiao
Institute of Computer Science and Technology
The MOE Key Laboratory of Computational Linguistics
Peking University
No. 5, Yiheyuan Road, Beijing, China
{zhouxinjie,wanxiaojun,xiaojianguo}@pku.edu.cn

Abstract

Microblog messages pose severe challenges for current sentiment analysis techniques due to some inherent characteristics such as the length limit and informal writing style. In this paper, we study the problem of extracting opinion targets of Chinese microblog messages. Such fine-grained word-level task has not been well investigated in microblogs yet. We propose an unsupervised label propagation algorithm to address the problem. The opinion targets of all messages in a topic are collectively extracted based on the assumption that similar messages may focus on similar opinion targets. Topics in microblogs are identified by hashtags or using clustering algorithms. Experimental results on Chinese microblogs show the effectiveness of our framework and algorithms.

1 Introduction

Microblogging services such as Twitter[1], Sina Weibo[2] and Tencent Weibo[3] have swept across the globe in recent years. Users of microblogs range from celebrities to ordinary people, who usually express their emotions or attitudes towards a broad range of topics. It is reported that there are more than 340 million tweets per day on Twitter and more than 200 million on Sina Weibo. A tweet means a post on Twitter. Since we mainly focus on Chinese microblogs instead of Twitter in this paper, we will refer to a post as a message. Each message is limited to 140 Chinese characters and usually contains several sentences.

Currently, researches on microblog sentiment analysis have been conducted on polarity classification (Barbosa and Feng, 2010; Jiang el al., 2011; Speriosu et al., 2011) and have been proved to be useful in many applications, such as opinion polling (Tang et al., 2012), election prediction (Tumasjan et al., 2010) and even stock market prediction (Bollen et al., 2011). However, classifying microblog texts at the sentence level is often insufficient for applications because it does not identify the opinion targets. In this paper, we will study the task of opinion target extraction for Chinese microblog messages.

Opinion target extraction aims to find the object to which the opinion is expressed. For example, in the sentence "The sound quality is good!", "sound quality" is the opinion target. This task is mostly studied in customer review texts in which opinion targets are often referred as features or aspects (Liu, 2012). Most of the opinion target extraction approaches rely on dependency parsing (Zhuang et al., 2006; Jakob and Gurevych, 2010; Qiu et al., 2011) and are regarded as a domain-dependent task (Li et al., 2012a). However, such approaches are not suitable for microblogs because the natural language processing tools perform poorly on microblog texts due to their inherent characteristics. Studies show that one of the state-of-the-art part-of-speech taggers - OpenNLP only achieves the accuracy of 74% on tweets (Liu et al. 2011). The syntactic analysis tool that generates dependency relation may perform even worse. Besides, microblog messages may express opinion in different ways and do not always contain opinion words, which lowers the performance of methods utilizing opinion words to find opinion targets.

In this study, we propose an unsupervised method to collectively extract the opinion targets from opinionated sentences in the same topic.

* Xiaojun Wan is the corresponding author.
[1] https://twitter.com
[2] http://weibo.com/
[3] http://t.qq.com/

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1840–1850,
Seattle, Washington, USA, 18-21 October 2013.

Topics are directly identified by hashtags. We first present a dynamic programming based segmentation algorithm for Chinese hashtag segmentation. By leveraging the contents in a topic, our segmentation algorithm can successfully identify out-of-vocabulary words and achieve promising results. Afterwards, all the noun phrases in each sentence and the hashtag segments are extracted as opinion target candidates. We propose an unsupervised label propagation algorithm to collectively rank the candidates of all sentences based on the assumption that similar sentences in a topic may share the same opinion targets. Finally, for each sentence, the candidate which gets the highest score after unsupervised label propagation is selected as the opinion target.

Our contributions in this study are summarized as follows: 1) our method considers not only the explicit opinion targets within the sentence but also the implicit opinion targets in the hashtag or mentioned in the previous sentence. 2) We develop an efficient algorithm to segment Chinese hashtags. It can successfully identify out-of-vocabulary words by leveraging contextual information and help to improve the segmentation performance of the messages in the topic. 3) We develop an unsupervised label propagation algorithm for collective opinion target extraction. Label propagation (Zhu and Ghahramani, 2002) aims to spread label distributions from a small training set throughout the graph. However, our unsupervised algorithm leverages the connection between two adjacent unlabeled nodes to find the correct labels for both of them. The proposed unsupervised method does not need any training corpus which will cost much human labor especially for fine-grained annotation. 4) To the best of our knowledge, the task of opinion target extraction in microblogs has not been well studied yet. It is more challenging than microblog sentiment classification and opinion target extraction in review texts.

2 Characteristics of Chinese Microblogs

Most of previous microblog sentiment analysis researches focus on Twitter and especially in English. However, the analysis of Chinese microblogs has some differences with that of Twitter: 1) Chinese word segmentation is a necessary step for Chinese sentiment analysis, but the existing segmentation tool performs poorly on microblogs because the microblog texts are much different from regular texts. 2) Wang et al. (2011) find that hashtags in English tweets are used to highlight the sentiment information such as “ #love”, “#sucks” or serve as user-annotated coarse topics such as “#news”, “#sports”. But in Chinese microblogs, most of the hashtags are used to indicate fine-grained topics such as #NBA 总决赛第七场# (#NBAFinalG7#). Besides, hashtags in Twitter always appear within a sentence such as “I love #BarackObama!” while hashtags in Chinese microblogs are always isolated and are surrounded by two # symbols such as “#巴拉克奥巴马# 我爱他!” (“#BarackObama# I love him！”).

It is noteworthy that topics aggregated by the same hashtag play an important role in Chinese microblog websites. These websites often provide an individual webpage[4] to list hot topics and invite people to participate in the discussion, where each topic consists of tens of thousands of messages with the same hashtag. The hot topics have a wide coverage of timely events and entities. Analyzing the opinion targets of these topics can help to get a deeper overview of the public attitudes towards the entities involved in the hot topics.

3 Motivation

As described above, #hashtags# in Chinese microblogs often indicate fine-grained topics. In this study, we aim to collectively extract the opinion targets of messages with the same hashtag, i.e. in the same topic. Opinion target of a sentence can be divided into two types, one of which called *explicit target* appears in the sentence such as “I love *Obama*”, and the other one called *implicit target*

Topic	Sentence
#官员财产公示# #Property publicity of government offic-ials#	1. 纯属作秀！ (Just for show！)
	2. 财产公示在中国就是作秀。 (Property publicity is just a show in China.)
#菲军舰恶意撞击# #Philippine navy vessel hits Chinese fishing boat#	1. 政府还是不够强硬。 (The government is not tough enough.)
	2. 政府为何不能强硬一些？ (Why cannot the government take a tougher line?)

Table 1. Motivation Examples

[4] http://huati.weibo.com/

may appear out of the sentence, for example, the sentence "Just for show!" in Table 1 directly comments on the target in the hashtag "#Property publicity of government officials#" . Such implicit opinion targets are not considered in previous works and are more difficult to extract than explicit targets. However, we believe that the contextual information will help to locate both of the two kinds of opinion targets because similar sentences in a topic may share the same opinion target, which provides the possibility for collective extraction.

Table 1 shows the motivation examples of two topics and four sentences. The two sentences in each topic are considered to be similar because they share several Chinese words. In the topic #官员财产公示# (#Property publicity of government officials#), the first sentence omits the opinion target. However, the second one contains an explicit target "财产公示" ("property publicity") in the sentence. If we find the correct opinion target for sentence 2, we can infer that sentence 1 may have an implicit opinion target similar to the opinion target in sentence 2. In the second topic, both sentences contain a noun word "政府" ("government"). The similarity between these two sentences may indicate that both of the two sentences are expressing opinion on "政府".

Based on the above observation, we can assume that similar sentences in a topic may have the same opinion targets. Such assumption can help to locate both explicit and implicit opinion targets. Following this idea, we firstly extract all the noun phrases in each sentence as opinion target candidates after applying Chinese word segmentation and part-of-speech tagging. Afterwards, an unsupervised label propagation algorithm is proposed to rank these candidates for all sentences in the topic.

In our methods, hashtags are used to find gold-standard topics. For messages without hashtags, an alternative way is to generate pseudo topics by clustering microblogs messages and then apply the proposed algorithm to each pseudo topic. The detailed discussion of such general circumstance is shown in Section 5.7.

4 Methodology

4.1 Context-Aware Hashtag Segmentation

In our approach, the Chinese word segmentations of hashtags and topic contents are treated separately. Existing Chinese word segmentation tools work poorly on microblog texts. The segmentation errors especially on opinion target words will directly influence the results of part-of-speech tagging and candidate extraction. However, some of the opinion target words in a topic are often included in the hashtag. By finding the correct segments of a hashtag and adding them to the user dictionary of the Chinese word segmentation tool, we can remarkably improve the overall segmentation performance.

The following example can help to understand the idea better. In the topic #90 后打老人# (means "A young man hits an old man"), "90 后" (literally "90 later" and means a young man born in the 90s) is an important word because it is the opinion target of many sentences. However, existing Chinese word segmentation tools will regard it as two separate words "90" and "后" ("later"). Then in the part-of-speech tagging stage, "90" will be tagged as number and "后" will be tagged as localizer. As we only extract noun phrases as opinion target candidates, the wrong segmentation on "90 后" makes it impossible to find the right opinion target. Such error may occur many times in sentences that mention the word "90 后" and express opinion on it. In our method, the message texts of the topic are utilized to identify such out-of-vocabulary words based on its frequency in the topic. For example, the high frequency of "90 后" is a strong indication that it should be regard as a single word. After segmenting the hashtag correctly into "90 后/打/老人", we can add the hashtag segments to the user dictionary of the segmentation tool to further segment the message texts of the topic.

The basic idea for our hashtag segmentation algorithm is to regard strings that appear frequently in a topic as words. Formally, given a hashtag h that contains n Chinese characters $c_1c_2...c_n$. We want to segment into several words $w_1w_2...w_m$, where each word is formed by one of more characters.

Firstly, we define the stickiness score for a Chinese string $c_1c_2...c_n$ based on the Symmetrical Conditional Probability (SCP) (Silva and Lopes, 1999):

$$SCP(c_1c_2...c_n) = \frac{\Pr(c_1c_2...c_n)^2}{\frac{1}{n-1}\sum_{i=1}^{n}\Pr(c_1...c_i)\Pr(c_{i+1}...c_n)} \quad (1)$$

and $SCP(c_1) = \Pr(c_1)^2$ for string with only one character. $\Pr(c_1c_2...c_n)$ is the occurrence frequency of the string in the topic.

Following (Li et al., 2012b), we smooth the SCP value by taking logarithm calculation. Besides, the length of the string is taken into consideration,

$$SCP'(c_1c_2...c_n) = n \times \log SCP(c_1c_2...c_n) \quad (2)$$

where n is the number of characters in the string.

Then the stickiness score is defined by the sigmoid function as follows:

$$Stickiness(c_1c_2...c_n) = \frac{2}{1+e^{-SCP'(c_1c_2...c_n)}} \quad (3)$$

For the hashtag $h = c_1c_2...c_n$, we want to segment it into m words $w_1w_2...w_m$ which maximize the following equation,

$$\max \sum_{i=1}^{m} Stickness(w_i) \quad (4)$$

The optimization of Equation (4) can be solved efficiently by dynamic programming which iteratively segments a string into two substrings. Different from (Li et al., 2012b) which calculates the SCP value of each string based on Microsoft Web N-Gram, our hashtag segmentation algorithm only uses the topic content and do not need any additional corpus.

4.2 Candidate Extraction

After segmenting the hashtag, all the hashtag segments with length greater than one are added to the user dictionary of the Chinese word segmentation tool ICTCLAS[5] to further segment the message texts of the topic. It also assigns the part-of-speech tag for each word after segmentation. The noun phrases in each sentence is extracted by the following regular expression: $(noun|adj)(noun|adj|的)*noun$. That means a noun phrase can only include nouns, adjectives and the Chinese word "的" ("of"). It should begin with a noun or adjective and end with a noun. For example, in the following sentence, "中国/n 的/u 教育/n 制度/n 有/v 问题/n 。/w" ("Chinese education system has problems."), "中国的教育制度" ("Chinese education system") and "问题" ("problem") are extracted as noun phrases.

The character number of a noun phrase is limited between two and seven Chinese characters. For each sentence, all phrases that match the regular expression and meet the length restriction are extracted as explicit opinion target candidates. The hashtag segments are regarded as implicit candidates for all sentences. Besides, some opinionated sentences in microblogs do not contain any noun phase, such as "无聊至极！" ("*So boring*!"). These sentences may express opinion on object that has been mentioned before. Therefore, the explicit candidates of the previous sentence in the same message are also taken as the implicit candidates for such sentences.

We do not use any syntactic parsing tool to extract noun phrases because the parsing results on microblogs are not reliable. A performance comparison of our rule based method and the state-of-the-art syntactic parser will be shown in Section 5.

4.3 Unsupervised Label Propagation for Candidate Ranking

We simply assume that each opinionated sentence has one opinion target, which is consistent with

Algorithm 1 Unsupervised Label Propagation

Input:

Graph: $G = < V, E, \tilde{W} >$

Candidate Similarity: $S \in R_+^{M \times M}$

Prior labeling: $Y_v \in R_+^{1 \times M}$ for $v \in V$

Filtering Matrix: $F_v \in R_+^{M \times M}$ for $v \in V$

Probability: p^{inj} and p^{cont}

Output:

Label vector: $\hat{Y}_v \in R_+^{1 \times M}$

1: **for all** $v \in V$ **do**
2: $\hat{Y}_v \leftarrow Y_v$
3: **end for**
4: **repeat**
5: **for all** $v \in V$ **do**
6: $D_v \leftarrow \sum_{u \in V, u \neq v} \tilde{W}_{uv} \left(\hat{Y}_u \times S \right) \times F_v$
7: $\hat{Y}_v \leftarrow p^{inj} Y_v + p^{cont} D_v$
8: **end for**
9: **until convergence**

[5] http://www.ictclas.org/

the statistical result of our dataset that over 93% sentences have only one opinion target and each sentence has an average of 1.09 targets. Therefore, the most confident candidate of each sentence will be selected as the opinion target. In this section, we introduce an unsupervised graph-based label propagation algorithm to collectively rank the candidates of all sentences in a topic.

Label propagation (Zhu and Ghahramani, 2002; Talukdar and Crammer, 2009) is a semi-supervised algorithm which spreads label distributions from a small set of nodes seeded with some initial label information throughout the graph. The basic idea is to use information from the labeled nodes to label the adjacent nodes in the graph. However, our idea is to use the connection between different nodes to find the correct labels for all of them. Our unsupervised label propagation algorithm is summarized in Algorithm 1. Sentences are regarded as nodes and candidates of each sentence are regarded as labels. The label vector for each node is initialized based on the results of the candidate extraction step, which means no manually-labeled instances are needed in our model. In each iteration, the label vector of one node is propagated to the adjacent nodes. Both the sentence (node) similarity and the candidate (label) similarity are considered during propagation. Finally, we select the candidate with the highest score in the label vector as the opinion target for each sentence. The details of Algorithm 1 are presented as follows.

Formally, an undirected graph $G = < V, E, \tilde{W} >$ is built for each topic. A node $v \in V$ represents a sentence in the topic and an edge $e = (a, b) \in E$ indicates that the labels of the two vertices should be similar. $\tilde{W}$ is the normalized weight matrix to reflect the strength of this similarity. The similarity between two nodes W_{ab} is simply calculated by using the cosine measure (Salton et al., 1975) of the two sentences.

$$W_{ab} = cos(T_a, T_b) = \frac{T_a \cdot T_b}{\|T_a\| \cdot \|T_b\|} \quad (5)$$

where T_a and T_b are the term vectors of sentences a and b represented by the standard vector space model and weighted by term frequency. After calculating the similarity matrix W, we get the weight matrix $\tilde{W}$ by normalizing each row of W such that $\sum_b \tilde{W}_{ab} = 1$.

For each sentence (node) v, a candidate set C_v is extracted in the previous step. The candidate set CT for the whole topic is the union of all C_v,

$$CT = \bigcup C_v \quad (6)$$

The total number of candidates in the topic is denoted by $M = |CT|$. We calculate the candidate similarity matrix $S \in R_+^{M \times M}$ based on Jaccard Index:

$$S_{ij} = \frac{\left|A(CT_i) \cap A(CT_j)\right|}{\left|A(CT_i) \cup A(CT_j)\right|} \quad 1 \le i \ne j \le M \quad (7)$$

where $A(CT_i)$ and $A(CT_j)$ are the Chinese character sets of the i-th and j-th candidates in CT respectively.

Candidates are regarded as labels in our model and without loss of generality we assume that the possible labels for the whole topic are $L = \{1 \ldots M\}$ and each label in L corresponds to a unique candidate in CT. For each node $v \in V$, a label vector $Y_v \in R_+^{1 \times M}$ is initialized as

$$(Y_v)_k = \begin{cases} w & L_k \in C_v \\ 0 & L_k \notin C_v \end{cases} \quad 1 \le k \le M \quad (8)$$

where w is the initial weight of the candidate. We set $w = w_e$ if L_k is an explicit candidate (extracted noun phrase) of v and $w = w_i$ if L_k is an implicit candidate (hashtag segment or inherited from previous sentence) of v. If L_k is not a candidate of the current sentence, then the corresponding value in the label vector is 0. These values which are initialized as zero should always remain zero during the propagation algorithm because the corresponding label does not belong to the candidate set C_v of node v. To reset the values on these positions, a diagonal matrix $F_v \in R_+^{M \times M}$ is created for all nodes v,

$$(F_v)_{kk} = \begin{cases} 1 & (Y_v)_k > 0 \\ 0 & (Y_v)_k = 0 \end{cases} \quad 1 \le k \le M \quad (9)$$

where the subscript kk denotes the k-th position in the diagonal of matrix F_v. We can right-multiply Y_v by F_v to clear the values of the invalid candi-

dates. Figure 1 shows an example of creating the filtering matrix for a label vector.

$$Y_v = [1 \;\; 1 \;\; 0.5 \;\; 0] \;\rightarrow\; F_v = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 1. Example of filtering matrix

The propagation process is formalized via two possible actions: inject and continue, with predefined probabilities p^{inj} and p^{cont}. Their sum is unit: $p^{inj} + p^{cont} = 1$. In each iteration, every node is influenced by its adjacent nodes. The propagation influence for each node v is

$$D_v = \sum_{u \in V, u \neq v} \tilde{W}_{uv} \left(\hat{Y}_u \times S \right) \times F_v \qquad (10)$$

where $\hat{Y}_u$ is the label vector of node u at the previous iteration. By multiplying the candidate similarity matrix S, we aim to propagate the score of the i-th candidate of node u not only to the i-th candidate of node v, but also to all the other candidates. W_{uv} measures the strength of such propagation. The filtering matrix F_v is used to clear the values of the invalid candidates as described above.

Then the label vector of node v is updated as follow,

$$\hat{Y}_v = p^{inj} Y_v + p^{cont} D_v \qquad (11)$$

When the positions of the largest values in all label vectors keep unchanged in ten iterations, it is regarded that the algorithm has already converged.

5 Experiments

5.1 Dataset

We use the dataset from the 2012 Chinese Microblog Sentiment Analysis Evaluation (CMSAE)[6] held by China Computer Federation (CCF). There are three tasks in the evaluation: subjectivity classification, polarity classification and opinion target extraction. The dataset contains 20 topics collected from Tencent Weibo, a popular Chinese microblogging website. All the messages in a topic contain the same hashtag. The dataset has a total of 17518 messages and 31675 sentences. In each topic, 100 messages are manually annotated with subjectivity, polarity and opinion targets. A total of 2361opinion targets are annotated for 2152 opinionated sentences.

5.2 Evaluation Metric

Precision, recall and F-measure are used in the evaluation. Since expression boundaries are hard to define exactly in annotation guidelines (Wiebe et al., 2005), both the strict evaluation metric and the soft evaluation metric are used in CMSAE.

Strict Evaluation: For a proposed opinion target, it is regarded as correct only if it covers the same span with the annotation result. Note that, in CMSAE, an opinion target should be proposed along with its polarity. The correctness of the polarity is also necessary.

Soft Evaluation: The soft evaluation metric presented in (Johansson and Moschitti, 2010) is adopted by CMSAE. The *span coverage c* between each pair of the proposed target span *s* and the gold standard span *s'* is calculated as follows,

$$c(s, s') = \frac{|s \cap s'|}{|s'|} \qquad (12)$$

In Equation 12, the operator $|\cdot|$ counts Chinese characters, and the intersection $\cap$ gives the set of characters that two spans have in common.

Using the span coverage, the span set coverage C of a set of spans S with respect to another set S' is

$$C(S, S') = \sum_{s \in S} \sum_{s' \in S'} c(s, s') \qquad (13)$$

The soft precision P and recall R of a proposed set of spans $\hat{S}$ with respect to a gold standard set S is defined as follows:

$$\text{Precision} = \frac{C(\hat{S}, S)}{|\hat{S}|} \quad \text{Recall} = \frac{C(\hat{S}, S)}{|S|} \qquad (14)$$

Note that the operator $|\cdot|$ counts spans in Equation 14. The soft F-measure is the harmonic mean of soft precision and recall.

5.3 Comparison Methods

Our proposed approach is first compared with the CMSAE teams.

CMSAE Teams: Sixteen teams participated in the opinion target extraction task of CMSAE. The methods of the top 3 teams are used as baselines

[6] http://tcci.ccf.org.cn/conference/2012/pages/page04_eva.html. The dataset can also be publicly accessed on the website.

here. They are denoted as **Team-1**, **Team-2** and **Team-3** respectively. The average result of all the sixteen teams is also included and is denoted as **Team-Avg**. We will briefly introduce the best team's method. The most important component of their model is a topic-dependent opinion target lexicon which is called object sheet. If a word or phrase in the object sheet appears in a sentence or a hashtag, it is extracted as opinion target. The object sheet is manually built for each topic, which means their method cannot be applied to new topics.

The following models are also used for comparison.

AssocMi: We implement the unsupervised method for opinion target extraction based on (Hu and Liu, 2004), which relies on association mining and a sentiment lexicon to extract frequent and infrequent product features.

CRF: The CRF-based method used in (Jakob and Gurevych, 2010) is also used for comparison. We implement both the single-domain and cross-domain models. Both models are evaluated using 5-fold cross-validation. More specifically, the single-domain model, denoted as **CRF-S**, trains different models for different topics. In each cross-validation round, 80 percent of each topic is used for training and the other 20 percent is used for test. The cross-domain model, denoted as **CRF-C**, uses 16 topics for training and the rest 4 topics for test in each round.

5.4 Comparison Results

CMSAE requires all the teams to perform the subjectivity and polarity classification task in advance. The opinion targets are extracted only for opinionated sentences and should be proposed along with their polarity. To make a fair comparison, we directly use the subjectivity and polarity classification results of Team-1. Then our unsupervised label propagation (**ULP**) method is used to extract the opinion targets for the proposed opinionated sentences. The parameters of our method are simply set as $p^{inj} = p^{cont} = 0.5$, $w_e = 1$ and $w_i = 0.5$.

Table 2 lists the comparison results with CMSAE teams. The average F-measure of all teams is 0.12 and 0.20 in strict and soft evaluation, respectively. It shows that opinion target extraction is a quite hard problem in Chinese microblogs. Our method performs better than all the teams. It increases by 10% and 13% in the two kinds of F-measure compared to the best team. Besides, we do not need any prior information of the topics while Team-1 has to manually build an opinion target lexicon for each topic.

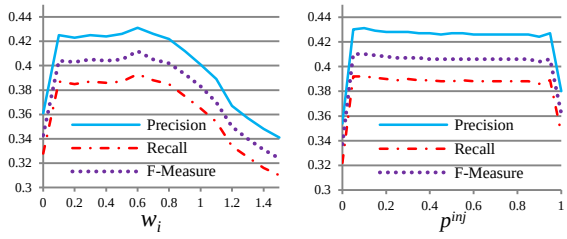
To compare with the other opinion target extraction methods, we only use gold-standard opinionated sentences for evaluation and do not classify the polarity of the opinion targets. Table 3 shows the experimental results of the four models. Our approach achieves the best result among them. AssocMi performs worst in strict evaluation but gets better results than the two CRF-based models in soft evaluation. The two CRF-based models achieve high precision but low recall. We can also observe that CRF-S is much more effective than CRF-C. It achieves high results because it has already seen the opinion targets in the training set. However, it is impossible to build such single-domain model in practical applications because

Method.	Strict			Soft		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Team-Avg	0.17	0.09	0.12	0.29	0.15	0.20
Team-3	0.26	0.16	0.20	0.40	0.25	0.31
Team-2	0.31	0.18	0.23	0.40	0.22	0.29
Team-1	0.30	**0.27**	0.29	0.39	0.36	0.37
ULP	**0.37**	**0.27**	**0.32**	**0.48**	**0.37**	**0.42**

Table 2. Comparison results with CMSAE teams (with subjectivity and polarity classification in advance)

Method	Strict			Soft		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
AssocMi	0.22	0.20	0.21	0.47	0.43	0.45
CRF-C	0.59	0.15	0.24	0.70	0.18	0.28
CRF-S	**0.61**	0.27	0.35	**0.73**	0.31	0.41
ULP	0.43	**0.39**	**0.41**	0.61	**0.55**	**0.58**

Table 3. Comparison results with baseline methods (only gold-standard opinionated sentences are used)



(a) Initial Candidate Weight (b) Injection Probability

Figure 2. Influence of the parameters

labeled instances are not available for new topics. Our proposed method does not require any training data and gets an increase of 17% over CRF-S and 70% over CRF-C in strict evaluation. In terms of soft evaluation, we achieve an increase of 41% and 107% over the two CRF models.

5.5 Parameter Sensitivity Study

In this section, we study the parameter sensitivity. There are two major parameters in our algorithm: the initial weight w for both explicit and implicit candidates in Equation 8 and the injection probability p^{inj} in Equation 11.

The initial weights of explicit and implicit candidates are set differently because the explicit candidates are more likely to be the opinion targets. These two kinds of initial weights are denoted as w_e and w_i for explicit and implicit candidate, respectively. To study the impact of the initial weights, we fix w_e at 1 and tune w_i because we only care about the relative contribution of them. The injection probability is fixed at 0.5. Figure 2(a) displays the opinion target extraction performance when w_i varies from 0 to 1.5. Due to limited space, we only list the strict F-measure of opinion target extraction evaluated on opinioned sentences (same experimental setup as Table 3).

In particular, when w_i is equal to 0, only explicit candidates are considered. When w_i becomes larger than 1, the implicit candidates become more important than explicit candidates. From the curve in Figure 2(a), we can observe that the implicit candidates help to improve the performance significantly when w_i varies from 0 to 0.1. The performance reaches the peak when w_i = 0.7 and declines rapidly when w_i gets larger than 1.

To study the impact of injection probability p^{inj}, we fix the initial weights for explicit and implicit candidates as 1 and 0.5, respectively. Figure 2(b) shows the results of opinion target extraction with respect to different values of the injection probability. We can observe that the performance keeps steady except for the two extreme values 0 and 1. From the above two figures, we can conclude that our proposed method performs well and robustly with a wide range of parameter values.

Method	Total	Correct	F-Measure of Opinion Target Extraction	
			Strict	Soft
Berkley Parser	4554	877	0.36	0.56
Rule	4105	918	0.37	0.56
HS + Rule	4094	1042	0.41	0.58

Table 4. Performance of candidate extraction and opinion target extraction

5.6 Analysis of Candidate Extraction

Candidate extraction is an important step in our proposed method. If the correct opinion target is not extracted as a candidate, the ranking step will be in vain. As described in Section 3, we develop a hashtag segmentation algorithm and use a rule based method to extract noun phrases from each sentence. We do not use any parsing tool because we believe the performance of these tools is not good enough when applied on microblogs. A quantitative comparison is shown in this section.

We use one of the state-of-the-art syntactic analysis tools - **Berkeley Parser** (Petrov et al., 2006) for comparison here. Noun phrases are directly extracted from the parsing results. Our method **HS+Rule** leverages the hashtag segments to enhance the segmentation result and extracts explicit candidate using a regular expression. To demonstrate the effectiveness of our hashtag segmentation algorithm, the second comparison baseline **Rule** directly uses ICTCLAS to segment the whole topic content and labels each word with its part-of-speech tag. The explicit candidates are extracted by using the same regular expression.

The performance on candidate extraction is compared in Table 4. The second column shows the number of all extracted candidates for all the opinionated sentences by different methods. The third column shows the number of correct opinion targets among them. We can find that the two rule-based models both outperform Berkeley Parser and our **HS+Rule** method finds 14% more correct opinion targets than **Rule**. It proves the effectiveness of our hashtag segmentation algorithm. The

total number of candidates extracted by **HS+Rule** is also less than the other two methods. Therefore, the performance of label propagation will be improved when there are fewer candidates to rank. It can be demonstrated by the F-measure of opinion target extraction in the fourth and fifth columns. The experiments are conducted on opinionated sentence only as above. By using **HS+Rule** to extract candidates, our label propagation algorithm gets the highest F-measure in both evaluation metrics.

5.7 Performance on Pseudo Topics by Message Clustering

In our collective extraction algorithm, topics are directly identified by hashtags. For messages without hashtags, we can first employ clustering algorithms to obtain pseudo topics (clusters) and then exploiting the topic-oriented algorithm for collective opinion target extraction. To test the performance of the proposed method in such circumstance, we use the popular clustering algorithm - Affinity Propagation (Frey and Dueck, 2007) to generate topics. The experimental results are shown in Table 5. **APCluster** means that the messages are clustered after removing all the hashtags. **APCluster+HS** means that all the hashtags are retained as normal texts for calculating message similarity. Therefore, the clustering performance can be largely improved. The standard cosine similarity is used to measure the distance between microblog messages for Affinity Propagation in the above two methods. The last method denoted as **GoldCluster** directly uses hashtags to identify the gold-standard topics which shows the upper bound of the performance. After clustering microblogs, the opinion targets of messages in each cluster are collectively extracted by the proposed unsupervised label propagation algorithm. The experiments are conducted on opinionated sentences only.

From the results, we can see that clustering microblogs without hashtags is a quite difficult job which only gets an F-Measure of 0.27. However, the corresponding opinion target extraction performance is still promising, which outperforms the AssocMi and CRF-C methods in Table 3. With the help of hashtags, the clustering performance of **APCluster+HS** is largely improved and the opinion target extraction performance is also increased.

Clustering Method	F-Measure of Clustering	F-Measure of Opinion Target Extraction	
		Strict	Soft
APCluster	0.27	0.35	0.50
APCluster+HS	0.71	0.37	0.55
GoldCluster	1.00	0.41	0.58

Table 5. Performance of clustering and opinion target extraction

It outperforms all the baseline methods in Table 3. The above results reveal that our proposed unsupervised label propagation algorithm works well in pseudo topics and the performance can be increased with better clustering results. Therefore, we can try to incorporate other social network information to improve the message clustering performance, which will be studied in our future work.

6 Related Work

Sentiment analysis, a.k.a. opinion mining, is the field of studying and analyzing people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions (Liu, 2012). Most of the previous sentiment analysis researches focus on customer reviews (Pang et al., 2002; Hu and Liu, 2004) and some of them focus on news (Kim and Hovy, 2006) and blogs (Draya et al., 2009). However, sentiment analysis on microblogs has recently attracted much attention and has been proved to be very useful in many applications.

Classification of opinion polarity is the most common task studied in microblogs. Go et.al (2009) follow the supervised machine learning approach of Pang et al. (2002) to classify the polarity of each tweet by distant supervision. The training dataset of their method is not manually labeled but automatically collected using the emoticons. Barbosa and Feng (2010) use the similar pseudo training data collected from three online websites which provide Twitter sentiment analysis services. Speriosu et al. (2009) explore the possibility of exploiting the Twitter follower graph to improve polarity classification.

Opinion target extraction is a fine-grained word-level task of sentiment analysis. Currently, this task has not been well studied in microblogs yet. It is mostly performed on product reviews where opinion targets are always described as product features or aspects. The pioneering research on this task is conducted by Hu and Liu

(2004) who propose a method which extracts frequent nouns and noun phrases as the opinion targets. Jakob and Gurevych (2010) model the problem as a sequence labeling task based on Conditional Random Fields (CRF). Qiu et al. (2011) propose a double propagation method to extract opinion word and opinion target simultaneously. Liu et al. (2012) use the word translation model in a monolingual scenario to mine the associations between opinion targets and opinion words.

7 Conclusion and Future Work

In this paper, we study the problem of opinion target extraction in Chinese microblogs which has not been well investigated yet. We propose an unsupervised label propagation algorithm to collectively rank the opinion target candidates of all sentences in a topic. We also propose a dynamic programming based algorithm for segmenting Chinese hashtags. Experimental results show the effectiveness of our method.

In future work, we will try to collect and annotate data for microblogs in other languages to test the robustness of our method. The repost and reply messages can also be integrated into our graph model to help improve the results.

Acknowledgments

The work was supported by NSFC (61170166), Beijing Nova Program (2008B03) and National High-Tech R&D Program (2012AA011101).

References

Barbosa Luciano and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters. Association for Computational Linguistics, 2010.

Johan Bollen, Huina Mao and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. Journal of Computational Science 2.1 (2011): 1-8.

Gérard Dray, Michel Planti é, Ali Harb, Pascal Poncelet, Mathieu Roche and Fran çois Trousset. 2009. Opinion Mining from Blogs. In International Journal of Computer Informa-tion Systems and Industrial Management Applications.

Brendan J. Frey and Delbert Dueck. 2007. "Clustering by passing messages between data points." Science 315.5814 (2007): 972-976.

Alec Go, Richa Bhayani and Lei Huang. 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford (2009): 1-12.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. 2004. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 168-177. ACM.

Long Jiang , Mo Yu, Ming Zhou, Xiaohua Liu and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 151-160.

Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. 2010. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. Proceedings of the Fourteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics.

Soo-Min Kim and Eduard Hovy. 2006. Extracting Opinions, Opinion Holders and Topics Expressed in Online News Media Text. In Proceedings of the ACL Workshop on Sentiment and Subjectivity in Text, 2006, pp. 1–8.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang and Xiaoyan Zhu. 2012a. Cross-Domain Co-Extraction of Sentiment and Topic Lexicons. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 410–419, Jeju, Republic of Korea, 8-14 July 2012.

Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun and Bu-Sung Lee. 2012b. Twiner: Named entity recognition in targeted twitter stream. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, pp. 721-730. ACM.

Xiaohua Liu, Kuan Li, Ming Zhou and Zhongyang Xiong. 2011. Collective semantic role labeling for tweets with clustering. In Proceedings of the Twenty-Second international joint conference on Artificial

Intelligence-Volume Volume Three, pp. 1832-1837. AAAI Press.

Bing Liu. 2012. Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies 5.1 (2012): 1-167.

Kang Liu, Liheng Xu and Jun Zhao. 2012. Opinion Target Extraction Using Word-Based Translation Model. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pp. 79-86. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. Learning accurate, compact, and interpretable tree annotation. 2006. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 433-440.

Guang Qiu, Bing Liu, Jiajun Bu and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. Computational linguistics 37, no. 1 (2011): 9-27.

G. Salton, A. Wong and C. S. Yang. 1975. A Vector Space Model for Automatic Indexing, Communications of the ACM, vol. 18, nr. 11, pages 613–620.

J. F. da Silva and G. P. Lopes. 1999. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In Proc. of the 6th Meeting on Mathematics of Language .

Michael Speriosu, Nikita Sudan, Sid Upadhyay and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In Proceedings of the First Workshop on Unsupervised Learning in NLP, pp. 53-63. Association for Computational Linguistics, 2011.

Partha Talukdar and Koby Crammer. New regularized algorithms for transductive learning. 2009. Machine Learning and Knowledge Discovery in Databases (2009): 442-457.

Jie Tang, Yuan Zhang, Jimeng Sun, Jinhai Rao, Wenjing Yu, Yiran Chen and A. C. M. Fong. 2012. Quantitative study of individual emotional states in social networks. Affective Computing, IEEE Transactions on 3, no. 2 (2012): 132-144.

Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner and Isabell M. Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In Proceedings of the fourth international aaai conference on weblogs and social media, pp. 178-185.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, CMU CALD tech report.

Li Zhuang, Feng Jing and Xiaoyan Zhu. 2006. Movie review mining and summarization. In Proceedings of the ACM 15th Conference on Information and Knowledge Management, pages 43–50, Arlington, Virginia, USA, November.

Detecting Promotional Content in Wikipedia

Shruti Bhosale **Heath Vinicombe** **Raymond J. Mooney**
Department of Computer Science
The University of Texas at Austin
{shruti,vini,mooney}@cs.utexas.edu

Abstract

This paper presents an approach for detecting promotional content in Wikipedia. By incorporating stylometric features, including features based on n-gram and PCFG language models, we demonstrate improved accuracy at identifying promotional articles, compared to using only lexical information and meta-features.

1 Introduction

Wikipedia is a free, collaboratively edited encyclopedia. Since normally anyone can create and edit pages, some articles are written in a promotional tone, violating Wikipedia's policy requiring a neutral viewpoint. Currently, such articles are identified manually and tagged with an appropriate Cleanup message[1] by Wikipedia editors. Given the scale and rate of growth of Wikipedia, it is infeasible to manually identify all such articles. Hence, we present an approach to automatically detect promotional articles.

Related work in quality flaw detection in Wikipedia (Anderka et al., 2012) has relied on meta-features based on edit history, Wikipedia links, structural features and counts of words, sentences and paragraphs. However, we hypothesize that there are subtle differences in the linguistic style that distinguish promotional tone, which we attempt to capture using stylometric features, particularly deeper syntactic features. We model the style of promotional and normal articles using language models based on both n-grams and Probabilistic Context Free Grammars (PCFGs). We show that using such stylometric features improves over using only shallow lexical and meta-features.

2 Related Work

Anderka et al. (2012) developed a general model for detecting ten of Wikipedia's most frequent quality flaws. One of these flaw types, "Advert"[2], refers to articles written like advertisements. Their classifiers were trained using a set of lexical, structural, network and edit-history related features of Wikipedia articles. However, they used no features capturing syntactic structure, at a level deeper than Part-Of-Speech (POS) tags.

A related area is that of vandalism detection in Wikipedia. Several systems have been developed to detect vandalizing edits in Wikipedia. These fall into two major categories: those analyzing author information and edit metadata (Wilkinson and Huberman, 2007; Stein and Hess, 2007); and those using NLP techniques such as n-gram language models and PCFGs (Wang and McKeown, 2010; Harpalani et al., 2011). We combine relevant features from both these categories to train a classifier that distinguishes promotional content from normal Wikipedia articles.

3 Dataset Collection

We extracted a set of about 13,000 articles from English Wikipedia's category, "Category:All arti-

[1] http://en.wikipedia.org/wiki/Wikipedia:Template_messages/Cleanup

[2] "Advert" is the flaw-type of majority of the articles in the Category 'Articles with a promotional tone'.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1851–1857,
Seattle, Washington, USA, 18-21 October 2013.

Content Features
Number of characters
Number of words
Number of sentences
Average Word Length
Average, Minimum, Maximum Sentence Lengths, Ratio of Maximum to minimum sentence lengths
Ratio of long sentences (>48 words) to Short Sentences (<33 words)
Percentage of Sentences in the passive voice
Relative Frequencies of POS tags for pronouns, conjunctions, prepositions, auxiliary verbs, modal verbs, adjectives and adverbs
Percentage of sentences beginning with a pronoun, article, conjunction, preposition, adjective, adverb
Percentage of special phrases[3] such as peacock terms ('legendary', 'acclaimed', 'world-class'), weasel terms ('many scholars state', 'it is believed/regarded', 'many are of the opinion', 'most feel', 'experts declare', 'it is often reported') , editorializing terms ('without a doubt', 'of course', 'essentially')
Percentage of easy words, difficult words (Dale-Chall List), long words and stop words
Overall Sentiment Score based on SentiWordNet[4]

Table 1: Content Features of a Wikipedia Article

cles with a promotional tone" as a set of positive examples. We extracted a set of 26,000 untagged articles to form a noisy set of negative examples, which may contain some promotional articles that have not yet been tagged by Wikipedia editors. To counter this noise, we repeated the experiment using Wikipedia's Featured Articles and Good Articles (approx. 11,000) as a set of clean negative examples. We used 70% of the articles in each category to train language models for each of the three categories (promotional articles, featured/good articles, untagged articles), and used the remaining 30% to evaluate classifier performance using 10-fold cross-validation.

4 Features

4.1 Content and Meta Features of an Article

We used the content and meta features proposed by Anderka et al. (2012) as given in Tables 1-4. We also

[3] http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch

[4] This feature is not included in Anderka et al. (2012)

Structural Features
Number of Sections
Number of Images
Number of Categories
Number of Wikipedia Templates used
Number of References, Number of References per sentence and Number of references per section

Table 2: Structural Features of a Wikipedia Article

Wikipedia Network Features
Number of Internal Wikilinks (to other Wikipedia pages)
Number of External Links (to other websites)
Number of Backlinks (i.e. Number of wikilinks from other Wikipedia articles to an article)
Number of Language Links (i.e. Number of links to the same article in other languages)

Table 3: Network Features of a Wikipedia Article

added a new feature, "Overall Sentiment Score" for an article. This feature is the average of the sentiment scores assigned by SentiWordnet (Baccianella et al., 2010) to all positive and negative sentiment bearing words in an article. In total, this results in 58 basic document features.

4.2 N-Gram Language Models

Language models are commonly used to measure stylistic differences in language usage between authors. For this work, we employed them to model the difference in style of neutral vs. promotional Wikipedia articles. We trained trigram word language models and trigram character language models[5] with Witten-Bell smoothing to produce probabilistic models of both classes.

4.3 PCFG Language Models

Probabilistic Context Free Grammars (PCFG) capture the syntactic structure of language by modeling sentence generation using probabilistic CFG productions. We hypothesize that sentences in promotional articles and those in neutral articles tend to have different kinds of syntactic structures and therefore, we explored the utility of PCFG models for detecting this difference. Since we do not have ground-truth parse trees for sentences in our dataset,

[5] Modeling longer character sequences did not help.

Features based on PCFG models and n-gram Language models
Difference in the probabilities assigned to an article by the positive and the negative class *character trigram language models* (LM_char_trigram)
Difference in the probabilities assigned to an article by the positive and the negative class *word trigram language models* (LM_word_trigram)
Difference in the *mean* values of the probabilities assigned to sentences of an article by the positive and negative class *PCFG models* (PCFG_mean)
Difference in the *maximum* values of the probabilities assigned to sentences of an article by the positive and negative class *PCFG models* (PCFG_max)
Difference in the *minimum* values of the probabilities assigned to sentences of an article by the positive and negative class *PCFG models* (PCFG_min)
Difference in the *standard deviation* values of the probabilities of sentences of an article by the positive and negative class *PCFG models* (PCFG_std_deviation)

Table 5: Features of a Wikipedia Article based on PCFG models and n-gram Language models

Edit History Features
Age of the article
Days since last revision of the article
Number of edits to the article
Number of unique editors
Number of edits made by registered users and by anonymous IP addresses
Number of edits per editor
Percentage of edits by top 5% of the top contributors to the article

Table 4: Edit-History Features of a Wikipedia Article

we followed the method of (Raghavan et al., 2010; Harpalani et al., 2011), which uses the output of the Stanford parser to train PCFG models for stylistic analysis. We used the PCFG implementation of Klein and Manning (2003) to learn a PCFG model for each category.

4.4 Classification

The n-gram and PCFG language models were used to create a set of additional document features. We used the probability assigned by the language models to each sentence in a test document to calculate document-wide statistics such as the mean, maximum, and minimum probability and standard deviation in probabilities of the set of sentences in an article. The language-modeling features used are shown in Table 5.

Since we have a wide variety of features, we experimented with various ensemble learning techniques and found that LogitBoost performed best empirically. We used the Weka implementation of LogitBoost (Friedman et al., 2000) to train a classifier using various combinations of features. We used Decision Stumps as a base classifier and ran boosting for 500 iterations.

5 Experimental Evaluation

5.1 Methodology

We used 10-fold cross-validation to test the performance of our classifier using various combinations of features. We ran the classifier on the portion (30%) of the dataset not used for language modeling.[6] We measured overall classification accuracy as well as precision, recall, F-measure, and area under the ROC curve for all experiments. We tested performance in two settings (Anderka et al., 2012):

- *Pessimistic Setting*: The negative class consists of articles from the Untagged set. Since some of these could be manually undetected promotional articles, the accuracy measured in this setting is probably an under-estimate.
- *Optimistic Setting*: The negative class consists of articles from the Featured/Good set. These articles are at one end of the quality spectrum, making it relatively easier to distinguish them from promotional articles.

The true performance of the classifier is likely somewhere between that achieved in these two settings.

[6] We maintain an equal number of positive and negative test cases in both the settings.

Features	Pessimistic Setting				Optimistic Setting			
	P	R	F1	AUC	P	R	F1	AUC
Bag-of-words Baseline	0.823	0.820	0.821	0.893	0.931	0.931	0.931	0.979
PCFG	0.881	0.870	0.865	0.903	0.910	0.910	0.910	0.961
Character trigrams	0.889	0.887	0.888	0.952	0.858	0.843	0.841	0.877
Word trigrams	0.863	0.863	0.863	0.931	0.887	0.883	0.882	0.931
Character trigrams + Word trigrams	0.89	0.888	0.889	0.952	0.908	0.907	0.907	0.962
PCFG+Char. trigrams+Word trigrams	0.914	0.915	0.914	0.974	0.950	0.950	0.950	0.983
58 Content and Meta Features	0.866	0.867	0.867	0.938	0.986	0.986	0.986	0.996
All Features	**0.940**	**0.940**	**0.940**	**0.986**	**0.989**	**0.989**	**0.989**	**0.997**

Table 6: Performance (Precision(P), Recall(R), F1 score, AUC) of the classifier in the two settings

5.2 Results for Pessimistic Setting

From Table 6, we see that all features perform better than the bag-of-words baseline. We also see that character trigrams, one of the simplest features, gives the best individual performance. However, deeper syntactic features using PCFGs also performs quite well. Combining all of the language-modeling features (PCFG + character trigrams + Word trigrams) further improves performance. Compared to the 58 content and meta features utilized by Anderka et al., (2012) described in Section 4.1, the PCFG and character trigram features give much better performance, both individually and when combined. It is interesting to note that adding Anderka et al.'s features to the language-modeling ones gives a fairly small improvement in performance. This validates our hypothesis that promotional articles tend to have a distinct linguistic style that is captured well using language models.

5.3 Results for Optimistic Setting

In the Optimistic Setting, as shown in Table 6, the content and meta features give the best performance, which improves only slightly when combined with language-modeling features. The bag-of-words baseline performs better than all the language modeling features. This performance could be because there is a much clearer distinction between promotional articles and featured/good articles that can be captured by simple features alone. For example, featured/good articles are generally longer than usual Wikipedia articles and have more references.

5.4 Top Ranked Features and their Performance

To analyze the performance of different features, we determined the top ranked features using Information Gain. In the Pessimistic Setting, the top six features are all language-modeling features (character trigram model feature works best), followed by basic meta-features such as character count, word count, category count and sentence count. The new feature we introduced, "Overall Sentiment Score" is the 18th most informative feature in the pessimistic setting, indicating that the cumulative sentiment of a bag of words is not as discriminative as we would intuitively assume. Using the 10 top-ranked features, we get an F1 of 0.93, which is only slightly worse than that achieved using all features (F1 = 0.94).

In the Optimistic Setting, the top-ranked features are the number of references and the number of references per section. This is consistent with the observation that featured/good articles have very long and comprehensive lists of references, since Wikipedia's fundamental policy is to maintain verifiability by citing relevant sources. Features based on the n-gram and PCFG models also appear in the list of ten best features. Using only the top 10 features, gives an F1 of 0.988, which is almost as good as using all features (F1 = 0.989).

5.5 Optimistic and Pessimistic Settings

In the optimistic setting, there is a clear distinction between the positive (promotional) and negative (featured/good) classes. But there are only subtle differences between the positive and negative (untagged articles) classes in the pessimistic setting.

Best Features in Pessimistic Setting	Best Features in Optimistic Setting
LM_char_trigram	Number of References
LM_word_trigram	Number of References per Section
PCFG_min	LM_word_trigram
PCFG_max	Number of Words
PCFG_mean	PCFG_mean
PCFG_std_deviation	Number of Sentences
Number of Characters	LM_char_trigram
Number of Words	Number of Words
Number of Categories	Number of Characters
Number of Sentences	Number of Backlinks

Table 7: Top 10 Features (listed in order) in both Settings ranked using Information Gain

These two classes are superficially similar, in terms of length, reference count, section count etc. Stylometric features based on the trained language models are successful at detecting the subtle linguistic differences in the two types of articles. This is useful because the pessimistic setting is closer to the real-world setting of articles in Wikipedia.

5.6 Error Analysis

Since the pessimistic setting is close to the real setting of Wikipedia articles, it is useful to do an error analysis of the classifier's performance in this setting. There is an approximately equal proportion of false positives and false negatives.

A significant number of false positives seem to be cases of manually undetected promotional articles. This demonstrates the practical utility of our classifier. But there are also many false positives that seem to be truly unbiased. These articles appear to have been poorly written, without following Wikipedia's editing policies. Examples include use of very long lists of nouns, use of ambiguous terms like "many believe" and excessive use of superlatives. Other common characteristics of most of the false positives are presence of a considerable number of complex sentences with multiple subordinate clauses. These stylistic cues seem to be misleading the classifier.

A common thread underlying most of the false negatives is the fact that they are written in a narrative style or they have excessive details in terms of the content. Examples include narrating a detailed story of a fictional character in an unbiased manner or writing a minutely detailed account of the history of an organization. Another source of false negatives comes from biographical Wikipedia pages which are written in a resume style, listing all their qualifications and achievements. These cues could help one manually detect that the article, though not promotional in style, is probably written with the view of promoting the entity. As possible future work, we could incorporate features derived from language models for narrative style trained using an appropriate external corpus of narrative text. This might enable the classifier to detect some cases of unbiased promotional articles.

6 Conclusion

Our experiments and analysis show that stylometric features based on n-gram language models and deeper syntactic PCFG models work very well for detecting promotional articles in Wikipedia. After analyzing the errors that are made during classification, we realize that though promotional content is non-neutral in majority of the cases, there do exist promotional articles that are neutral in style. Adding additional features based on language models of narrative style could lead to a better model of Wikipedia's promotional content.

7 Acknowledgements

This research was supported in part by the DARPA DEFT program under AFRL grant FA8750-13-2-0026 and by MURI ARO grant W911NF-08-1-0242. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the view of DARPA, AFRL, ARO, or the US government.

References

Maik Anderka, Benno Stein, and Nedim Lipka. 2012. Predicting quality flaws in user-generated content: the case of Wikipedia. In *Proceedings of the 35th International ACM SIGIR Conference on Research and development in Information Retrieval*, SIGIR '12, pages 981–990, New York, NY, USA. ACM.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, May*.

Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. 2003. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1-2):109–123.

Hugo J Escalante, Thamar Solorio, and M Montes-y Gómez. 2011. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 288–298.

Rudolf Flesch. 1948. A new readability yardstick. *The Journal of Applied Psychology*, 32(3):221.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407.

Michael Gamon. 2004. Linguistic correlates of style: Authorship classification with deep linguistic analysis features. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 611. Association for Computational Linguistics.

Manoj Harpalani, Michael Hart, Sandesh Singh, Rob Johnson, and Yejin Choi. 2011. Language of vandalism: Improving Wikipedia vandalism detection via stylometric analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*, volume 2, pages 83–88.

Daniel Hasan Dalip, Marcos André Gonçalves, Marco Cristo, and Pável Calado. 2009. Automatic quality assessment of content created collaboratively by web communities: a case study of Wikipedia. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital libraries*, JCDL '09, pages 295–304, New York, NY, USA. ACM.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79. Association for Computational Linguistics.

Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. 2003. N-gram-based author profiles for authorship attribution. In *Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING*, volume 3, pages 255–264.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Emily Pitler, Annie Louis, and Ani Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proceedings of the 48th annual meeting of the Association for Computational Linguistics*, pages 544–554. Association for Computational Linguistics.

Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. 2010. Authorship attribution using probabilistic context-free grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 38–42, Stroudsburg, PA, USA. Association for Computational Linguistics.

Congzhou He Ramyaa and Khaled Rasheed. 2004. Using machine learning techniques for stylometry. In *Proceedings of International Conference on Machine Learning*.

Paul Rayson, Andrew Wilson, and Geoffrey Leech. 2001. Grammatical word class variation within the british national corpus sampler. *Language and Computers*, 36(1):295–306.

Klaus Stein and Claudia Hess. 2007. Does it matter who contributes: a study on featured articles in the German Wikipedia. In *Proceedings of the Eighteenth Conference on Hypertext and Hypermedia*, pages 171–174. ACM.

Kristina Toutanova and Christopher D Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

William Yang Wang and Kathleen R. McKeown. 2010. "Got you!": Automatic vandalism detection in Wikipedia with web-based shallow syntactic-semantic modeling. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1146–1154, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dennis M Wilkinson and Bernardo A Huberman. 2007. Assessing the value of coooperation in Wikipedia. *arXiv preprint cs/0702140*.

Learning Topics and Positions from Debatepedia

Swapna Gottipati[†] **Minghui Qiu**[†] **Yanchuan Sim**[‡] **Jing Jiang**[†] **Noah A. Smith**[‡]
[†]School of Information Systems, Singapore Management University, Singapore
[‡]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[†]{swapnag.2010,minghui.qiu.2010,jingjiang}@smu.edu.sg
[‡]{ysim,nasmith}@cs.cmu.edu

Abstract

We explore Debatepedia, a community-authored encyclopedia of sociopolitical debates, as evidence for inferring a low-dimensional, human-interpretable representation in the domain of issues and positions. We introduce a generative model positing latent topics and cross-cutting positions that gives special treatment to person mentions and opinion words. We evaluate the resulting representation's usefulness in attaching opinionated documents to arguments and its consistency with human judgments about positions.

1 Introduction

The social web has evolved into a forum for large portions of the population to discuss and debate complex issues of societal importance. Websites like Debatepedia,[1] an online, community-authored encyclopedia of debates (§2), seek to organize some of this exchange into structured information resources that summarize arguments and link externally to texts (editorials, blog posts, etc.) that express and evoke them. Empirical NLP, we propose, has a role to play in creating a more compact and easily-interpretable way to understand the opinion space. In particular, we envision applications to computational journalism, where there is high demand for transformation of and pattern discovery in unmanageable, unstructured, evolving data (including text) to inform the public (Cohen et al., 2011).

In this paper, we develop a generative model for discovering such a representation (§3), using Debatepedia as a corpus of evidence. We draw inspiration from Lin et al. (2008) and Ahmed and Xing (2010), who used generative models to infer **topics**—distributions over words—and other word-associated variables representing perspectives or ideologies. We view topics as lexicons, and propose that *grounding* a topic model with evidence beyond bags of words can lead to more lexicon-like representations. Specifically, our generative topic model grounds topics using the hierarchical organization of arguments within Debatepedia. Further, we use named entity recognition as a preprocessing step, an existing sentiment lexicon to construct an informed prior, and we incorporate a latent, discrete **position** variable that cuts across debates.[2]

We evaluate the model informally and formally (§4). Subjectively, the model identifies reasonable topic and perspective terms, and it associates topics sensibly with important public figures. In quantitative evaluations, we find the model's representation superior to topics from vanilla latent Dirichlet allocation (Blei et al., 2003) and the joint sentiment topic model (Lin and He, 2009) in matching external texts to debates. Further, the position variables can be used to infer the *side* of an argument within a debate; our model performs with an accuracy of 86% on position prediction of the debate argument. The cross-cutting position variable is not especially consistent with human judgments, suggesting that further knowledge sources may be required to improve interpretability across issues.

2 Data

Debatepedia, like Wikipedia, is constructed by volunteer contributors and has a system of community

[1] http://dbp.idebate.org

[2] This variable might serve to cluster debate sides according to "abstract beliefs commonly shared by a group of people," sometimes called *ideologies* (Van Dijk, 1998). We do not claim that our model infers ideologies (see §4).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1858–1868,
Seattle, Washington, USA, 18-21 October 2013.

Debate: *Gun control; should laws be passed to limit gun ownership further?*	
Question: *Self-defense – Is self-defense a good reason for gun ownership?*	
Side: Yes	Side: No
Argument: A citizen has a "right" to guns as a means to self-defense: Many groups argue that a citizen should have the "right" to defend themselves, and that a gun is frequently the ...	Argument: The protection of property is not a good justification for yielding a lethal weapon. While people have a right to their property, this should not justify wielding a lethal ...
Argument: Gun restrictions and bans disadvantage citizens against armed criminals. Citizens that are not allowed to carry guns are disadvantaged against lawless criminals that ...	Argument: Robert F. Drinan, Former Democratic US Congressman, "Gun Control: The Good Outweighs the Evil", 1976 – "These graphic examples of individual instances of ...
Question: *Economic benefits – Is gun control economically beneficial?*	
Side: Yes	Side: No
Argument: Lax gun control laws are economically costly. The Coalition for Gun Control claims that, "in Canada, the costs of firearms death and injury alone have been estimated at ...	Argument: Gun sports have economic benefits. Field sports bring money into poor rural economies and provide a motivation for landowners to value environmental protection.

Table 1: An example of a Debatepedia debate on the topic "Gun control."

moderation. Many of the debate issues covered are controversial and salient in current public discourse. Because it is primarily expressed as text, Debatepedia is a *corpus* of debate topics, but it is organized hierarchically, with multiple issues in each debate topic, questions within each issue, and arguments on two sides of each question. An important feature of the corpus is the widespread quotation and linking to external articles on the web, including news stories, blog postings, wiki pages, and social media forums; here we use these external articles in evaluation (§4).

Table 1 shows excerpts from a debate page[3] from Debatepedia. Each debate contains "questions," which reflect the different aspects of a debate. In this particular debate, there are 13 questions (2 shown), ranging from economic benefits to enforceability to social impacts. For each question, there are two distinct sides, each with its own set of supporting arguments. Many of these arguments also contains links to online articles where the quotes are extracted from (not shown in Table 1). For example, in the second argument on the "No" side, there is an inline link to the article written by Congressman Drinan.[4]

Within a debate topic, the sides cut across different questions, aligning arguments together. In general, the questions are phrased so that a consistent "pro" and "con" structure is apparent throughout each debate, aligned to a high-level question (i.e., the "Yes" sides of all the questions are consistent with the same side of the larger debate). The example of Table 1 deviates from this pattern, with the self-defense "Yes" arguing "no" to the high-level debate question—*Should laws be passed to limit gun ownership further?*—and the economic "Yes" arguing "yes" to the high-level question.

Debates	1,303
Arguments	33,556
Articles linked by exactly one argument	3,352
Tokens	1,710,814
Types (excluding NE mentions)	59,601
Person named entity mentions	9,496

Table 2: Debatepedia corpus statistics. Types and tokens include unigrams, bigrams and person named entities.

Table 2 presents statistics of our corpus.

2.1 Preprocessing

We scraped the Debatepedia website and extracted the debate, question, argument, and side structure of the debate topics. We crawled the external web articles that were linked from the Debatepedia arguments. For the web articles, we extracted the main text content (ignoring boilerplate elements such as navigation and advertisments) using Boil-

[3] http://dbp.idebate.org/en/index.php/Debate:_Gun_control

[4] http://www.saf.org/LawReviews/Drinan1.html

erpipe (Kohlschütter et al., 2010).[5] We tokenized the text and filtered stopwords.[6] We considered both unigrams and bigrams in our model, keeping all unigrams and removing bigram types that appeared less than 5 times in the corpus. Although our modeling approach ultimately treats texts as bags of terms (unigrams and bigrams), one important preprocessing step was taken to further improve the interpretability of the inferred representation: named entity mentions of persons. We identified these mentions of persons using Stanford NER (Finkel et al., 2005) and treated each person mention as a single token. In our qualitative analysis of the model (§4.2), we will show how this special treatment of person mentions enables the association of well-known individuals with debate topics. Though not part of our experimental evaluation in this paper, such associations are, we believe, an interesting direction for future applications of the model.

3 Model

Our model defines a probability distribution over terms[7] that are observed in the corpus. Each term occurs in a context defined by the tuple $\langle d, q, s, a\rangle$ (respectively, a *debate*, a *question* within the debate, a *side* within the debate, and an *argument*). At each level of the hierarchy is a different latent variable:

- Each question q within debate d is associated with a distribution over topics, denoted $\boldsymbol{\theta}_{d,q}$.[8]
- Each side s of the debate d is associated with a position, denoted $i_{d,s}$ and we posit a global distribution ι that cuts across different questions and arguments. In our experiments, there are two positions, and the two sides of a debate are constrained to associate with opposing positions. As illustrated by Table 1, this assumption is not always correct, though it tends to hold most of the time.
- Each term $w_{d,q,s,a,n}$ (n is the position index of the term within an argument) is associated with one of five *functional term types*, denoted $y_{d,q,s,a,n}$. This variable is latent, except when it takes the value "entity" (e) for terms marked as named entity mentions. When it is not an entity, it takes one of the other four values: "general position" (i), "topic-specific position" (o), "topic" (t), or "background" (b). Thus, every term w is drawn from one of these 5 types of bags, and y acts as a switching variable to select the type of bag.
- For some term types (the ones where $y \in \{\mathsf{o}, \mathsf{t}\}$), each term $w_{d,q,s,a,n}$ is associated with one of T discrete topics, as indexed by $z_{d,q,s,a,n}$.

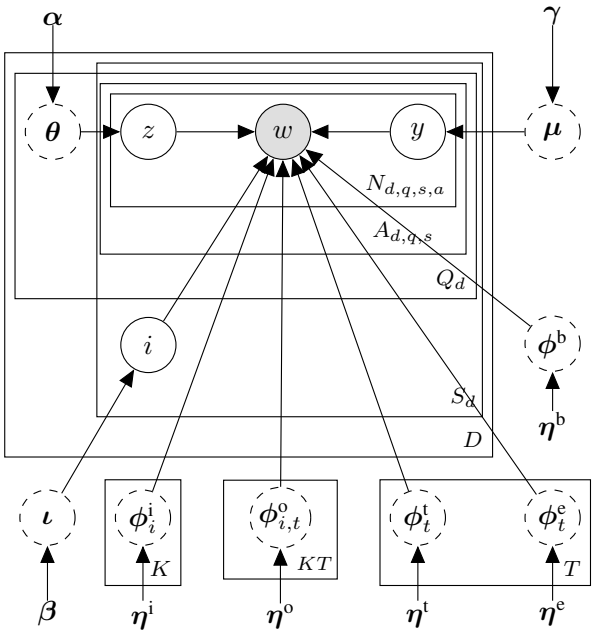


Figure 1: Plate diagram. K is the number of positions, and T is number of topics. The shaded variables are observed and dashed variables are marginalized. $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ and all $\boldsymbol{\eta}$ are fixed hyperparameters (§3.1).

Figure 1 illustrates the plate diagram for the graphical model underlying our approach. The generative story is given in Figure 2.

3.1 Priors

Typical probabilistic topic models assume a symmetric Dirichlet prior over its term distributions or

[5] http://code.google.com/p/boilerpipe

[6] www.ranks.nl/resources/stopwords.html

[7] Recall that our model includes bigrams. We treat each unigram and bigram token (after filtering discussed in §2.1) as a separate term.

[8] In future work, more sharing across questions within a debate, or more differentiation among the topic distributions for arguments under a question, might be explored. Wallach (2006) describes suitable techniques using hierarchical Dirichlet draws, and Eisenstein et al. (2011) suggests the use of sparse shocks to log-odds at different levels. Here we work on the assumption that Debatepedia's questions are the most topically coherent level, and work with a single topic mixture at this level.

1. $\forall$ topics t, draw topic-term distribution $\boldsymbol{\phi}^{\mathsf{t}}_t \sim \text{Dirichlet}(\boldsymbol{\eta}^{\mathsf{t}})$ and topic-entity distribution $\boldsymbol{\phi}^{\mathsf{e}}_t \sim \text{Dirichlet}(\boldsymbol{\eta}^{\mathsf{e}})$.
2. $\forall$ positions i, draw position-term distribution $\boldsymbol{\phi}^{\mathsf{i}}_i \sim \text{Dirichlet}(\boldsymbol{\eta}^{\mathsf{i}})$.
3. $\forall$ topics t, $\forall$ positions i, draw topic-position term distribution $\boldsymbol{\phi}^{\mathsf{o}}_{i,t} \sim \text{Dirichlet}(\boldsymbol{\eta}^{\mathsf{o}})$.
4. Draw background term distribution $\boldsymbol{\phi}^{\mathsf{b}} \sim \text{Dirichlet}(\boldsymbol{\eta}^{\mathsf{b}})$.
5. Draw functional term type distribution $\boldsymbol{\mu} \sim \text{Dirichlet}(\boldsymbol{\gamma})$.
6. Draw position distribution $\boldsymbol{\iota} \sim \text{Dirichlet}(\boldsymbol{\beta})$.
7. $\forall$ debates d:
   a. Draw $i_{d,1}, i_{d,2} \sim \text{Multinomial}(\boldsymbol{\iota})$, assigning each of the two sides to a position.
   b. $\forall$ questions q in d:
      i. Draw topic mixture proportions $\boldsymbol{\theta}_{d,q} \sim \text{Dirichlet}(\boldsymbol{\alpha})$.
      ii. $\forall$ arguments a under question q and term positions n in a:
         A. Draw topic label $z_{d,q,s,a} \sim \text{Multinomial}(\boldsymbol{\theta}_{d,q})$.
         B. Draw functional term type $y_{d,q,s,a} \sim \text{Multinomial}(\boldsymbol{\mu})$.
         C. Draw term $w_{d,q,s,a} \sim \text{Multinomial}\left(\boldsymbol{\phi}^{y_{d,q,s,a}} \mid i_{d,1}, i_{d,2}, z_{d,q,s,a}\right)$.

Figure 2: Generative story for our model of Debatepedia.

apply empirical Bayesian techniques to estimate the hyperparameters. Motivated by past efforts to exploit prior knowledge (Zhao et al., 2010; Lin and He, 2009), we use the OpinionFinder sentiment lexicon[9] (Wilson et al., 2005) to construct $\boldsymbol{\eta}^{\mathsf{i}}$ and $\boldsymbol{\eta}^{\mathsf{o}}$. Specifically, terms w in the lexicon were given parameters $\eta^{\mathsf{i}}_w = \eta^{\mathsf{o}}_w = 0.01$, and other terms were given $\eta^{\mathsf{i}}_w = \eta^{\mathsf{o}}_w = 0.001$, capturing our prior belief that opinion-expressing terms are likely to be used in expressing positions. 5,451 types were given a "boost" through this prior.

Information retrieval has long exploited the observation that a term's document frequency (i.e., the number of documents a term occurs in) is inversely related its usefulness in retrieval (Jones, 1972). We encode this in $\boldsymbol{\eta}^{\mathsf{b}}$, the prior over the background term distribution, by setting each value to the logarithm of the term's argument frequency.

The other priors were set to be symmetric: $\eta^{\mathsf{e}} = 0.01$ (entity topics), $\eta^{\mathsf{t}} = 0.001$ (topics), $\alpha = 50/T = 1.25$ (topic mixture coefficients), $\beta = 0.01$ (positions), and $\gamma = 0.01$ (functional term types). Preliminary tests showed that final topics are relatively insensitive to the values of the hyperparameters.

3.2 Inference and Parameter Estimation

Exact inference under this model, like most latent-variable topic models, is intractable. We apply collapsed Gibbs sampling, a standard approach for such models (Griffiths and Steyvers, 2004).[10] The notable deviations from typical uses of collapsed Gibbs sampling are: (i) we jointly sample $i_{d,1}$ and $i_{d,2}$ to respect the constraint that they differ; and (ii) we fix the priors, in some cases to be asymmetric, as discussed in §3.1. We perform Gibbs sampling for 2,000 iterations over the dataset, discarding the first 500 iterations for burn-in, and averaging over every 10th iteration thereafter to get estimates for our term distributions.

3.3 T and K

In all experiments, we use $T = 40$ topics and $K = 2$ positions. We did not extensively explore different values for T and K; preliminary exploration suggested that interpretability, gauged informally by the authors, degraded for higher values of either.

4 Evaluation

Recall that the aim of this work is to infer a low-dimensional representation of debate text. We estimated our model on the Debatepedia debates (not including hyperlinked articles), and conducted several evaluations of the model, each considering a different aspect of the goal. We exploit external articles hyperlinked from Debatepedia described in §2 as supporting texts for arguments, treating each one's association to an argument as variable to be predicted. Firstly, we evaluate our model on the article associating task. Secondly, we evaluate our model on the position prediction task. Then, we compare

[9] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

[10] Because this technique is well known in NLP, details are relegated to supplementary material.

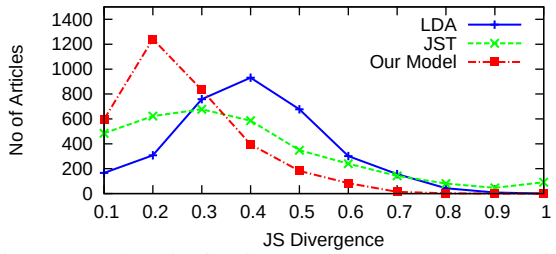


Figure 3: The distribution over Jensen-Shannon divergences between a hyperlinked article and the corresponding Debatepedia argument, $n = 3,352$.

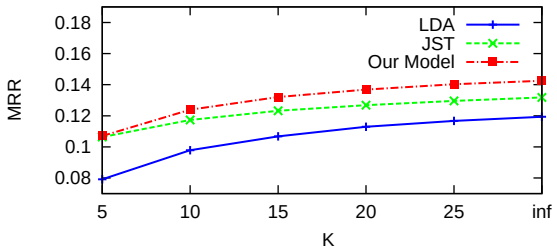


Figure 4: Mean reciprocal ranks for the association task.

our model's positional assignment of arguments to human annotated clusterings. Finally, we present qualitative discussion.

4.1 Quantitative Evaluation

4.1.1 Topics

As described in §2, our corpus includes 3,352 articles hyperlinked by Debatepedia arguments.[11] Our model can be used to infer the posterior over topics associated with such an article, and we compare that distribution to that of the Debatepedia article that links to it. Calculating the similarity of these distributions, we get an estimate of how closely our model can associate text related to a debate with the specific argument that linked to it. We compare with LDA (Blei et al., 2003), which ignores sentiment, and the joint sentiment topic (JST) model (Lin and He, 2009), an unsupervised model that jointly captures sentiment and topic.[12] Using Jensen-Shannon divergence, we find that our approach embeds these pairs significantly closer than LDA and JST (also trained with 40 topics), under a Wilcoxon signed rank test ($p < 0.001$). Figure 3 shows the histogram of divergences between our model, JST, and LDA.

Associating external articles. More challenging, of course, is *selecting* the argument to which an external article should be associated. We used the Jensen-Shannon divergence between topic distributions of articles and arguments to rank the latter, for each article. The mean reciprocal rank scores (Voorhees, 1999) for LDA, JST, and our model were 0.1272, 0.1421, and 0.1507, respectively; the difference is significant (Wilcoxon signed rank test, $p < 0.001$). We found the same pattern for MRR@k, $k \in \{5, 10, 15, 20, 25, \infty\}$, as shown in Figure 4.

It is likely possible to engineer more accurate models for attaching articles to arguments, but the attachment task is our aim only insofar as it contributes to an overall assessment of an inferred representation's quality.

4.1.2 Positions

Positional distance by topic. We next consider the JS divergences of position term distributions by topic; for each topic t, we consider the divergence between inferred values for $\phi^{\circ}_{1,t}$ and $\phi^{\circ}_{2,t}$. Figure 5 shows these measurements sorted from most to least different; these might be taken as evidence for which issue areas' arguments are more *lexically* distinguishable by side, perhaps indicating less common ground in discourse or (more speculatively) greater controversy. For example, our model suggests that debates relating to topics like presidential politics, foreign policy, teachers, women's health, religion, and Israel/Palestine are more heated (within the Debatepedia community at the time the debates took place) than those about the minimum wage, Iran as a nuclear threat, or immigration.

Predicting positions for arguments. We tested our model's ability to infer the positions of arguments. In this experiment (only), we held out 3,000 arguments during parameter estimation. The held-out arguments were selected so that every debate side maintained at least one argument whose inferred side could serve as the correct answer for the held-out argument. We then inferred i for each held-out argument from debate d and side s, given the parameters, and compared it with the value of $i_{d,s}$ inferred during parameter estimation. The model achieved 86% accuracy (Table 3 shows the confu-

[11] We consider only those articles linked by a single Debatepedia argument.

[12] JST multiplies topics out by the set of sentiment labels, assigning each token to both a topic and a sentment. We use the OpinionFinder lexicon in JST's prior in the same way it is used in our model.

sion matrix). Note that JST does not provide a baseline for comparison, since it does not capture debate sides.

	$i=1$	$i=2$
$i^*=1$	1,272	216
$i^*=2$	199	1,313

Table 3: Confusion matrix for position prediction on held-out *arguments*.

Predicting positions for external articles. We can also use the model to predict the position adopted in an external text. For articles linked from within Debatepedia, we have a gold standard: from which side of a debate was it linked? After using the model to infer a position variable for such a text, we can check whether the inferred position variable matches that of the argument that links to it. Table 4 shows that our model does not successfully complete this task, assigning about 60% of both kinds of articles $i = 1$.

	$i=1$	$i=2$
$i^*=1$	1,042	623
$i^*=2$	1,043	644

Table 4: Confusion matrix for position prediction on hyperlinked *articles*.

Genre. We manually labeled 500 of these articles into six genre categories. We had two annotators for this task (Cohen's $\kappa = 0.856$). These categories, in increasing order of average Jensen-Shannon divergence, are: blogs, editorials, wiki pages, news, other, and government. Figure 6 shows the results. While the only difference between the first and last groups are surprising by chance, we are encouraged by our model's suggestion that blogs and editorials may be more "Debatepedia argument-like" than news and government articles.

Note that our model is learned only from text *within* Debatepedia; it does not observe the text of external linked articles. Future work might incorporate this text as additional evidence in order to capture effects on language stemming from the interaction of position and genre.

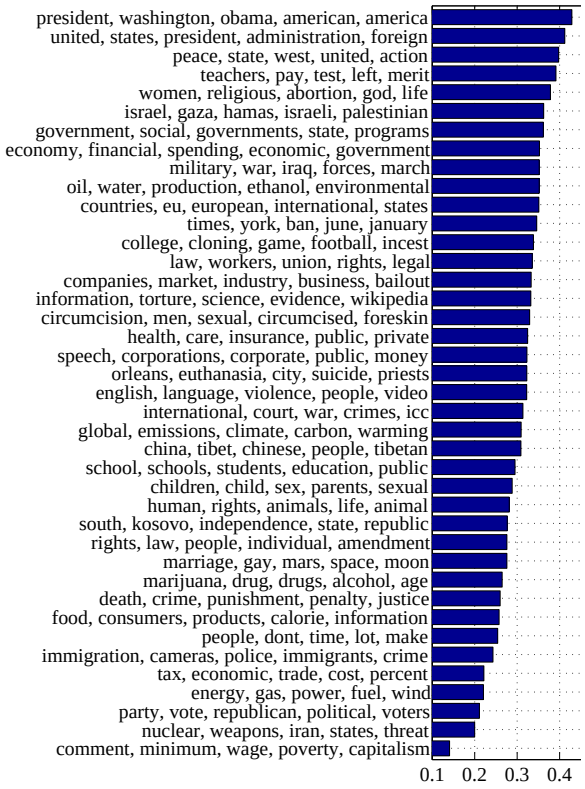


Figure 5: Jensen-Shannon divergences between topic-specific positional term distributions, for each topic. Topics are labeled by their most frequent terms from ϕ^t.

4.1.3 Comparison to Human Judgments of Positions

We compared our model's inferred positions to human judgments. For each of the 11 topics in Table 8, we selected two associated debates with more arguments than average (24.99). The debates were provided to each of three human annotators,[13] who

[13] All were native English-speaking American graduate students not otherwise involved in this research. Each is known by the authors to have basic literacy with issues and debates in

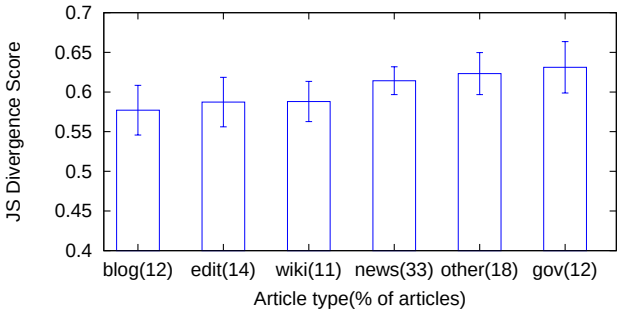


Figure 6: Position prediction on 500 hyperlinked *articles* by genre.

	"Israel-Palestine"	"Same-sex marriage"	"Drugs"	"Healthcare"	"Death penalty"	"Abortion"
i_1	pre emptive	same sex	hands free	single payer	anti death	pro choice
	israeli palestinian	long term	performance enhancing	so called	non violent	pro life
	open and shut	second class	in depth	self sustaining	african american	non muslim
i_2	two state	opposite sex	long term	government run	semi automatic	would be
	long term	well intentioned	high speed	government approved	high profile	full time
	self destructive	day time	short term	high risk	hate crime	late term

a. Our model: topic-specific position bigrams associated with six selected topics.

–	war	large	illegal	support	death	power
	assault	possibility	abuse	force	penalty	limit
	disproportionate	problems	high	threat	murder	civil
+	peace	civil	disease	care	power	care
	independence	rights	nature	universal	clean	suicide
	self-determination	affirmative	potential	uninsured	waste	death

b. JST: sentiments associated with six selected topics manually aligned to our model's topics.

Table 6: Terms associated with selected topics. The labels and alignments between the two models' topics were assigned manually. (a.) Our model: topic-specific position bigrams which are ranked by comparing the log odds conditioned on the position and topic: $\log \phi^o_{i_1,t,w} - \log \phi^o_{i_2,t,w}$. We show the top three terms for each position (b.) JST: we show the top three terms for each sentiment (negative and positive).

	A1 (11)	A2 (5)	A3 (16)
Model (2)	3.21	2.58	3.45
A1 (11)		2.15	2.15
A2 (5)			2.63

Table 5: Variation of information scores for each pairing of annotators and model.

were instructed to group the 44 sides of the debates. The instructions stated:

> Our goal is to see what you think about how the different sides of different debates can be lined up. You might find it convenient to think of these in terms of political philosophies, contemporary political party platforms, or something else. Any of these is fine; we want you to tell us the grouping you find most reasonable.

All three annotators (hereafter denoted A1, A2, and A3) used fairly involved labeling schemes; the annotators used 37, 30, and 16 unique labels, respectively.[14] A1 used keyword lists to label items; we coarsened his labels manually by removing or merging less common keywords (resulting in: *Republican*, *Democrat*, *science/environment*, *nanny*, *political reform*, *fiscal liberal*, *fiscal conservative*, *libertarian*, *Israel*, *Palestine*, and one unlabeled side). A2 provided a coarse annotation along with each fine-grained one (*liberal*, *conservative*, *?*, and two unlabeled sides). We used 100 samples from our Gibbs sampler to estimate posteriors for each $i_{d,s}$; these were always 99% or more in agreement, so we mapped each debate side into its single most probable cluster. Recall that the two sides of each debate must be in different clusters.

Table 5 shows the variation of information measure (Meila, 2003) for each pairing among the three annotators and our model. The model agrees with A2's coarse clustering most closely, and in fact is closer to A2's clustering than A2 is to A3's; it also agrees with A2's coarse clustering better than A2's coarse and fine clusterings agree (3.36, not shown in the table). This is promising, but we do not have confidence that the positional dimension is being captured especially well in this model; for those debate-sides labeled *liberal* or *conservative* by A2, the best match of our two positions was still only in agreement only about 60% of the time, and agreement with each human annotator is within the interval of what would be expected if each debate's sides were assigned uniformly at random to positions.[15]

Remarks. Within debates and within topics, the model uses the position variable to distinguish sides well. For external text, the model performs well on articles such as blogs and editorials but on others the positional categories do not seem meaning-

American politics.

[14] In a small number of cases, an annotator declined to label a side. Each unlabeled item received its own cluster.

[15] This was determined using a Monte Carlo simulation with 1,000 samples.

Topic	$i = 1$	$i = 2$
None (ϕ^{i})	vice president, c sections, twenty four, cross pressures, pre dates, anti ballistic, cost effectiveness, anti landmine, court appointed, child poverty	cross examination, under runs, hand outs, half million, non christians, break down, counter argument, seventy five, co workers, run up
"Israel-Palestine"	pre emptive, israeli palestinian, open and shut, first time, hamas controlled, democratically elected	two state, long term, self destructive, secretary general, right wing, all out, near daily, short term
"Same-sex marriage"	same sex, long term, second class, blankenhorn rauch, wrong headed, self denial, left handed	opposite sex, well intentioned, day time, planet wide, day night, child rearing, low earth, one way, one third
"Drugs"	hands free, performance enhancing, in depth, hand held, best kept, non pharmaceutical, anti marijuana	long term, high speed, short term, peer reviewed, alcohol related, mind altering, inner city, long lasting
"Healthcare"	single payer, so called, self sustaining, public private, for profit, long run, high cost, multi payer	government run, government approved, high risk, two tier, government appointed, low cost, set up
"Death penalty"	anti death, non violent, african american, self help, cut and cover, heavy handed, dp equivalent	semi automatic, high profile, hate crime, assault weapons, military style, high dollar, self protective
"Abortion"	pro choice, pro life, non muslim, well educated, anti abortion, much needed, church state, birth control	would be, full time, late term, judeo christian, life style, day to day, non christian, child bearing

Table 7: General position (first row) and topic-specific position bigrams associated with six selected topics.

Topic	Terms	Person entity mentions
"Israel-Palestine"	israel, gaza, hamas, israeli, palestinian	Benjamin Netanyahu, Al Jazeera, Mavi Marmara, Nicholas Kristoff, Steven R. David
"Same-sex marriage"	marriage, gay, mars, space, moon	Buzz Aldrin, Andrew Sullivan, Moon Base, Scott Bidstrup, Ted Olson
"Drugs"	marijuana, drug, drugs, alcohol, age	Four Loko, Evo Morales, Toni Meyer, Sean Flynn, Robert Hahn
"Healthcare"	health, care, insurance, public, private	Kent Conrad, Paul Hsieh, Paul Krugman, Ezra Klein, Jacob Hacker
"Death penalty"	death, crime, punishment, penalty, justice	Adam Bedau, Thomas R. Eddlem, Jeff Jacoby, John Baer, Peter Bronson
"Abortion"	women, religious, abortion, god, life	Ronald Reagan, John Paul II, Sara Malkani, Mother Teresa, Marcella Alsan

Table 8: For 6 selected topics (labels assigned manually), top terms (ϕ^{t}) and person entities (ϕ^{e}). Bigrams were included but did not rank in the top five for these topics. The model has conflated debates relating to same-sex marriage with the space program.

ful, perhaps due to the less argumentative nature of other kinds of articles. Noting the vast literature focusing on ideological positions expressed in text, we believe this failure suggests (i) that broad-based positions that hold across many topics may require richer textual representations (see, e.g., the "syntactic priming" of Greene and Resnik, 2009), or (ii) that an alternative representation of positions, such as the spatial models favored by political scientists (Poole and Rosenthal, 1991), may be more discoverable. Aside from those issues, a stronger theory of positions may be required. Such a theory could be encoded in a more informative prior or weaker independence assumptions across debates. Finally, exploiting explicitly ideological texts alongside the moderated arguments of Debatepedia might also help to identify textual associations with general positions (Sim et al., 2013). We leave these directions to future work.

4.2 Qualitative Analysis

Of the $T = 40$ topics our model inferred, we subjectively judged 37 to be coherent; a glimpse of each is given in Figure 5. We manually selected six of the most interpretable topics for further evaluation.

As a generative modeling approach, our model was designed for the purpose of reducing the dimensionality of the sociopolitical debate space, as evidenced by Debatepedia. It is like other topic models in this regard, but we believe that some effects of our design choices are noteworthy. Table 6 compares the positional bigrams of our model to the sentiments inferred by JST. We observe the benefit of our model in identifying terms associated with positions on social issues, while JST selects more general sentiment terms.

Table 7 shows bigrams most strongly associated with general position distributions ϕ^{i} and selected topic-position distributions ϕ°.[16] We see the potential benefit of multiword expressions. Although we have used frequent bigrams as a poor man's approximation to multiword expression analysis, we find the topic-specific positions terms to be subjectively evocative. While somewhat internally coherent, we do not observe consistent alignment across topics, and the general distributions ϕ^{i} are not suggestive.

The separation of personal name mentions into their own distributions, shown for some topics in Table 8, gives a distinctive characterization of topics based on relevant personalities. Subjectively, the top individuals are relevant to the subject matter associated with each topic (though the topics are not always pure; same-sex marriage and the space program are merged, for example).

5 Related Work

Insofar as debates are subjective, our study is related to **opinion mining**. Subjective text classification (Wiebe and Riloff, 2005) leads to opinion mining tasks such as opinion extraction (Dave et al., 2003), positive and negative polarity classification (Pang et al., 2002), sentiment target detection (Hu and Liu, 2004; Ganapathibhotla and Liu, 2008), and feature-opinion extraction (Wu et al., 2009). The above studies are conducted mostly on product reviews, a domain with a simpler opinion landscape and more concrete rationales for those opinions, compared to sociopolitical debates.

Generative **topic models** have been successfully implemented in opinion mining tasks such as feature identification (Titov and McDonald, 2008), entity-topic extraction (Newman et al., 2006), mining contentious expressions and interactions (Mukherjee and Liu, 2012) and specific aspect-opinion word extraction from labeled data (Zhao et al., 2010). Most relevant to this research is work on feature-sentiment extraction (Lin and He, 2009; Mei et al., 2007). Mei et al. (2007) built on PLSI, which is problematic for generalizing beyond the training sample. The JST model of Lin and He (2009) is an LDA-based topic model in which each word token is assigned both a sentiment and a topic; they exploited a sentiment lexicon in the prior distribution. Our model is closely related, but introduces a switching variable that assigns *some* tokens to positions, some to topics, and some to both. Unlike Lin and He's sentiments, our model's positions are associated with the two sides of a debate, and we incorporate topics at the level of questions within debates.

Some studies have specifically analyzed **contrastive viewpoints** or **stances** in general discussion text.Agrawal et al. (2003) used graph mining based method to classify authors in to opposite camps for a given topic. Paul et al. (2010) developed an unsupervised method for summarizing contrastive opinions from customer reviews. Abu-Jbara et al. (2012) and Dasigi et al. (2012) developed techniques to address the problem of automatically detecting subgroups of people holding similar stances in a discussion thread.

Several prior studies have considered **debates**. Cabrio and Villata (2012) developed a system based on argumentation theory which recognizes the entailment and contradiction relationships between two texts. Awadallah et al. (2011) used a debate corpus as a seed for extracting person-opinion-topic tuples from news and other web documents and in later work classified the quotations to specific topics and polarity using language models (Awadallah et al., 2012). Somasundaran and Wiebe (2009) and Anand et al. (2011) were interested in ideological content in debates, relying on discourse structure and leveraging sentiment lexicons to recognize stances.

Closer to the methodology we describe, Lin et al. (2008) presented a statistical model for political discourse that incorporates both topics and ideologies; they used debates on the Israeli-Palestinian conflict. Fortuna et al. (2009) showed that it is possible to isolate a subset of terms from media content that are informative of a news organization's bias towards a particular issue. Ahmed and Xing (2010) introduced multi-level latent Dirichlet allocation, and Eisenstein et al. (2011) introduced sparse additive generative models, both conceived as extensions to well-established probabilistic modeling techniques (Blei et al., 2003); these were applied to debates and political blog datasets. Our approach builds on these models (especially the switching variables of Ahmed and Xing). We go farther in jointly modeling

[16] For more topics, please refer to the supplementary notes.

text across *many* debates evidenced by the structure of Debatepedia, thus grounding our models more solidly in familiar sociopolitical issues, and in making extensive use of existing NLP resources.

6 Conclusion

Using text from Debatepedia, we inferred topics and position term lexicons in the domain of sociopolitical debates. Our approach brings together tools from information extraction and sentiment analysis into a latent-variable topic model and exploits the hierarchical structure of the dataset. Our qualitative and quantitative evaluations show the model's strengths and weaknesses.

Acknowledgments

The authors thank several anonymous reviewers, Justin Gross, David Kaufer, and members of the ARK group at CMU for helpful feedback on this work and gratefully acknowledge the assistance of the annotators. This research is supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme Office, by an A*STAR fellowship to Y.S., and by Google's support of the Reading is Believing project at CMU.

References

Amjad Abu-Jbara, Mona Diab, Pradeep Dasigi, and Dragomir Radev. 2012. Subgroup detection in ideological discussions. In *Proceedings of ACL*.

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *WWW '03*.

Amr Ahmed and Eric P. Xing. 2010. Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective. In *Proceedings of EMNLP*.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: classifying stance in online debate. In *Proceedings of the Second Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*.

Rawia Awadallah, Maya Ramanath, and Gerhard Weikum. 2011. OpinioNetIt: Understanding the opinions-people network for politically controversial topics. In *Proceedings of CIKM*.

Rawia Awadallah, Maya Ramanath, and Gerhard Weikum. 2012. PolariCQ: Polarity classification of political quotations. In *Proceedings of CIKM*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of ACL*.

Sarah Cohen, James T. Hamilton, and Fred Turner. 2011. Computational journalism. *Communications of the ACM*, 54(10):66–71.

Pradeep Dasigi, Weiwei Guo, and Mona Diab. 2012. Genre independent subgroup detection in online discussion threads: a pilot study of implicit attitude using latent textual semantics. In *Proceedings of ACL*.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of WWW*.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text. In *Proceedings of ICML*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Blaz Fortuna, Carolina Galleguillos, and Nello Cristianini. 2009. Detecting the bias in media with statistical learning methods. In Ashok N . Srivastava and Mehran Sahami, editors, *Text Mining: Classification, Clustering, and Applications*, pages 27–50. Chapman & Hall/CRC.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of COLING*.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of HLT-NAACL*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of CIKM*.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of WSDM*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*.

Wei-Hao Lin, Eric Xing, and Alexander Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Proceedings of ECML-PKDD*.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of WWW*.

Marina Meila. 2003. Comparing clusterings by the variation of information. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer.

Arjun Mukherjee and Bing Liu. 2012. Mining contentions from discussions and debates. In *Proceedings of KDD*.

David Newman, Chaitanya Chemudugunta, and Padhraic Smyth. 2006. Statistical entity-topic models. In *Proceedings of KDD*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

Michael J. Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of EMNLP*.

Keith Poole and Howard Rosenthal. 1991. Patterns of congressional voting. *American Journal of Political Science*, pages 118–178.

Yanchuan Sim, Brice Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of EMNLP*.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of ACL*.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*.

Teun A. Van Dijk. 1998. *Ideology: A Multidisciplinary Approach*. Sage Publications Limited.

Ellen M. Voorhees. 1999. The trec-8 question answering track report. In *Proceedings of TREC*.

Hanna M. Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of ICML*.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of CICLing*.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: a system for subjectivity analysis. In *Proceedings of HLT-EMNLP*.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of EMNLP*.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of EMNLP*.

A Unified Model for Topics, Events and Users on Twitter

Qiming Diao
Living Analytics Research Centre
School of Information System
Singapore Management University
qiming.diao.2010@smu.edu.sg

Jing Jiang
Living Analytics Research Centre
School of Information System
Singapore Management University
jingjiang@smu.edu.sg

Abstract

With the rapid growth of social media, Twitter has become one of the most widely adopted platforms for people to post short and instant message. On the one hand, people tweets about their daily lives, and on the other hand, when major events happen, people also follow and tweet about them. Moreover, people's posting behaviors on events are often closely tied to their personal interests. In this paper, we try to model topics, events and users on Twitter in a unified way. We propose a model which combines an LDA-like topic model and the Recurrent Chinese Restaurant Process to capture topics and events. We further propose a duration-based regularization component to find bursty events. We also propose to use event-topic affinity vectors to model the association between events and topics. Our experiments shows that our model can accurately identify meaningful events and the event-topic affinity vectors are effective for event recommendation and grouping events by topics.

1 Introduction

Twitter is arguably the most popular microblog site where people can post short, instant messages to share with families, friends and the rest of the world. For content analysis on Twitter, two important concepts have been repeatedly visited: (1) **Topics**. These are longstanding themes that many personal tweets revolve around. Example topics range from music and sports to more serious ones like politics and religion. Much work has been done to analyze topics on Twitter (Ramage et al., 2010; Hong and Davison, 2010; Zhao et al., 2011; Lau et al., 2012). (2) **Events**. These are things that take place at a certain time and attract many people's short-term attention in social media. Example events include concerts, sports games, scandals and elections. Event detection on Twitter has been a hot research topic in recent years (Petrović et al., 2010; Weng and Lee, 2011; Becker et al., 2011; Diao et al., 2012; Li et al., 2012).

The concepts of topics and events are orthogonal in that many events fall under certain topics. For example, concerts fall under the topic about music. Furthermore, being *social* media, Twitter users play important roles in forming topics and events on Twitter. Each user has her own topic interests, which influence the content of her tweets. Whether a user publishes a tweet related to an event also largely depends on whether her topic interests match the nature of the event. Modeling the interplay between topics, events and users can deepen our understanding of Twitter content and potentially aid many predication and recommendation tasks. In this paper, we aim to construct a unified model of topics, events and users on Twitter. Although there has been a number of recent studies on event detection on Twitter, to the best of our knowledge, ours is the first that links the topic interests of users to their tweeting behaviors on events.

Specifically, we propose a probabilistic latent variable model that identifies both topics and events on Twitter. To do so, we first separate tweets into *topic tweets* and *event tweets*. The former are related to a user's personal life, such as a tweet complaining about the traffic condition or wishing a friend

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1869–1879,
Seattle, Washington, USA, 18-21 October 2013.

happy birthday. The latter are about some major global event interesting to a large group of people, such as a tweet advertising a concert or commenting on an election result. Although considering only topic tweets and event tweets is a much simplified view of the diverse range of tweets, we find it effective in finding meaningful topics and events. We further use an LDA-like model (Blei et al., 2003) to discover topics and the Recurrent Chinese Restaurant Process (Ahmed and Xing,) to discover events. Details are given in Section 3.1.

Our major contributions lie in two novel modifications to the base model described above. The first is a duration-based regularization component that punishes long-term events (Section 3.2). Because events on Twitter tend to be bursty, this modification presumably can produce more meaningful events. More specifically, we borrow the idea of using pseudo-observed variables to regularize graphical models (Balasubramanyan and Cohen, 2013), and carefully design the pseudo-observed variable in our task to capture the burstiness of events. The second modification is adding event-topic affinity vectors inspired by PMF-based collaborative filtering (Salakhutdinov and Mnih, 2008) (Section 3.3). It uses the latent topics to explain users' preferences of events and subsequently infers the association between topics and events.

We use a real Twitter data set consisting of 500 users to evaluate our model (Section 4). We find that the model can discover meaningful topics and events. Comparison with our base model and with an existing model for event discovery on Twitter shows that the two modifications are both effective. The duration-based regularization helps find more meaningful events; the event-topic affinity vectors improve an event recommendation task and helps produce a meaningful organization of events by topics.

2 Related Work

Study of topics, events and users on Twitter is related to several branches of work. We review the most interesting and relevant work below.

Event detection on Twitter: There have been quite a few studies in this direction in recent years, including both online detection (Sakaki et al., 2010; Petrović et al., 2010; Weng and Lee, 2011; Becker et al., 2011; Li et al., 2012) and offline detection (Diao et al., 2012). Online detection is mostly concerned with early detection of major events, so efficiency of the algorithms is the main focus. These algorithms do not aim to identify *all* relevant tweets, nor do they analyze the association of events with topics. In comparison, our work focuses on modeling topics, events and users as well as their relation. Recently, Petrović et al. (2013) pointed out that Twitter stream does not lead news stream for major news events, but Twitter stream covers a much wider range of events than news stream. Our work helps better understand these additional events on Twitter and their relations with users' topic interests. Our model bears similarity to our earlier work (Diao et al., 2012), but we use a non-parametric model (RCRP) to discover events directly inside the probabilistic model.

Temporal topic modeling: A number of models have been proposed for the temporal aspect of topics (Blei and Lafferty, 2006; Wang and McCallum, 2006; Wang et al., 2007; Hong et al., 2011), but most of them fix the number of topics. The Recurrent Chinese Restaurant Process (Ahmed and Xing,) was proposed to model the life cycles of topics and allows an infinite number of topics. It has later been combined with LDA to model both topics and events in news streams and social media streams (Ahmed et al., 2011; Tang and Yang, 2012). Our work also jointly models topics and events, but different from previous work, we do not assume that every document (tweet in our case) belongs to an event, which is important because Twitter contains many personal posts unrelated to major events.

Collaborative filtering with LDA: Part of our model is inspired by work on collaborative filtering based on probabilistic matrix factorization (PMF) (Salakhutdinov and Mnih, 2008). Recently there has been some work combining LDA with PMF to recommend items with textual content such as news articles and advertisements (Wang and Blei, 2011; Agarwal and Chen, 2010). They use topics to interpret the latent structure of users and items. We borrow their idea but our items are events, which are not known and have to be discovered by our model.

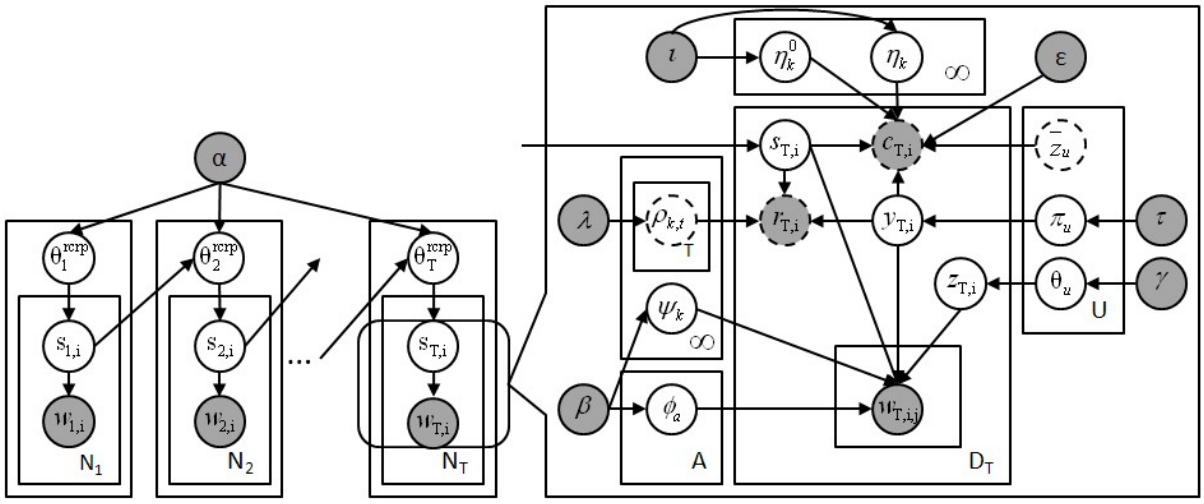


Figure 1: Plate notation for the whole model, in which pseudo-observed variables and distributions based on empirical counts are shown as dotted nodes.

3 Our Model

In this section, we present our model for topics, events and users on Twitter. We assume that we have a stream of tweets which are divided into T epoches. Let $t \in \{1, 2, \ldots, T\}$ be the index of an epoch. Each epoch contains a set of tweets and each tweet is a bag of words. We use $w_{t,i,j} \in \{1, 2, \ldots, V\}$ to denote the j-th word of the i-th tweet in the t-th epoch, where V is the vocabulary size. The author of the i-th tweet in the t-th epoch (i.e. the Twitter user who publishes the tweet) is denoted as $u_{t,i} \in \{1, 2, \ldots, U\}$, where U is the total number of Twitter users we consider.

We first present our base model in Section 3.1. We then introduce a duration-based regularization mechanism to ensure the burstiness of events in Section 3.2. In Section 3.3 we discuss how we model the relation between topics and events using event-topic affinity vectors. Finally we discuss model inference in Section 3.4.

3.1 The Base Model

Recall that our objective is to model topics, events, users and their relations. As in many topic models, our topic is a multinomial distribution over words, denoted as ϕ_a where a is a topic index. Each event is also a multinomial distribution over words, denoted as ψ_k where k is an event index. Because topics are long-standing and stable, we fix the number of topics to be A, where A can be tuned based on historical data. In contrast, events emerge and die along the timeline. We therefore use a non-parametric model called the Recurrent Chinese Restaurant Process (RCRP) (Ahmed and Xing,) to model the birth and death of events. To model the relation between users and topics, we assume each user u has a multinomial distribution over topics, denoted as θ_u.

As we have discussed, we separate tweets into two categories, topic tweets and event tweets. Separation of these two categories is done through a latent variable y sampled from a user-specific Bernoulli distribution π_u. For topic tweets, the topic is sampled from the corresponding user's topic distribution θ_u. For event tweets, the event is sampled according to RCRP. We now briefly review RCRP. Generally speaking, RCRP assumes a Chinese Restaurant Process (CRP) (Blackwell and MacQueen, 1973) for items within an epoch and chains up the CRPs in adjacent epochs along the timeline. Specifically, in our case, the generative process can be described as follows. Tweets come in according to their timestamps. In the t-th epoch, for the i-th tweet, we first flip a biased coin based on probability π_u to decide whether this tweet is event-related. If it is, then we need to decide which event it belongs to. It could be an existing event that has at least one related tweet in the

previous epoch or the current epoch, or it could be a new event. Let $n_{k,t-1}$ denote the number of tweets related to event k at the end of epoch $(t-1)$. Let $n_{k,t}^{(i)}$ denote the number of tweets related to event k in epoch t before the i-th tweet comes. Let N_{t-1} denote the total number of event-related tweets in epoch $(t-1)$ and $N_t^{(i)}$ denote the number of event-related tweets in epoch t before the i-th tweet. Then RCRP assumes that the probability for the i-th tweet to join event k is $\frac{n_{k,t-1}+n_{k,t}^{(i)}}{N_{t-1}+N_t^{(i)}+\alpha}$ and the probability to start a new event is $\frac{\alpha}{N_{t-1}+N_t^{(i)}+\alpha}$, where α is a parameter. As we can see, RCRP naturally captures the "rich-get-richer" phenomenon in social media.

Finally we place Dirichlet and Beta priors on the various parameters in our model. Formally, the generative process of our base model is outlined in Figure 2, excluding the lines in bold and blue. We also show the plate notation in Figure 1, in which the Recurrent Chinese Restaurant Process is represented as an infinite dynamic mixture model (Ahmed and Xing,) and θ_t^{rcrp} means the distribution on an infinite number of events in epoch t. D_t is the total number of tweets (both event-related and topic tweets), while N_t represents the number event-related tweets in epoch t.

3.2 Regularization on Event Durations

As we have pointed out, events on Twitter tend to be bursty, i.e. the duration of an event tends to be short, but this characteristic is not captured by RCRP. While there can be different ways to incorporate this intuition, here we adopt the idea of regularization using pseudo-observed variables proposed recently by Balasubramanyan and Cohen (2013). We introduce a pseudo-observed binary variable $r_{t,i}$ for each tweet, where the value of $r_{t,i}$ is set to 1 for all tweets. We assume that this variable is dependent on the hidden variables y and s. Specifically, if $y_{t,i}$ is 0, i.e. the tweet is topic-related, then $r_{t,i}$ gets a value of 1 with probability 1. If $y_{t,i}$ is 1, then we look at all the tweets that belong to event $s_{t,i}$. Our goal is to make sure that this tweet is temporally close to these other tweets. So we assume that $r_{t,i}$ gets a value of 1 with probability $\exp(-\sum_{t'=1,|t'-t|>1}^{T}\lambda|t-t'|n_{s_{t,i},t'})$, where $n_{s_{t,i},t'}$ is the number of tweets in epoch t' that belong to event $s_{t,i}$ and $\lambda > 0$ is a parameter. We can see that when we factor in the generation of these pseud-observed variables r, we penalize long-term events and favor events whose tweets are concentrated along the timeline. Generation of these variables r is shown in bold and blue in Figure 2.

- For each topic $a = 1, \ldots, A$
  - draw $\phi_a \sim \text{Dirichlet}(\beta)$
- For each user $u = 1, \ldots, U$
  - draw $\theta_u \sim \text{Dirichlet}(\gamma), \pi_u \sim \text{Beta}(\tau)$
- For each epoch t and tweet i
  - draw $y_{t,i} \sim \text{Bernoulli}(\pi_{u_{t,i}})$
  - If $y_{t,i} = 0$
    - draw $z_{t,i} \sim \text{Multinomial}(\theta_{u_{t,i}})$
    - For each j, draw $w_{t,i,j} \sim \text{Multinomial}(\phi_{z_{t,i}})$
  - If $y_{t,i} = 1$
    - draw $s_{t,i}$ from RCRP
    - If $s_{t,i}$ is a new event
      - draw $\psi_{s_{t,i}} \sim \text{Dirichlet}(\beta)$
      - **draw $\eta^0_{s_{t,i}} \sim$ Gaussian$(0, \iota^{-1})$**
      - **draw $\boldsymbol{\eta}_{s_{t,i}} \sim$ Gaussian$(0, \iota^{-1} I_A)$**
    - **draw $r_{t,i} \sim$ Bernoulli$(\rho_{s_{t,i},t})$, where $\rho_{s_{t,i},t} = \exp(-\sum_{t'=1,|t'-t|>1}^{T}\lambda|t'-t|n_{s_{t,i},t'})$**
    - **draw $c_{t,i} \sim$ Gaussian$(\eta^0_{s_{t,i}} + \boldsymbol{\eta}^T_{s_{t,i}} \cdot \bar{\boldsymbol{z}}_{u_{t,i}}, \epsilon^{-1})$**
    - For each j, draw $w_{t,i,j} \sim \text{Multinomial}(\psi_{s_{t,i}})$

Figure 2: The generative process of our model, in which the duration-based regularization (section 3.2) and the event-topic affinity vector (section 3.3) are in blue and bold lines.

3.3 Event-Topic Affinity Vectors

So far in our model topics and events are not related. However, many events are highly related to certain topics. For example, a concert is related to music while a football match is related to sports. We would like to capture these relations between topics and events. One way to do it is to assume that event tweets also have topical words sampled from the event's topic distribution, something similar to the models by Ahmed et al. (2011) and by Tang and Yang (2012). However, our prelimiary experiments show that this idea does not work well on Twitter, mainly because tweets are too short. Here we explore another approach inspired by recommendation methods based on probabilistic matrix factorization (Salakhutdinov and Mnih, 2008). The idea is that when a user posts a tweet about an event, we can treat the event as an item and this posting be-

havior as adoption of the item. If we assume that the adoption behavior is influenced by some latent factors, i.e. the latent topics, then basically we would like the topic distribution of this user to be close to that of the event.

Specifically, we assume that each event k has associated with it an A-dimensional vector $\boldsymbol{\eta}_k$ and a parameter η_k^0. The vector $\boldsymbol{\eta}_k$ represents the event's affinity to topics. η_k^0 is a bias term that represents the inner popularity of an event regardless of its affinity to any topic. We further assume that each tweet has another pseudo-observed variable $c_{t,i}$ that is set to 1. For topic tweets, $c_{t,i}$ gets a value of 1 with probability 1. For event tweets, $c_{t,i}$ is generated by a Gaussian distribution with mean equal to $\eta^0_{s_{t,i}} + \boldsymbol{\eta}_{s_{t,i}} \cdot \bar{\boldsymbol{z}}_{u_{t,i}}$, where $\bar{\boldsymbol{z}}_u$ is an A-dimensional vector denoting the empirical topic distribution of user u's tweets. This treatment follows the practice of fLDA by Agarwal and Chen (2010). Let $\bar{C}_{u,a}$ be the number of tweets by user u assigned to topic a, based on the values of the latent variables y and z. Then

$$\bar{\boldsymbol{z}}_{u,a} = \frac{\bar{C}_{u,a}}{\sum_{a'=1}^{A} \bar{C}_{u,a'}},$$
$$c_{t,i} \sim \text{Gaussian}(\eta^0_{s_{t,i}} + \boldsymbol{\eta}_{s_{t,i}} \cdot \bar{\boldsymbol{z}}_{u_{t,i}}, \epsilon^{-1}),$$

where ϵ is a parameter. We generate $\boldsymbol{\eta}_k$ and η_k^0 using Gaussian priors once event k emerges. The generation of the variables c is shown in bold and blue in Figure 2.

3.4 Inference

We train the model using a stochastic EM sampling scheme. In this scheme, we alternate between Gibbs sampling and gradient descent. In the Gibbs sampling part, we fix the values of η_k^0 and $\boldsymbol{\eta_k}$ for each event k, and then we sample the latent variables $y_{t,i}$,$z_{t,i}$ and $s_{t,i}$ for each tweet. In the gradient descent part, we update the event-topic affinity vectors $\boldsymbol{\eta_k}$ and the bias term η_k^0 of each event k by keeping the assignment of the variables $y_{t,i}$,$z_{t,i}$ and $s_{t,i}$ fixed.

For the Gibbs sampling part, we jointly sample $y_{t,i} = 0, z_{t,i} = a$ (topic tweet) and $y_{t,i} = 1, s_{t,i} = k$ (event tweet) as follows:

Topic tweet:

$$p(y_{t,i} = 0, z_{t,i} = a | \mathbf{y}_{\neg t,i}, \mathbf{z}_{\neg t,i}, \mathbf{w}, \mathbf{r}, \mathbf{c}, u_{t,i})$$
$$\propto \frac{n^{(\pi)}_{u,0} + \tau}{n^{(\pi)}_{u,(.)} + 2\tau} \frac{n^{(\theta)}_{u,a} + \gamma}{n^{(\theta)}_{u,(.)} + A\gamma} \frac{\prod_{v=1}^{V} \prod_{i=0}^{E_{(v)}-1} (n^{(\phi)}_{a,v} + i + \beta)}{\prod_{i=0}^{E_{(.)}-1} (n^{(\phi)}_{a,(.)} + i + V\beta)}$$
$$\prod_{t',i' \in I_u} \frac{\mathcal{N}(c_{t',i'} | \eta^0_{s_{t',i'}} + \boldsymbol{\eta}_{s_{t',i'}} \cdot \bar{\boldsymbol{z}}^*_u, \epsilon^{-1})}{\mathcal{N}(c_{t',i'} | \eta^0_{s_{t',i'}} + \boldsymbol{\eta}_{s_{t',i'}} \cdot \bar{\boldsymbol{z}}_u, \epsilon^{-1})}$$

Event tweet:

$$p(y_{t,i} = 1, s_{t,i} = k | \mathbf{y}_{\neg t,i}, \mathbf{z}_{\neg t,i}, \mathbf{w}, \mathbf{r}, \mathbf{c}, u_{t,i})$$
$$\propto \frac{n^{(\pi)}_{u,1} + \tau}{n^{(\pi)}_{u,(.)} + 2\tau} \frac{1}{\boldsymbol{N}} \Bigg(n^{\text{RCRP}}_{k,t} \mathcal{N}(c_{t,i} | \eta^0_{s_{t,i}} + \boldsymbol{\eta}_{s_{t,i}} \cdot \bar{\boldsymbol{z}}_u, \epsilon^{-1})$$
$$\cdot \exp(- \sum_{\substack{t'=1 \\ |t'-t|>1}}^{T} \lambda |t - t'| n_{k,t'}) \Bigg) \frac{\prod_{v=1}^{V} \prod_{i=0}^{E_{(v)}-1} (n^{(\psi)}_{k,v} + i + \beta)}{\prod_{i=0}^{E_{(.)}-1} (n^{(\psi)}_{k,(.)} + i + V\beta)}$$

in which,

$$n^{\text{RCRP}}_{k,t} = \begin{cases} (n_{k,t-1} + n_{k,t}) & \\ \quad \cdot \frac{n_{k,t} + n_{k,t+1}}{n_{k,t}} & \text{if } n_{k,t-1} > 0, n_{k,t} > 0, \\ n_{k,t-1} & \text{if } n_{k,t-1} > 0, n_{k,t} = 0, \\ n_{k,t+1} & \text{if } n_{k,t+1} > 0, n_{k,t} = 0, \\ \alpha & \text{if } k \text{ is a new event,} \end{cases}$$

where we use u to represent $u_{t,i}$. $n^{(\pi)}_{u,0}$ is the number of topic tweets by user u while $n^{(\pi)}_{u,1}$ is the number of event tweets by user u. They stem from integrating out the user's Bernoulli distribution π_u. $n^{(\pi)}_{u,(.)}$ is the total number of tweets by user u. Similarly, $n^{(\theta)}_{u,a}$ is the number of tweets assigned to topic a for this user, resulting from integrating out the user's topic distribution θ_u. $n^{(\theta)}_{u,(.)}$ is the same as $n^{(\pi)}_{u,0}$. $E_{(v)}$ is the number of times word type v appears in the current tweet, and $E_{(.)}$ is the total number of words in the current tweet. $n^{(\phi)}_{a,v}$ is the number of times word type v is assigned to topic a, and $n^{(\phi)}_{a,(.)}$ is the number of words assigned to topic a. $n^{(\psi)}_{k,v}$ is the number of times word type v is assigned to event k, and $n^{(\psi)}_{k,(.)}$ is the total number of words assigned to event k. These word counters stem form integrating out each event's word distribution and are set to zero when k is a new event. $I_u = \{t', i' | y_{t',i'} = 1, u_{t',i'} = u\}$, which is the set of event tweets published by user u, and u represents $u_{t,i}$ for short. $\bar{\boldsymbol{z}}^*_u$ is the empirical

counting vector which considers the current tweet's topic assignment, while $\bar{\boldsymbol{z}}_u$ and all other counters do not consider the current tweet. Finally, $\boldsymbol{N}$ is a local normalization factor for event tweets, which includes the RCRP, event-topic affinity and regularization on event duration.

With the previous Gibbs sampling step, we can get the assignment of variables $y_{t,i}$,$z_{t,i}$ and $s_{t,i}$. Given the assignment, we use gradient descent to update the values of the bias term η_k^0 and the event-topic affinity vectors $\boldsymbol{\eta}_k$ for each current existing event k. First, we can get the logarithm of the posterior distribution:

$$\begin{aligned}&\ln P(\mathbf{y}, \mathbf{z}, \mathbf{s}, \mathbf{r}, \mathbf{c}|\mathbf{w}, \mathbf{u}, \text{all priors})\\ &= \text{constant} - \sum_{k=1}^{\infty}\{\frac{\iota}{2}({\eta_k^0}^2 + \boldsymbol{\eta}_k \cdot \boldsymbol{\eta}_k)\\ &+ \sum_{u=1}^{U} n_{u,k}\frac{\epsilon}{2}[1 - (\eta_k^0 + \boldsymbol{\eta}_k \cdot \bar{\boldsymbol{z}}_u)]^2\},\end{aligned}$$

where $n_{u,k}$ is the number of event tweets about event k published by user u. The derivative of the logarithm of the posterior distribution with respect to the bias term η_k^0 and the event-topic affinity vector $\boldsymbol{\eta}_k$ are as follows:

$$\begin{aligned}\frac{\partial \ln P}{\partial \eta_k^0} &= -\iota\eta_k^0 + \sum_{u=1}^{U}\epsilon n_{u,k}[1 - (\eta_k^0 + \boldsymbol{\eta}_k \cdot \bar{\boldsymbol{z}}_u)],\\ \frac{\partial \ln P}{\partial \boldsymbol{\eta}_k} &= -\iota\boldsymbol{\eta}_k + \sum_{u=1}^{U}\epsilon n_{u,k}[1 - (\eta_k^0 + \boldsymbol{\eta}_k \cdot \bar{\boldsymbol{z}}_u)]\bar{\boldsymbol{z}}_u.\end{aligned}$$

4 Experiment

4.1 Dataset and Experiment Setup

We evaluate our model on a Twitter dataset that contains 500 users. These users are randomly selected from a much larger pool of around 150K users based in Singapore. Selecting users from the same country/city ensures that we find coherent and meaningful topics and events. We use tweets published between April 1 and June 30, 2012 for our experiments. For preprocessing, we use the CMU Twitter POS Tagger[1] to tag these tweets and remove those non-standard words (i.e. words tagged as punctuation marks, emoticons, urls, at-mentions, pronouns, etc.) and stop words. We also remove tweets with less than three words. After preprocessing, the dataset contains 655,881 tweets in total.

Recall that our model is designed to identify topics, events and their relations with users. We therefore would like to evaluate the quality of the identified topics and events as well as the usefulness of the discovered topic distributions of users and event-topic affinity vectors. Because our topic discovery mechanism is fairly standard and a quick inspection shows that the discovered topics are generally meaningful and comparable to those discovered by standard LDA, here we do not focus on evaluation of topics. In Section 4.2 we evaluate the quality of the discovered events. In Section 4.3 we show how the discovered event-topic affinity vectors can be useful.

For comparison, we consider an existing method called **TimeUserLDA** introduced in our previous work (Diao et al., 2012). TimeUserLDA also models topics and events by separating topic tweets from event tweets. However, it groups event tweets into a *fixed* number of *bursty topics* and then uses a two-state machine in a postprocessing step to identify events from these bursty topics. Thus, events are not directly modeled within the generative process itself. In contrast, events are inherent in our generative model. We do not compare with other event detection methods because our objective is not online event detection.

We also compare our final model with two degenerate versions of it. We refer to the base model described in Section 3.1 as **Base** and the model with the duration-based regularization as **Base+Reg**. Comparison with these two degenerate models allows us to assess the effect of the two modifications we propose. We refer to the final model with both the duration-based regularization and the event-topic affinity vectors as **Base+Reg+Aff**.

For the parameter setting, we empirically set A to 40, γ to $\frac{50}{A}$, τ to 1, β to 0.01, α to 1, ι to 10, ϵ to 1, and the duration regularization parameter λ to 0.01. When a new event k is created, the inner popularity bias term η_k^0 is set to 1, and the factors in event-topic affinity vectors $\boldsymbol{\eta}_k$ are all set to 0. We run the stochastic EM sampling scheme for 300 iterations. After Gibbs sampling assigns each variable a value at the end of each iteration, we update the values of η_k^0 and $\boldsymbol{\eta}_k$ for the existing events using gradient descent.

[1] http://www.ark.cs.cmu.edu/TweetNLP/

Event	Top words	Duration	Inner popularity (η_k^0)
debate caused by Manda Swaggie	singapore, bieber, europe, amanda, justin, trending, manda, hates, swaggie, hate	17 June - 19 June	0.9457
Indonesia tsunami	tsunami, earthquake, indonesia, singapore, hit, warning, aceh, 8.9, safe, magnitude	10 April - 12 April	0.9439
SJ encore concert	#ss4encore, cr, #ss4encoreday2, hyuk, 120526, super, leader, changmin, fans, teuk	26 May - 28 May	0.8360
Mother's Day	day, happy, mother's, mothers, love, mom, mum, everyday, mother, moms	11 May - 14 May	0.9370
April Fools' Day	april, fools, day, fool, joke, prank, happy, today, trans, fool's	1 April - 3 April	0.9322

Table 1: The top-5 events identified by Base+Reg+Aff. We show the story name which is manually labeled, top ten ranking words, lasting duration and the inner popularity (η_k^0) for each event.

4.2 Events

First we quantitatively evaluate the quality of the detected events. Our model finds clusters of tweets that represent events. We first assess whether these events are meaningful. We then judge whether the detected event tweets are indeed related to the corresponding event.

Quality of Top Events

Method	P@5	P@10	P@20	P@30
Base+Reg+Aff	**1.000**	**1.000**	**0.950**	**0.900**
Base+Reg	**1.000**	**1.000**	**0.950**	0.867
Base	0.000	0.200	0.250	0.367
TimeUserLDA	**1.000**	0.800	0.750	0.600

Table 2: Precision@K for the various methods.

Usually we are interested in the most popular events on Twitter. We therefore assess whether the top events are meaningful. For each method, we rank the detected events based on the number of tweets assigned to them and then pick the top-30 events for each method. We randomly mix these events and ask two human judges to label them. The judges are given 100 randomly selected tweets for each event (or all tweets if an event contains less than 100 tweets). The judges can use external sources to help them. If an event is meaningful based on the 100 sample tweets, a score of 1 is given. Otherwise it is scored 0. The inter-annotator agreement score is 0.744 using Cohen's kappa, showing substantial agreement. Finally we treat an event as meaningful if both judges have scored it 1.

Table 2 shows the performance in terms of precision@K, and Table 1 shows the top 5 events of our model (i.e., Base+Reg+Aff). We have the following findings from the results: (1) Our base model performs quite poorly for the top events while Base+Reg and Base+Reg+Aff perform much better. This shows that the duration-based regularization is critical in finding meaningful events. A close examination shows that the base model clusters many general topic tweets as events, such as tweets about transportation and music and even foursquare tweets. (2) TimeUserLDA performs well for the very top events (P@5 and P@10) but its performance drops for lower-ranked events (P@20 and P@30), similar to what was reported by Diao et al. (2012). A close examination shows that this method is good at finding major events that do not have strong topic association and thus attract most people's attention, e.g. earthquakes, but not good at finding topic-oriented events such as some concerts and sports games. This is because this method mixes topics and events first and only detects events from bursty topics in a second stage of postprocessing. In contrast, our model performs well for topic-oriented events. (3) The difference between Base+Reg and Base+Reg+Aff is small, suggesting that the event-topic affinity vectors are not crucial for event detection.

Precision of Event Tweets

Next, we evaluate the relevance of the detected event tweets to each event. To make a fair comparison, we select only the common events identified by all the methods. We pick 3 out of 5 common events shared by all methods within top-30 events

Event	TimeUserLDA	Base	Base+Reg	Base+Reg+Aff
Father's Day	0.61	0.63	0.71	**0.72**
debate caused by Manda Swaggie	0.73	0.74	**0.84**	0.80
Indonesia tsunami	0.75	0.75	**0.82**	0.80
Super Junior album release	N/A	0.72	0.78	**0.81**

Table 3: Precision of the event tweets for the 4 common events.

(we pick "Fathers' day" to represent public festivals, and ignore the similar events "Mothers' day" and "April fools"). We also pick one event shared by three RCRP based models. We further ask one of the judges to score the 100 tweets as either 1 or 0 based on their relevance to the event. The precision of the 100 tweets for each event and each method is shown in Table 3. We can see that again Base+Ref and Base+Ref+Aff perform similarly, and both outperform the other two methods. We also take a close look at the tweets and find that the false positives mislabeled by Base is mainly due to the long-duration of the discovered events. For example, for the event "Super Junior album release," Base finds other music-related tweets surrounding the peak period of the event itself.

In summary, our evaluation on event quality shows that (1) Using the non-parametric RCRP model to identify events within the generative model itself is advantageous over TimeUserLDA, which identifies events by postprocessing. (2) The duration-based regularization is crucial for finding more meaningful events.

4.3 Event-Topic Association

Besides event identification, our model also finds the association between events and topics through the event-topic affinity vectors. The discovered event-topic association can potentially be used for various tasks. Here we conduct two experiments to demonstrate its usefulness.

Event Recommendation

Recall that to discover event-topic association, we treat an event as an item and a tweet about the event as indication of the user's adoption of the item. Following this analogy with item recommendation, we define an event recommendation task where the goal is to recommend an event to users who have not posted any tweet about the event but may potentially be interested in the event. Intuitively, if a user's topic distribution is similar to the event-topic affinity vector of the event, then the user is likely to be interested in the event.

Specifically, we use the first two months' data (April and May 2012) as training data to learn all the users' topic distributions. We then use a ransom subset of 250 training users and their tweets in June to identify events in June as well as the event-topic affinity vectors of these events. We pick 8 meaningful events that are ranked high by all methods for testing. For each event, we try to find among the remaining 250 users those who may be interested in the event and compare the results with ground truth obtained by human judgment. Because it is time consuming to obtain the ground truth for all 250 users, we randomly pick 100 of these 250 users for testing purpose. For each test user and each event, we manually inspect the user's tweets around the peak days of the event to judge whether she has commented on the event. This is used as ground truth.

With our complete model Base+Reg+Aff, we can simply rank the 100 test users in decreasing order of $\boldsymbol{\eta}_k \cdot \bar{\boldsymbol{z}}_u$. For the other methods, because we do not have any parameter that directly encodes event-topic association, we cannot rank users based on how similar their topic distributions are to the event's affinity to topics. We instead adopt a collaborative filtering strategy and rank the test users by their similarity with those training users who have tweeted about the event. Specifically, each of these methods produces a topic distribution θ_u for each user. In addition, for each test event these methods identify a list of training users who have tweeted on it. By taking the average topic distribution of these training users and compute its cosine similarity with a test user's topic distribution, we can rank the 100 test users.

Since we have turned the recommendation task into a ranking task, we use Average Precision, a commonly used metric in information retrieval, to compare the performance. Average Precision is the

Event	TimeUserLDA	Base	Base+Reg	Base+Reg+Aff	Inner popularity (η_k^0)
debate caused by Manda Swaggie	0.3533	0.3230	**0.3622**	0.2956	0.943
Father's Day	0.3811	0.3525	0.3596	**0.4362**	0.917
Big Bang album release	0.1406	0.1854	0.1533	**0.1902**	0.893
City Harvest Church scandal	N/A	0.2832	0.1874	**0.3347**	0.890
Alex Ong pushing an old lady	N/A	**0.1540**	0.1539	0.1113	0.876
final episode of Super Spontan (reality show)	N/A	0.0177	0.0331	**0.2900**	0.862
Super Junior album release	N/A	0.0398	0.0330	**0.5900**	0.792
LionsXII 9-0 Sabah FA (soccer)	0.0711	0.1207	0.2385	**0.3220**	0.773
MAP	N/A	0.1845	0.1901	**0.3213**	

Table 4: For the 8 test events that happened in June 2012, we compute the Average Precision for each event. We also show the Mean Average Precision (MAP) when applicable.

Topic	Top words of the topic	Related event	Top words of the event
Food	eat, food, eating, ice, hungry, dinner, cream, lunch, chicken, buy	Ben&Jerry free cone day	free, cone, day, ben, jerry's, today, b&j, zoo, #freeconeday, singapore
Korean Music	music, big, cr, super, bang, junior, love, concert, bank, album	Super Junior encore concert	#ss4encore, cr, #ss4encoreday2, hyuk, 120526, super, leader, changmin, fans, teuk
		Super Junior Shanghai concert	#ss4shanghai, cr, 120414, donghae, eunhyuk, giraffe, solo, hyuk, ryeowook, shanghai
		Super Junior Paris concert	#ss4paris, cr, paris, super, 120406, ss4, junior, siwon, show, update
Malay	aku, nak, tak, kau, ni, lah, tk, je, mcm, nk	final episode of Super Spontan	zizan, johan, friendship, jozan, #superspontan, skips, forever, real, juara, gonna
		LionsXII 9-0 Sabah FA	sabah, 9-0, #lionsxii, lions, singapore, 7-0, amet, sucks, sabar, goal
Soccer	win, game, man, chelsea, match, city, goal, good, united, team	Man City crowned English champions	man, city, united, qpr, fuck, bored, lah, love, glory, update

Table 5: Example topics and their corresponding correlated events.

average of the precision value obtained for the set of top items existing after each relevant item is retrieved (Manning et al., 2008). We also rank the 8 events in decreasing order of their inner popularity η_k^0 learned by our complete model. The results are shown in Table 4. We have the following findings from the table. (1) Our complete method outperforms the other methods for 6 out of the 8 test events, suggesting that with the inferred event-topic affinity vectors we can do better event recommendation. (2) The improvement brought by the event-topic affinity vectors, as reflected in the difference in Average Precision between Base+Reg+Aff and Base (or Base+Reg) is more pronounced for events with lower inner popularity. Recall that the inner popularity of an event shows the inherent popularity of an event regardless of its association with any topic, that is, an event with high inner popularity attracts attention of many people regardless of their topic interests, while an event with low inner popularity tends to attract attention of certain people with similar topic interests. The finding above suggests that the event-topic affinity vectors are especially useful for recommending events that attract only certain people's attention, such as those related to sports, music, etc.

One may wonder for the events with low inner popularity why we could not achieve the same effect by Base or Base+Reg where we consider the topic similarity of test users with training users who have tweeted about the event. Our close examination shows that for these events although Base and Base+Reg may identify relevant event tweets with decent precision, the users they identify who have

tweeted about the event may not share similar topic interests. As a result, when we average these users' topic interests, we cannot obtain a clear skewed topic distribution that explains the event's affinity to different topics. In contrast, Base+Reg+Aff explicitly models the event-topic affinity vector and prefers to assign a tweet to an event if its author's topic distribution is similar to the event's affinity vector. Through the training iterations, the users who have tweeted about an event as identified by Base+Reg+Aff will gradually converge to share similar topic distributions.

Grouping Events by Topics

Finally, we show that the event-topic affinity vectors can also be used to group events by topics. This can potentially be used to better organize and present popular events in social media. In Table 5 we show a few highly related events for a few popular topics in our Twitter data set. Specifically given a topic a we rank the meaningful events that contain at least 70 tweets based on $\eta_{k,a}$. We can see from the table that the events are indeed related to the corresponding topic. The event "LionsXII 9-0 Sabah FA" is particularly interesting in that it is highly related to both the topic on Malay and the topic on soccer. (LionsXII is a soccer team from Singapore and Sabah FA is a soccer team from Malaysia.)

5 Conclusion

In this paper, we propose a unified model to study topics, events and users jointly. The base of our method is a combination of an LDA-like model and the Recurrent Chinese Restaurant Process, which aims to model users' longstanding personal topic interests and events over time simultaneously. The Recurrent Chinese Restaurant Process is appealing in the sense that it provides a principled dynamic nonparametric model in which the number of events is not fixed overtime. We further use a time duration-based regularization to capture the fast emergence and disappearance of events on Twitter, which is effective to produce more meaningful events. Finally, we use an inner popularity bias parameter and event-topic affinity vectors to interpret an event's inherent popularity and its affinity to different topics. Our experiments quantitatively show that our proposed model can effectively identify meaningful events and accurately find relevant tweets for these events. Furthermore, the event-topic association inferred by our model can help an event recommendation task and organize events by topics.

6 Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. We thank the reviewers for their valuable comments.

References

Deepak Agarwal and Bee-Chung Chen. 2010. fLDA: matrix factorization through latent Dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100.

Amr Ahmed and Eric P. Xing. Dynamic non-parametric mixture models and the recurrent Chinese restaurant process: with applications to evolutionary clustering. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008*.

Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J. Smola, and Choon Hui Teo. 2011. Unified analysis of streaming news. In *In Proceedings of the 20th international conference on World wide web*, pages 267 – 276.

Ramnath Balasubramanyan and William W. Cohen. 2013. Regularization of latent variable models to obtain sparsity. In *Proceedings of SIAM Conference on Data Mining*.

Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond trending topics: Real-world event identification on twitter. In *Fifth International AAAI Conference on Weblogs and Social Media*.

D. Blackwell and J. MacQueen. 1973. Ferguson distributions via Polya urn schemes. *The Annals of Statistics*, pages 353–355.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113 – 120.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. 2012. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 536 – 544.

Liangjie Hong and Brian D. Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, pages 80–88.

Liangjie Hong, Byron Dom, Siva Gurumurthy, and Kostas Tsioutsiouliklis. 2011. A time-dependent topic model for multiple text streams. In *Proceedings of SIGKDD*.

Jey Han Lau, Nigel Collier, and Timothy Baldwin. 2012. On-line trend analysis with topic models: #twitter trends detection topic model online. In *Proceedings of the 24th International Conference of on Computational Linguistics*, pages 1519–1534.

Chenliang Li, Aixin Sun, and Anwitaman Datta. 2012. Twevent: segment-based event detection from tweets. In *In Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 155 – 164.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze, 2008. *Introduction to Information Retrieval*, chapter Evaluation in information retrieval. Cambridge University Press.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to Twitter. In *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181 – 189.

Saša Petrović, Miles Osborne, Richard McCreadie, Richard Macdonald, Richard Ounis, and Luke Shrimpton. 2013. Can twitter replace newswire for breaking news? In *Proceedings of the 7th International Conference on Weblogs and Social Media*.

Daniel Ramage, Susan T. Dumais, and Daniel J. Liebling. 2010. Characterizing microblogs with topic models. In *Proceedings of the 4th International Conference on Weblogs and Social Media*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, pages 851–860.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20.

Xuning Tang and Christopher C. Yang. 2012. TUT: a statistical model for detecting trends, topics and user interests in social media. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 972 – 981.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448 – 456.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.*, pages 424 – 433.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Proceedings of the 13th acm sigkdd international conference on knowledge discovery and data mining. In *Proceedings of SIGKDD*, pages 784 – 793.

Jianshu Weng and Bu-Sung Lee. 2011. Event detection in Twitter. In *Proceedings of the Fifth International Conference on Weblogs and Social Media*.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing Twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on IR Research*, pages 338–349.

Authorship Attribution of Micro-Messages

Roy Schwartz Oren Tsur Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
{roys02|oren|arir}@cs.huji.ac.il

Moshe Koppel
Department of Computer Science
Bar Ilan University
koppel@macs.biu.ac.il

Abstract

Work on authorship attribution has traditionally focused on long texts. In this work, we tackle the question of whether the author of a very short text can be successfully identified. We use Twitter as an experimental testbed. We introduce the concept of an author's unique "signature", and show that such signatures are typical of many authors when writing very short texts. We also present a new authorship attribution feature ("flexible patterns") and demonstrate a significant improvement over our baselines. Our results show that the author of a single tweet can be identified with good accuracy in an array of flavors of the authorship attribution task.

1 Introduction

Research in authorship attribution has developed substantially over the last decade (Stamatatos, 2009). The vast majority of such research has been dedicated towards finding the author of long texts, ranging from single passages to book chapters. In recent years, the growing popularity of social media has created special interest, both theoretical and computational, in short texts. This has led to many recent authorship attribution projects that experimented with web data such as emails (Abbasi and Chen, 2008), web forum messages (Solorio et al., 2011) and blogs (Koppel et al., 2011b). This paper addresses the question to what extent the authors of very short texts can be identified. To answer this question, we experiment with Twitter tweets.

Twitter messages (tweets) are limited to 140 characters. This restriction imposes major difficulties on authorship attribution systems, since authorship attribution methods that work well on long texts are often not as useful when applied to short texts (Burrows, 2002; Sanderson and Guenter, 2006).

Nonetheless, tweets are relatively self-contained and have smaller sentence length variance compared to excerpts from longer texts (see Section 3). These characteristics make Twitter data appealing as a testbed when focusing on short texts. Moreover, an authorship attribution system of tweets may have various applications. Specifically, a range of cybercrimes can be addressed using such a system, including identity fraud and phishing.

In this paper, we introduce the concept of k-*signatures*. We denote the k-*signatures* of an author a as the features that appear in at least $k\%$ of a's training samples, while not appearing in the training set of any other author. When k is large, such signatures capture a unique style used by a. An analysis of our training set reveals that unique k-signatures are typical of many authors. Moreover, a substantial portion of the tweets in our training set contain at least one such signature. These findings suggest that a single tweet, although short and sparse, often contains sufficient information for identifying its author. Our results show that this is indeed the case.

We train an SVM classifier with a set of features that include character n-grams and word n-grams. We use a rigorous experimental setup, with varying number of authors (values between 50-1,000) and various sizes of the training set, ranging from 50 to 1,000 tweets per author. In all our experiments, a single tweet is used as test document. We also use a setting in which the system is allowed to respond *don't know* in cases of uncertainty. Applying this option results in higher precision, at the expense of

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1880–1891,
Seattle, Washington, USA, 18-21 October 2013.

lower recall.

Our results show that the author of a tweet can be successfully identified. For example, when using a dataset of as many as 1,000 authors with 200 training tweets per author, we are able to obtain 30.3% accuracy (as opposed to a random baseline of only 0.1%). Using a dataset of 50 authors with as few as 50 training tweets per author, we obtain 50.7% accuracy. Using a dataset of 50 authors with 1,000 training tweets per author, our results reach as high as 71.2% in the standard classification setting, and exceed 91% accuracy with 60% recall in the *don't know* setting.

We also apply a new set of features, never previously used for this task – flexible patterns. Flexible patterns essentially capture the context in which function words are used. The effectiveness of function words as authorship attribution features (Koppel et al., 2009) suggests using flexible pattern features. The fact that flexible patterns are learned from plain text in a fully unsupervised manner makes them domain and language independent. We demonstrate that using flexible patterns gives significant improvement over our baseline system. Furthermore, using flexible patterns, our system obtains a 6.1% improvement over current state-of-the-art results in authorship attribution on Twitter.

To summarize, the contribution of this paper is threefold.

- We provide the most extensive research to date on authorship attribution of micro-messages, and show that authors of very short texts can be successfully identified.
- We introduce the concept of an author's unique k-signature, and demonstrate that such signatures are used by many authors in their writing of micro-messages.
- We present a new feature for authorship attribution – flexible patterns – and show its significant added value over other methods. Using this feature, our system obtains a 6.1% improvement over the current state-of-the-art.

The rest of the paper is organized as follows. Sections 2 and 3 describe our methods and our experimental testbed (Twitter). Section 4 presents the concept of k-signatures. Sections 5 and 6 present our experiments and results. Flexible patterns are presented in Section 7 and related work is presented in Section 8.

2 Methodology

In the following we briefly describe the main features employed by our system. The features below are binary features.

Character n-grams. Character n-gram features are especially useful for authorship attribution on micro-messages since they are relatively tolerant to typos and non-standard use of punctuation (Stamatatos, 2009). These are common in the non-formal style generally applied in social media services. Consider the example of misspelling "Britney" as "Brittney". The misspelled name shares the 4-grams "Brit" and "tney" with the correct name. As a result, these features provide information about the author's style (or at least her topic of interest), which is not available through lexical features.

Following standard practice, we use 4-grams (Sanderson and Guenter, 2006; Layton et al., 2010; Koppel et al., 2011b). White spaces are considered characters (i.e., a character n-gram may be composed of letters from two different words). A single white-space is appended to the beginning and the end of each tweet. For efficiency, we consider only character n-gram features that appear at least t_{cng} times in the training set of at least one author (see Section 5).

Word n-grams. We hypothesize that word n-gram features would be useful for authorship attribution on micro-messages. We assume that under a strict length restriction, many authors would prefer using short, repeating phrases (word n-grams).

In our experiments, we consider $2 \leq n \leq 5$.[1] We regard sequences of punctuation marks as words. Two special words are added to each tweet to indicate the beginning and the end of the tweet. For efficiency, we consider only word n-gram features that appear at least t_{wng} times in the training set of at least one author (see Section 5).

Model. We use libsvm's Matlab implementation of a multi-class SVM classifier with a linear kernel

[1] We skip unigrams as they are generally captured by the character n-gram features.

(Chang and Lin, 2011). We use ten-fold cross validation on the training set to select the best regularization factor between 0.5 and 0.005.[2]

3 Experimental Testbed

Our main research question in this paper is to determine the extent to which authors of very short texts can be identified. A major issue in working with short texts is selecting the right dataset. One approach is breaking longer texts into shorter chunks (Sanderson and Guenter, 2006). We take a different approach and experiment with micro-messages (specifically, tweets).

Tweets have several properties making them an ideal testbed for authorship attribution of short texts. First, tweets are posted as single units and do not necessarily refer to each other. As a result, they tend to be self contained. Second, tweets have more standardized length distribution compared to other types of web data. We compared the mean and standard deviation of sentence length in our Twitter dataset and in a corpus of English web data (Ferraresi et al., 2008).[3] We found that (a) tweets are shorter than standard web data (14.2 words compared to 20.9), and (b) the standard deviation of the length of tweets is much smaller (6.4 vs. 21.4).

Pre-Processing. We use a Twitter corpus that includes approximately 5×10^8 tweets.[4] All non-English tweets and tweets that contain fewer than 3 words are removed from the dataset. We also remove tweets marked as retweets (using the RT sign, a standard Twitter symbol to indicate that this tweet was written by a different user). As some users retweet without using the RT sign, we also remove tweets that are an exact copy of an existing tweet posted in the previous seven days.

Apart from plain text, some tweets contain references to other Twitter users (in the format of @<user>). Since using reference information makes this task substantially easier (Layton et al., 2010), we replace each user reference with the special meta tag REF. For sparsity reasons, we also replace web addresses with the meta tag URL, numbers with the meta tag NUM, time of day with the meta tag TIME and dates with the meta tag DATE.

[2] In practice, 0.05 or 0.1 are selected in almost all cases.

[3] `http://wacky.sslmit.unibo.it`

[4] These comprise ∼15% of all public tweets created from May 2009 to March 2010.

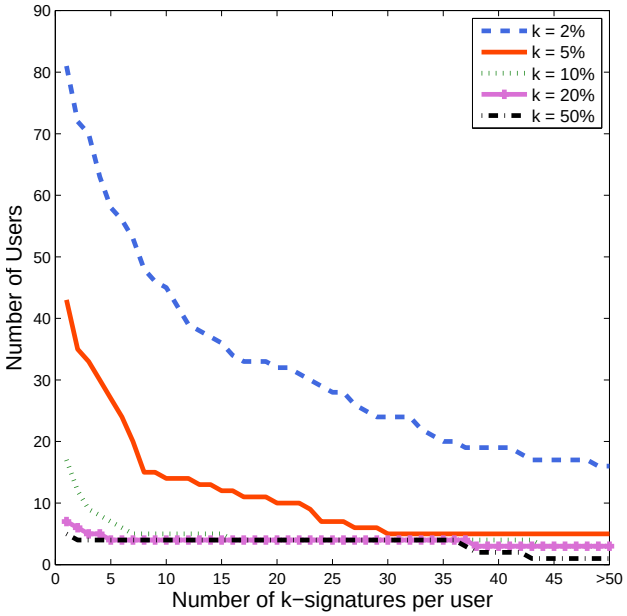


Figure 1: Number of users with at least x k-signatures (100 authors, 180 training tweets per author).

4 k-Signatures

In this section, we show that many authors adopt a unique style when writing micro-messages. This style can be detected by a strong classification algorithm (such as SVM), and be sufficient to correctly identify the author of a single tweet.

We define the concept of the *k-signature* of an author a to be a feature that appears in at least $k\%$ of a's training set, while not appearing in the training set of any other user. Such signatures can be useful for identifying future (unlabeled) tweets written by a.

To validate our hypothesis, we use a dataset of 100 authors with 180 tweets per author. We compute the number of k-signatures used by each of the authors in our dataset. Figure 1 shows our results for a range of k values (2%, 5%, 10%, 20% and 50%). Results demonstrate that 81 users use at least one 2%-signature, 43 users use at least one 5%-signature, and 17 users use at least one 10%-signature. These results indicate that a large portion of the users adopt a unique signature (or set of signatures) when writing short texts. Table 1 provides examples of 10%-signatures.

Signature Type	10%-signature	Examples
Character n-grams	‘ ^_^’	REF oh ok ^_^ Glad you found it!
		Hope everyone is having a good afternoon ^_^
		REF Smirnoff lol keeping the goose in the freezer ^_^
	‘**yew** ’	gurl **yew** serving me tea nooch
		REF about wen **yew** and ronnie see each other
		REF lol so **yew** goin to check out tini’s tonight huh???
Word n-grams	**.. lal**	REF aww those are cool where u get those.. how do ppl react**.. lal**
		Ludas album is gone be hott**.. lal**
		Dayum refs don’t get injury timeouts**.. lal**.. get him off the field..
	smoochies , e3	I’m just back after takin’ a very long, icy cold shower.....Shivering **smoochies,E3** http://bit.ly/4CzzP9
		A blue stout or two would be nice as well, Purr!Blue smooth **smoochies,E3** http://bit.ly/75D4fO
		That is sooooooooooooooooo unfair!Double **smoochies,E3** http://bit.ly/07sXRGX

Table 1: Examples of 10%-signatures.

Results also show that seven users use one or more 20%-signatures, and five users even use one or more 50%-signatures. Looking carefully at these users, we find that they write very structured messages, and are probably bots, such as news feeds, bidding systems, etc. Table 2 provides examples of tweets posted by such users.[5]

Another interesting question is how many tweets contain at least one k-signature. Figure 2 shows for each user the number of tweets in her training set for which at least one k-signature is found. Results demonstrate that a total of 18.6% of the training tweets contain at least one 2%-signature, 10.3% the training tweets contain at least one 5%-signature and 6.5% of the training tweets contain at least one 10%-signature. These findings validate our assumption that many users use k-signatures in short texts.

These findings also have direct implications on authorship attribution of micro-messages, since k-signatures are reliable classification features. As a result, texts written by authors that tend to use k-signatures are likely to be easily identified by a reasonable classification algorithm. Consequently, k-signatures provide a possible explanation for the high quality results presented in this paper.

In the broader context, the presence (and contribution) of k-signatures is in line with the hypothesis proposed by (Davidov et al., 2010a): while still using an informal and unstructured (grammatical) language, authors tend to use typical and unique structures in order to allow a short message to stand alone without a clear conversational context.

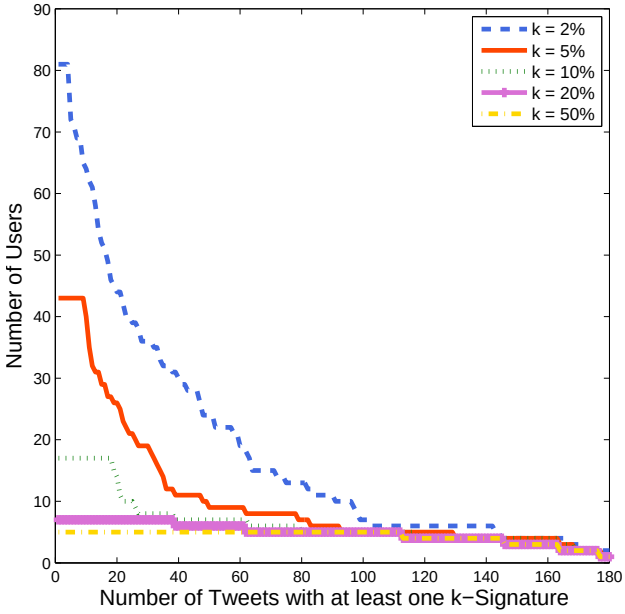


Figure 2: Number of users with at least x training tweets that contain at least one k-signature (100 authors, 180 training tweets per author).

[5]Our k-signature method can actually be useful for automatically identifying such users. We defer this to future work.

User	20%-signature	Examples
1	**I'm listening to :**	**I'm listening to:** Sigur R?s ? Intro: http://www.last.fm/music/Sigur+R%C3%B3s http://bit.ly/3XJHyb
		I'm listening to: Tina Arena ? In Command: http://www.last.fm/music/Tina+Arena http://bit.ly/7q9E25
		I'm listening to: Midnight Oil ? Under the Overpass: http://www.last.fm/music/Midnight+Oil http://bit.ly/7IH4cg
2	**news now (str)**	#Hotel **News Now(STR)** 5 things to know: 27 May 2009: From the desks of the HotelNewsNow.com editor... http://bit.ly/aZTZOq #Tourism #Lodging
		#Hotel **News Now(STR)** Five sales renegotiating tactics: As bookings representatives press to reneg... http://bit.ly/bHPn2L
		#Hotel **News Now(STR)** Risk of hotel recession retreats: The Hotel Industry's Pulse Index increases... http://bit.ly/a8EKrm #Tourism #Lodging
3	**(NUM bids) end date :**	NEW PINK NINTENDO DS LITE CONSOLE WITH 21 GIFTS + CASE: £66.50 **(13 Bids) End Date:** Tuesday Dec-08-2009 17:.. http://bit.ly/7uPt6V
		Microsoft Xbox 360 Game System - Console Only - Working: US $51.99 **(25 Bids) End Date:** Saturday Dec-12-2009 13:.. http://bit.ly/8VgdTv
		Microsoft Sony Playstation 3 (80 GB) Console 6 Months Old: £190.00 **(25 Bids) End Date:** Sunday Dec-13-2009 21:21:39 G.. http://bit.ly/7kwtDS

Table 2: Examples of tweets published by very structured users, suspected to be bots, along with one of their 20%-signatures.

5 Experiments

We report of three different experimental configurations. In the experiments described below, each dataset is divided into training and test sets using ten-fold cross validation. On the test phase, each document contains a single tweet.

Experimenting with varying Training Set Sizes. In order to test the affect of the training set size, we experiment with an increasingly larger number of tweets per author. Experimenting with a range of training set sizes serves two purposes: (a) to check whether the author of a tweet can be identified using a very small number of (short) training samples, and (b) check how much our system can benefit from training on a larger corpus.

In our experiments we only consider users who posted between 1,000–2,000 tweets[6] (a total of 10,183 users), and randomly select 1,000 tweets per user. From these users, we select 10 groups of 50 users each.[7] We perform a set of classification experiments, selecting for each author an increasingly larger subset of her 1,000 tweets as training set. Subset sizes are (50, 100, 200, 500, 1,000). Threshold values for our features in each setting (see Section 2) are $(2, 2, 4, 10, 20)$ for t_{cng} and $(2, 2, 2, 3, 5)$ for t_{wng}, respectively.

Experimenting with varying Numbers of Authors. In a second set of experiments, we use an increasingly larger number of authors (values between 100-1,000), in order to check whether the author of a very short text can be identified in a "needle in a haystack" type of setting.

Due to complexity issues, we only experiment with 200 tweets per author as training set. We select groups of size 100, 200, 500 and 1,000 users (one group per size). We use the same threshold values as the 200 tweets per author setting previously described (t_{cng} = 4, t_{wng} = 2).

[6] This range is selected since on one hand we want at least 1,000 tweets per author for our experiments, and on the other hand we noticed that users with a larger number of tweets in corpus tend to be spammers or bots that are very easy to identify, so we limit this number to 2,000.

[7] An eleventh group is selected as development set.

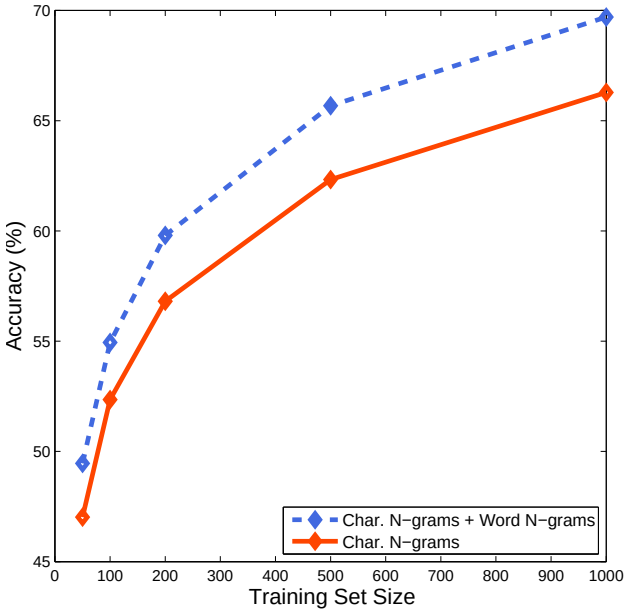


Figure 3: Authorship attribution accuracy for 50 authors with various training set sizes. The values are averaged over 10 groups. The random baseline is 2%.

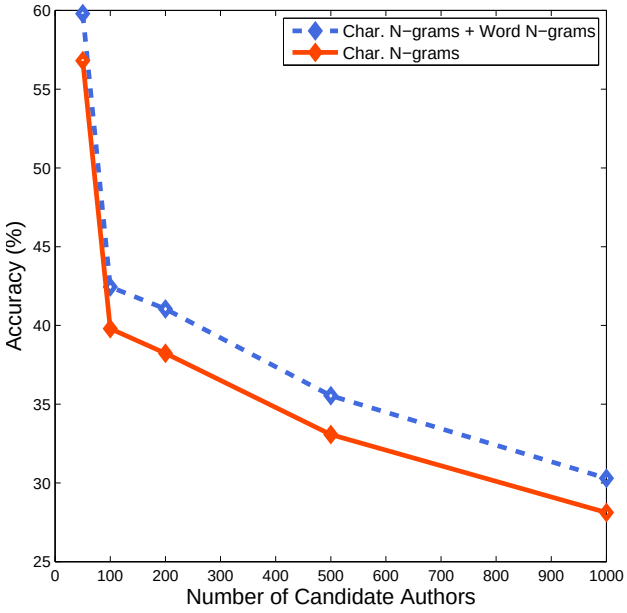


Figure 4: Authorship attribution accuracy with varying number of candidate authors, using 200 training tweets per author. The random baselines for 50[9], 100, 200, 500 and 1,000 authors are 2%, 1%, 0.5%, 0.2% and 0.1%, respectively.

Recall-Precision Tradeoff. Another aspect of our research question is the level of certainty our system has when suggesting an author for a given tweet. In cases of uncertainty, many real life applications would prefer not to get any response instead of getting a response with low certainty. Moreover, in real life applications we are often not even sure that the real author is part of our training set. Consequently, we allow our system to respond "*don't know*" in cases of low confidence (Koppel et al., 2006; Koppel et al., 2011b). This allows our system to obtain higher precision, at the expense of lower recall.

To implement this feature, we use SVM's probability estimates, as implemented in libsvm. These estimates give a score to each potential author. These scores reflect the probability that this author is the correct author, as decided by the prediction model. The selected author is always the one with the highest probability estimate.

As selection criterion, we use a set of increasingly larger thresholds (0.05-0.9) for the probability of the selected author. This means that we do not select test samples for which the selected author has a probability estimate value lower than the threshold.

6 Basic Results

Experimenting with varying Training Set Sizes. Figure 3 shows results for our experiments with 50 authors and various training set sizes. Results demonstrate that authors of very short texts can be successfully identified, even with as few as 50 tweets per author (49.5%). When given more training samples, authors are identified much more accurately (up to 69.7%). Results also show that, according to our hypothesis, word n-gram features substantially improve over character n-grams features only (3% averaged improvement over all settings).

Experimenting with varying Numbers of Authors. Figure 4 shows our results for various numbers of authors, using 200 tweets per author as training set. Results demonstrate that authors of an unknown tweet can be identified to a large extent even when there are as many as 1,000 candidate authors (30.3%, as opposed to a random baseline of only 0.1%). Results further validate that word n-gram features substantially improve over character

[9]Results for 50 authors with 200 tweets per author are taken from Figure 3.

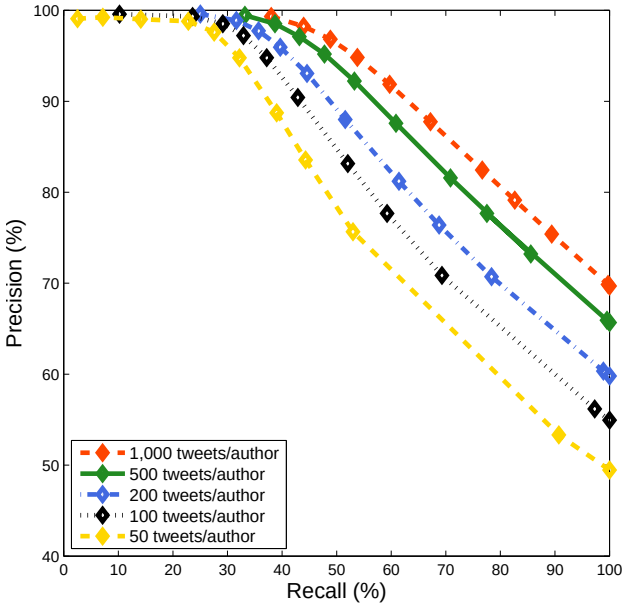


Figure 5: Recall-precision curves for 50 authors with varying training set sizes.

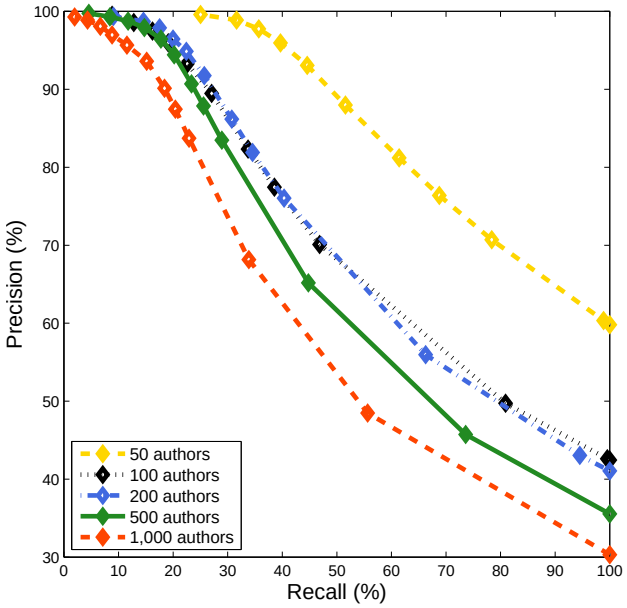


Figure 6: Recall-precision curves for varying number of authors.

n-grams features (2.6% averaged improvement).

Recall-Precision Tradeoff. Figure 5 shows the recall-precision curves for our experiments with 50 authors and varying training set sizes. Results demonstrate that we are able to obtain very high precision (over 90%) while still maintaining a relatively high recall (from ∼35% recall for 50 tweets per author up to $> 60\%$ recall for 1,000 tweets per author).

Figure 6 shows the recall-precision curves for our experiments with varying number of authors. Results demonstrate that even in the 1,000 authors setting, we are able to obtain high precision values (90% and 70%) with reasonable recall values (18% and ∼30%, respectively).

7 Flexible Patterns

In previous sections we provided strong evidence that authors of micro-messages can be successfully identified using standard methods. In this section we present a new feature, never previously used for this task – flexible patterns. We show that flexible patterns can be used to improve classification results.

Flexible patterns are a generalization of word n-grams, in the sense that they capture potentially unseen word n-grams. As a result, flexible patterns can pick up fine-grained differences between authors' styles. Unlike other types of pattern features, flexible patterns are computed automatically from plain text. As such, they can be applied to various tasks, independently of domain and language. We describe them in detail.

Word Frequency. Flexible patterns are composed of high frequency words (HFW) and content words (CW). Every word in the corpus is defined as either HFW or CW. This clustering is performed by counting the number of times each word appears in the corpus of size s. A word that appears more than $10^{-4} \times s$ times in a corpus is considered HFW. A word that appears less than $10^{-3} \times s$ times in a corpus is considered CW. Some words may serve both as HFWs and CWs (see Davidov and Rappoport (2008b) for discussion).

Structure of a Flexible Pattern. Flexible patterns start and end with an HFW. A sequence of zero or more CWs separates consecutive HFWs. At least one CW must appear in every pattern.[10] For efficiency, at most six HFWs (and as a result, five CW sequences) may appear in a flexible pattern. Examples of flexible patterns include

1. "the_{HFW} CW of_{HFW} the_{HFW}"

[10] Omitting this treats word n-grams as flexible patterns.

Flexible Pattern Features. Flexible patterns can serve as binary classification features; a tweet matches a given flexible pattern if it contains the flexible pattern sequence. For example, (1) is matched by (2).

2. "*Go to* $\underline{the_{HFW}\ house_{CW}\ of_{HFW}\ the_{HFW}}$ *rising sun*"

Partial Flexible Patterns. A flexible pattern may appear in a given tweet with additional words not originally found in the flexible pattern, and/or with only a subset of the HFWs (Davidov et al., 2010a). For example, (3) is a partial match of (1), since the word "great" is not part of the original flexible pattern. Similarly, (4) is another partial match of (1), since (a) the word "good" is not part of the original flexible pattern and (b) the second occurrence of the word "the" does not appear in (4) (missing word is marked by ▬).

3. "$\underline{The_{HFW}\ \mathbf{great}_{\mathbf{HFW}}\ king_{CW}\ of_{HFW}\ the_{HFW}}$ *ring*"

4. "$\underline{The_{HFW}\ \mathbf{good}_{\mathbf{HFW}}\ king_{CW}\ of_{HFW}\ \blacksquare}$ *Spain*"

We use such cases as features with lower weight, proportional to the number of found HFWs in the tweet ($w = \frac{0.5 \times n_{found}}{n_{expected}}$). For example, (1) receives a weight of 1 (complete match) against (2). Against (3), it receives a weight of 0.5 ($= \frac{0.5 \times 3}{3}$, partial match with no missing HFWs). Against (4) it receives a weight of $1/3$ ($= \frac{0.5 \times 2}{3}$, partial match with only 2/3 HFWs found).

Experimenting with Flexible Pattern Features. We repeat our experiments with varying training set sizes (see Section 5) with two more systems: one that uses character n-grams and flexible pattern features, and another that uses character n-grams, word n-grams and flexible patterns. High frequency word counts are computed separately for each author using her training set. We only consider flexible pattern features that appear at least t_{fp} times in the training set of at least one author. Values of t_{fp} for training set sizes (50, 100, 200, 500, 1,000) are (2, 3, 7, 7, 8), respectively.

Results. Figure 7 shows our results. Results demonstrate that flexible pattern features have an added value over both character n-grams alone (averaged 2.9% improvement) and over character n-grams and word n-grams together (averaged 1.5% improvement). We perform t-tests on each of our training set sizes to check whether the latter improvement is significant. Results demonstrate that it is highly significant in all settings, with p-values smaller than values between 10^{-3} (for 50 tweets per author) and 10^{-8} (1,000 tweets per author).

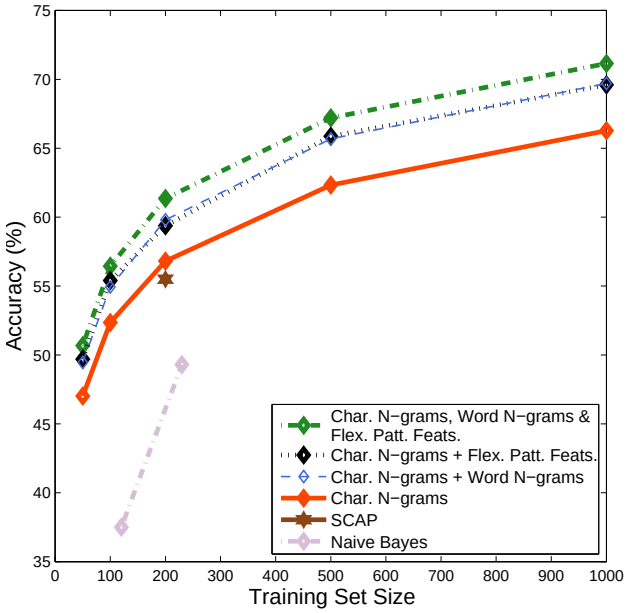


Figure 7: Authorship attribution accuracy for 50 authors with various training set sizes and various feature sets. The values are averaged over 10 groups. The random baseline is 2%.
Comparison to previous work: SCAP – SCAP algorithm results, as reported by (Layton et al., 2010), Naive Bayes – Naive Bayes algorithm results, as reported by (Boutwell, 2011).

Comparison to Previous Works. Figure 7 also shows results for the only two works that experimented in some of the settings we experimented in: Layton et al. (2010) and Boutwell (2011) (see Section 8). Our system substantially outperforms these two systems, by margins of 5.9% to 19%. These margins are explained by the choice of algorithm (SVM and not SCAP/naive Bayes) and our set of features (character n-grams + word n-grams + flexible patterns compared to character n-grams only). In order to rule out the possibility that these margins stem from using different datasets, we tested our system on the dataset used in (Layton et al., 2010). Our system obtains even higher results on this dataset than on our datasets (61.6%, a total im-

provement of 6.1% over (Layton et al., 2010)).

Discussion. To illustrate the additional contribution of flexible patterns over word n-grams, consider the following tweets, written by the same author.

5. "... $\underline{the_{HFW}\ way_{CW}\ I_{HFW}}$ *treated*$_{CW}$ *her*$_{HFW}$"
6. "... *half of* $\underline{the_{HFW}\ things_{CW}\ I_{HFW}}$ *have seen*"
7. "... $\underline{the_{HFW}\ friends_{CW}\ I_{HFW}}$ *have had for years"*
8. "... *in* $\underline{the_{HFW}\ neighborhood_{CW}\ I_{HFW}}$ *grew up in*"

Consider a case where (5) is part of the test set, while (6-8) appear in the training set. As (5) shares no sequence of words with (6-8), no word n-gram feature is able to identify the author's style in (5). However, this style can be successfully identified using the flexible pattern (9), shared by (5-8).

9. the_{HFW} CW I_{HFW}

This demonstrates the added value flexible pattern features have over word n-gram features.

8 Related Work

Authorship attribution dates back to the end of 19th century, when (Mendenhall, 1887) applied sentence length and word length features to plays of Shakespeare. Ever since, many methods have been developed for this task. For recent surveys, see (Koppel et al., 2009; Stamatatos, 2009; Juola, 2012).

Authorship attribution methods can be generally divided into two categories (Stamatatos, 2009). In similarity-based methods, an anonymous text is attributed to some author whose writing style is most similar (by some distance metric). In machine learning methods, which we follow in this paper, anonymous texts are classified, using machine learning algorithms, into different categories (in this case, different authors).

Machine learning papers differ from each other by the features and machine learning algorithm. Examples of features include HFWs (Mosteller and Wallace, 1964; Argamon et al., 2007), character n-gram (Kjell, 1994; Hoorn et al., 1999; Stamatatos, 2008), word n-grams (Peng et al., 2004), part-of-speech n-grams (Koppel and Schler, 2003; Koppel et al., 2005) and vocabulary richness (Abbasi and Chen, 2005).

The various machine learning algorithms used include naive Bayes (Mosteller and Wallace, 1964; Kjell, 1994), neural networks (Matthews and Merriam, 1993; Kjell, 1994), K-nearest neighbors (Kjell et al., 1995; Hoorn et al., 1999) and SVM (De Vel et al., 2001; Diederich et al., 2003; Koppel and Schler, 2003).

Traditionally, authorship attribution systems have mainly been evaluated against long texts such as theater plays (Mendenhall, 1887), essays (Yule, 1939; Mosteller and Wallace, 1964), biblical books (Mealand, 1995; Koppel et al., 2011a) and book chapters (Argamon et al., 2007; Koppel et al., 2007). In recent year, many works focused on web data such as emails (De Vel et al., 2001; Koppel and Schler, 2003; Abbasi and Chen, 2008), web forum messages (Abbasi and Chen, 2005; Solorio et al., 2011), blogs (Koppel et al., 2006; Koppel et al., 2011b) and chat messages (Abbasi and Chen, 2008). Some works focused on SMS messages (Mohan et al., 2010; Ishihara, 2011).

Authorship Attribution on Twitter. The performance of authorship attribution systems on short texts is affected by several factors (Stamatatos, 2009). These factors include the number of candidate authors, the training set size and the size of the test document.

Very few authorship attribution works experimented with Twitter. Unlike our work, all used a single group of authors (group sizes varied between 3-50). Layton et al. (2010) used the SCAP methodology (Frantzeskou et al., 2007) with character n-gram features. They experimented with 50 authors and compared different numbers of tweets per author (values between 20-200). Surprisingly, they showed that their system does not improve when given more training tweets. In our work, we noticed a different trend, and showed that more data can be extremely valuable for authorship attribution systems on micro-messages (see Section 6). Silva et al. (2011) trained an SVM classifier with various features (e.g., punctuation and vocabulary features) on a small dataset of three authors only, with varying training set size. Although their work used a set of Twitter-specific features that we do not explicitly use, our features implicitly cover a large portion of their features (such as punctuation and emoticon

features, which are largely covered by character n-grams).

Boutwell (2011) used a naive Bayes classifier with character n-gram features. She experimented with 50 authors and two training size values (120 and 230). She also provided a set of experiments that studied the effect of joining several tweets into a single document. Mikros and Perifanos (2013) trained an SVM classifier with character n-gram and word n-grams. They experimented with 10 authors of Greek text, and also joined several tweets into a single document. Joining several tweets into a longer document is appealing since it can lead to substantial improvement of the classification results, as demonstrated by the works above. However, this approach requires the test data to contain several tweets that are known a-priori to be written by the same author. This assumption is not always realistic. In our paper, we intentionally focus on a single tweet as document size.

Flexible Patterns. Patterns were introduced by (Hearst, 1992), who used hand crafted patterns to discover hyponyms. Hard coded patterns were used for many tasks, such as discovering meronymy (Berland and Charniak, 1999), noun categories (Widdows and Dorow, 2002), verb relations (Chklovski and Pantel, 2004) and semantic class learning (Kozareva et al., 2008).

Patterns were first extracted in a fully unsupervised manner ("flexible patterns") by (Davidov and Rappoport, 2006), who used flexible patterns in order to establish noun categories, and (Biciçi and Yuret, 2006) who used them for analogy question answering. Ever since, flexible patterns were used as features for various tasks such as extraction of semantic relationships (Davidov et al., 2007; Turney, 2008b; Bollegala et al., 2009), detection of synonyms (Turney, 2008a), disambiguation of nominal compound relations (Davidov and Rappoport, 2008a), sentiment analysis (Davidov et al., 2010b) and detection of sarcasm (Tsur et al., 2010).

9 Conclusion

The main goal of this paper is to measure to what extent authors of micro-messages can be identified. We have shown that authors of very short texts can be successfully identified in an array of authorship attribution settings reported for long documents. This is the first work on micro-messages to address some of these settings. We introduced the concept of *k-signature*. Using this concept, we proposed an interpretation of our results. Last, we presented the first authorship attribution system that uses flexible patterns, and demonstrated that using these features significantly improves over other systems. Our system obtains 6.1% improvement over the current state-of-the-art.

Acknowledgments

We would like to thank Elad Eban and Susan Goodman for their helpful advice, as well as Robert Layton for providing us with his dataset. This research was funded (in part) by the Harry and Sylvia Hoffman leadership and responsibility program (for the first author) and the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

Ahmed Abbasi and Hsinchun Chen. 2005. Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20:67–75.

Ahmed Abbasi and Hsinchun Chen. 2008. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Transactions on Information Systems*, 26(2):7:1–7:29.

Shlomo Argamon, Casey Whitelaw, Paul Chase, Sobhan Raj Hota, Navendu Garg, and Shlomo Levitan. 2007. Stylistic text classification using functional lexical features: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 58(6):802–822.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proc. of ACL*, pages 57–64, College Park, Maryland, USA.

Ergun Biciçi and Deniz Yuret. 2006. Clustering word pairs to answer analogy questions. In *Proc. of TAINN*, pages 1–8.

Danushka T. Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2009. Measuring the similarity between implicit semantic relations from the web. In *Proc. of WWW*, New York, New York, USA. ACM Press.

Sarah R. Boutwell. 2011. Authorship Attribution of Short Messages Using Multimodal Features. Master's thesis, Naval Postgraduate School.

John Burrows. 2002. 'Delta': a Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing*, 17(3):267–287.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In Dekang Lin and Dekai Wu, editors, *Proc. of EMNLP*, pages 33–40, Barcelona, Spain.

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proc. of ACL-Coling*, pages 297–304, Sydney, Australia.

Dmitry Davidov and Ari Rappoport. 2008a. Classification of semantic relationships between nominals using pattern clusters. In *Proceedings of ACL-08: HLT*, pages 227–235, Columbus, Ohio, June. Association for Computational Linguistics.

Dmitry Davidov and Ari Rappoport. 2008b. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proc. of ACL-HLT*, pages 692–700, Columbus, Ohio.

Dmitry Davidov, Ari Rappoport, and Moshe Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proc. of ACL*, pages 232–239, Prague, Czech Republic.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010a. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proc. of CoNLL*, pages 107–116, Uppsala, Sweden.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010b. Enhanced sentiment learning using twitter hashtags and smileys. In *Proc. of Coling*, pages 241–249, Beijing, China.

Olivier De Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2001. Mining e-mail content for author identification forensics. *ACM Sigmod Record*, 30(4):55–64.

Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. 2003. Authorship attribution with support vector machines. *Applied intelligence*, 19(1-2):109–123.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proc. of the 4th Web as Corpus Workshop*, WAC-4.

Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, and Carole E Chaski. 2007. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *Int Journal of Digital Evidence*, 6(1):1–18.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of Coling – Volume 2*, pages 539–545, Stroudsburg, PA, USA.

Johan F Hoorn, Stefan L Frank, Wojtek Kowalczyk, and Floor van der Ham. 1999. Neural network identification of poets using letter sequences. *Literary and Linguistic Computing*, 14(3):311–338.

Shunichi Ishihara. 2011. A forensic authorship classification in sms messages: A likelihood ratio based approach using n-gram. In *Proc. of the Australasian Language Technology Association Workshop 2011*, pages 47–56, Canberra, Australia.

Patrick Juola. 2012. Large-scale experiments in authorship attribution. *English Studies*, 93(3):275–283.

Bradley Kjell, W Addison Woods, and Ophir Frieder. 1995. Information retrieval using letter tuples with neural network and nearest neighbor classifiers. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1222–1226. IEEE.

Bradley Kjell. 1994. Authorship determination using letter pair frequency features with neural network classifiers. *Literary and Linguistic Computing*, 9(2):119–124.

Moshe Koppel and Jonathan Schler. 2003. Exploiting stylistic idiosyncrasies for authorship attribution. In *Proc. of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, volume 69, page 72.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proc. of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 624–628, New York, NY, USA.

Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri. 2006. Authorship attribution with thousands of candidate authors. In *SIGIR*, pages 659–660.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *JMLR*, 8:1261–1276.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *J. Am. Soc. Inf. Sci. Technol.*, 60(1):9–26.

Moshe Koppel, Navot Akiva, Idan Dershowitz, and Nachum Dershowitz. 2011a. Unsupervised decomposition of a document into authorial components. In *Proc. of ACL-HLT*, pages 1356–1364, Portland, Oregon, USA.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011b. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym

pattern linkage graphs. In *Proc. of ACL-HLT*, pages 1048–1056, Columbus, Ohio.

Robert Layton, Paul Watters, and Richard Dazeley. 2010. Authorship attribution for twitter in 140 characters or less. In *Proc. of the 2010 Second Cybercrime and Trustworthy Computing Workshop*, CTC '10, pages 1–8, Washington, DC, USA. IEEE Computer Society.

Robert AJ Matthews and Thomas VN Merriam. 1993. Neural computation in stylometry i: An application to the works of shakespeare and fletcher. *Literary and Linguistic Computing*, 8(4):203–209.

DL Mealand. 1995. Correspondence analysis of luke. *Literary and linguistic computing*, 10(3):171–182.

Thomas Corwin Mendenhall. 1887. The characteristic curves of composition. *Science*, ns-9(214S):237–246.

George K Mikros and Kostas Perifanos. 2013. Authorship attribution in greek tweets using authors multilevel n-gram profiles. In *2013 AAAI Spring Symposium Series*.

Ashwin Mohan, Ibrahim M Baggili, and Marcus K Rogers. 2010. Authorship attribution of sms messages using an n-grams approach. Technical report, CERIAS Tech Report 2011.

Frederick Mosteller and David Lee Wallace. 1964. *Inference and disputed authorship: The Federalist*. Addison-Wesley.

Fuchun Peng, Dale Schuurmans, and Shaojun Wang. 2004. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval*, 7(3-4):317–345.

Conrad Sanderson and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In *Proc. of EMNLP*, pages 482–491, Sydney, Australia.

Rui Sousa Silva, Gustavo Laboreiro, Luís Sarmento, Tim Grant, Eugénio Oliveira, and Belinda Maia. 2011. 'twazn me!!! ;(' automatic authorship analysis of micro-blogging messages. In *Proc. of the 16th international conference on Natural language processing and information systems*, NLDB'11, pages 161–168, Berlin, Heidelberg. Springer-Verlag.

Thamar Solorio, Sangita Pillay, Sindhu Raghavan, and Manuel Montes-Gomez. 2011. Modality specific meta features for authorship attribution in web forum posts. In *Proc. of IJCNLP*, pages 156–164, Chiang Mai, Thailand, November.

Efstathios Stamatatos. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Inf. Process. Manage.*, 44(2):790–799.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm–a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proc. of ICWSM*.

Peter Turney. 2008a. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proc. of Coling*, pages 905–912, Manchester, UK, August. Coling 2008 Organizing Committee.

Peter D. Turney. 2008b. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33:615–655.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proc. of Coling*, pages 1–7, Stroudsburg, PA, USA.

George Udny Yule. 1939. On sentence-length as a statistical characteristic of style in prose: with application to two cases of disputed authorship. *Biometrika*, 30(3-4):363–390.

Detection of Product Comparisons – How Far Does an Out-of-the-box Semantic Role Labeling System Take You?

Wiltrud Kessler and **Jonas Kuhn**
Institute for Natural Language Processing
University of Stuttgart
wiltrud.kessler@ims.uni-stuttgart.de

Abstract

This short paper presents a pilot study investigating the training of a standard Semantic Role Labeling (SRL) system on product reviews for the new task of detecting comparisons. An (opinionated) comparison consists of a comparative "predicate" and up to three "arguments": the entity evaluated positively, the entity evaluated negatively, and the aspect under which the comparison is made. In user-generated product reviews, the "predicate" and "arguments" are expressed in highly heterogeneous ways; but since the elements *are* textually annotated in existing datasets, SRL is technically applicable. We address the interesting question how well training an out-of-the-box SRL model works for English data. We observe that even without any feature engineering or other major adaptions to our task, the system outperforms a reasonable heuristic baseline in all steps (predicate identification, argument identification and argument classification) and in three different datasets.

1 Introduction

Sentiment analysis deals with the task of determining the polarity of an opinionated document or a sentence, in product reviews typically with regard to some target product. A common way to express sentiment about some product is by comparing it to a different product. In the corpus data we use, around 10% of sentences contain at least one comparison. Here are some examples of comparison sentences from our corpus:

(1) a. "[This camera]$_{\mathrm{E}+}$ ... its [screen]$_{\mathrm{A}}$ is much **bigger** than the [400D]$_{\mathrm{E}-}$."

b. "[D70]$_{\mathrm{E}+}$ **beats** [EOS 300D]$_{\mathrm{E}-}$ in almost [every category]$_{\mathrm{A}}$, EXCEPT ONE."

c. "[Noise suppression]$_{\mathrm{A}_1\mathrm{A}_2}$ was generally **better**$_1$ than the [D80]$_{\mathrm{E}-_1}$'s and much **better**$_2$ than the [Rebel]$_{\mathrm{E}-_2}$'s."

d. "A striking **difference** between the [EOS 350D]$_{\mathrm{E}-}$ and the new [EOS 400D]$_{\mathrm{E}+}$ concerns the [image sensor]$_{\mathrm{A}}$."

Note that our definition of comparisons is broader than the linguistic category of comparative sentences, which only includes sentences that contain a comparative adjective or adverb. For our work, we consider comparisons expressed by any Part of Speech (POS).

A comparison contains several parts that must be identified in order to get meaningful information. We call the word or phrase that is used to express the comparison ("better", "beats", ...) a *comparative predicate*. A comparison involves two *entities*, one or both of them may be implicit. In our data, most of the entities are products, e.g., the two cameras "D70" and "EOS 300D" in sentence 1b. In graded comparisons, entity+ (E+) is the entity that is being evaluated positively, entity- (E-) the entity evaluated negatively. In many sentences one attribute or part of a product is being compared, like "image sensor" in sentence 1d. We call this the *aspect* (A).

The task we want to solve for a given comparison sentence is to detect the comparative predicate, the entities that are involved and the aspect that is being compared. We borrow our methodology from Semantic Role Labeling (SRL). In SRL, *events* are expressed by predicates and *participants* of these events are expressed by arguments that fill different semantic roles. Adapted to the problem of detecting comparisons, the events we are interested in are comparative predicates and the arguments are the two entities and the aspect that is being compared.

Due to the diversity of possible ways of expressing comparisons, the "predicates" and "arguments"

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1892–1897,
Seattle, Washington, USA, 18-21 October 2013.

in this task are more heterogeneous categories than in standard SRL based on PropBank and NomBank annotations. Moreoever, the existing labeled datasets are based on an annotation methodology which gave the annotators a lot of freedom in deciding on the linguistic anchoring of the "predicate" and "arguments". This adds to the heterogeneity of the observed constructions and makes it even more interesting to ask the question how far an out-of-the-box SRL model can take you.

In this work, we re-train an existing SRL system (Björkelund et al., 2009) on product review data labeled with comparative predicates and arguments. We show that we can get reasonable results without any feature engineering or other major adaptions. This is an encouraging result for a linguistically grounded modeling approach to comparison detection.

2 Related Work

The syntax and semantics of comparative sentences have been the topic of research in linguistics for a long time (Moltmann, 1992; Kennedy, 1999). However, our focus is on computational methods and we also treat comparisons that are not comparative sentences in a linguistic sense.

In sentiment analysis, some studies have been presented to identify comparison sentences. Jindal and Liu (2006a) report good results on English using class sequential rules based on keywords as features for a Naive Bayes classifier. A similar approach for Korean is presented by Yang and Ko (2009; 2011b; 2011a). In our work, we do not address the task of identifying comparison sentences, we assume that we are given a set of such sentences.

The step we are concerned with is the detection of relevant parts of a comparison. To identify entities and aspect, Jindal and Liu (2006b) use an involved pattern mining process to mine label sequential rules from annotated English sentences. A similar approach is again presented by Yang and Ko (2011a) for Korean. In contrast to their complicated processing, we simply use an existing SRL system out of the box. Both approaches consider only nouns and pronouns for entities and aspects, we use all POS and allow for multi-word arguments. Jindal and Liu (2006b) base the recognition of comparative predicates on a list of manually compiled keywords. We use this as our baseline. Our approach is not dependent on a set of keywords and is therefore more easily adaptable to a new domain.

All works label the entities according to their position with respect to the predicate. This requires the identification of the preferred entity in a non-equal comparison as an additional step. Ganapathibhotla and Liu (2008) use hand-crafted rules based on the polarity of the predicate for this task. As we label the entities with their roles from the start, we solve both problems at the same time.

Xu et al. (2011) cast the task as a relation extraction problem. They present an approach that uses conditional random fields to extract relations (*better*, *worse*, *same* and *no_comparison*) between two entitites, an attribute and a predicate phrase.

The approach of Hou and Li (2008) is most related to our approach. They use SRL with standard SRL features to extract comparative relations from Chinese sentences. We confirm that SRL is a viable method also for English. In their experiments they report good results on gold parses, but observe a drop in performance when they use their method on automatic parses. All our experiments are conducted on automatically obtained parses.

3 Approach

The input to our system is a sentence that we assume to contain at least one comparison. The result of our processing are one or more comparative predicates and for each predicate three arguments: The two entities that are being compared, and the aspect they are compared in. More formally speaking, for every sentence we expect to get one or more 4-tupels (predicate, entity+, entity-, aspect). Entity+ is the entity that is being evaluated as better than entity-. Any of the arguments may be empty. Currently, we treat only single words as comparative predicates. Annotated multi-word predicates are mapped to one word. We allow for multi-word arguments, but annotate only the head word of the phrase and treat it as a one word argument for evaluation. We do not place any restrictions on possible POS.

We use a standard pipeline approach from SRL. As a first step, the comparative predicate is identified. The next step in SRL would be predicate

disambiguation to identify the different frames this predicate can express. As we do not have such frame information, predicate disambiguation is not performed in our pipeline.

After we have identified the predicates, the next step is to identify their arguments. The identification step is a binary classification whether a word in the sentence is some argument of the identified predicate. As a final classification step, it is determined for each found argument whether this argument is entity+, entity- or the aspect.

We use an existing SRL system (Björkelund et al., 2009)[1] and the features developed for SRL, based on the output of the MATE dependency parser (Bohnet, 2010). Features use attributes of the predicate itself, its head or its dependents. Additionally, for argument identification and classification there are features that describe the relation of predicate and argument, the argument itself, its leftmost and rightmost dependent and left and right sibling.

For the classification tasks of the pipeline, the SRL system uses regularized linear logistic regression from the LIBLINEAR package (Fan et al., 2008). We set the SRL system to train separate classifiers for predicates of different POS. In preliminary experiments, we have found this to perform slightly better than training one classifier for all kinds of predicates, although the difference is not significant. We do not use the reranker.

4 Experiments

Data. We use the JDPA corpus[2] by J. Kessler et al. (2010) for our experiments. It contains blog posts about cameras and cars. We use the annotation class "Comparison" that has four annotation slots. We convert the "more" slot to entity+, the "less" slot to entity- and the "dimension" slot to the aspect. For now, we ignore the "same" slot which indicates if the two mentions are ranked as equal.

We have also tested our approach on the dataset used in (Jindal and Liu, 2006b)[3]. We use all comparisons annotated as types 1 to 3 (ignoring type 4, non-gradable comparisons). In this dataset (J&L), entities are annotated as entity 1 or entity 2 depending on their position before or after the predicate. We keep this annotation and train our system to assign these labels.

	JDPA		J&L
	cameras	cars	
all sentences	5230	14003	7986
comparison sentences	505	1094	649
predicates	642	1327	695
distinct predicates	147	252	122
preds. occurring once	87	147	61
Entity+ / 1	517	1091	657
Entity- / 2	511	1068	331
Aspect	623	1107	526

Table 1: Statistics about the datasets

We do sentence segmentation and tokenization with the Stanford Core NLP[4]. Annotations are mapped to the extracted tokens. We ignore annotations that do not correspond to complete tokens. In the JDPA corpus, if an annotated argument is outside the current sentence, we follow the coreference chain to find a coreferent annotation in the same sentence. If this is not successful, the argument is ignored. We extract all sentences where we found at least one comparative predicate as our dataset.

Table 1 shows some statistics of the data.

Evaluation Setup. We evaluate on each dataset separately using 5-fold cross-validation. We report precision (P), recall (R), F1-measure (F1), and for argument classification macro averaged F1-measure ($F1_m$) over the three arguments. Bold numbers denote the best result in each column and dataset. We mark a F1-measure result with * if it is significantly higher than all previous lines.[5]

Results on Predicates. We have implemented two baselines based on previous work. The simplest baseline, *BL POS* classifies all tokens with a comparative POS ('JJR', 'JJS', 'RBR', 'RBS') as predicates. A more sophisticated baseline, *BL Keyphrases*, uses a list of about 80 manually com-

[1] http://code.google.com/p/mate-tools/

[2] Available from http://verbs.colorado.edu/jdpacorpus/ – we ignore cars batch 009 where no arguments of comparative predicates are annotated.

[3] Available from http://www.cs.uic.edu/~liub/FBS/data.tar.gz – although the original paper works on some unknown subset of this data, so our results are not directly comparable to the results reported there.

[4] http://nlp.stanford.edu/software/corenlp.shtml

[5] Statistically significant at $p < .05$ using the approximate randomization test (Noreen, 1989) with 10000 iterations.

		P	R	F1
cams	BL POS	66.6	38.2	48.5
	BL Keyphrases	53.1	**62.8**	57.5*
	SRL	**73.8**	58.7	**65.4***
cars	BL POS	62.5	34.7	44.6
	BL Keyphrases	51.9	**56.5**	54.1*
	SRL	**73.2**	55.5	**63.2***
J&L	BL POS	74.3	52.9	61.8
	BL Keyphrases	61.5	**80.0**	69.5*
	SRL	**77.0**	68.1	**72.3***

Table 2: Results predicate identification

		P	R	F1
cams	BL	49.4	**47.1**	48.2
	SRL	**66.5**	38.0	**48.4**
cars	BL	50.2	**50.1**	50.1
	SRL	**68.7**	42.2	**52.3***
J&L	BL	38.7	44.6	41.5
	SRL	**68.5**	**45.2**	**54.5***

Table 3: Results argument identification (gold predicates)

		Entity+ / 1			Entity- / 2			Aspect			$F1_m$
		P	R	F1	P	R	F1	P	R	F1	
cams	BL	30.1	**31.7**	**30.9**	21.2	21.3	21.3	61.8	**51.2**	56.0	36.1
	SRL	**38.6**	17.4	24.0	**43.7**	**24.5**	**31.4**	**69.9**	47.7	**56.7**	**37.3**
cars	BL	31.1	**32.7**	**31.9**	23.0	24.0	23.5	49.3	**44.5**	**46.8**	34.0
	SRL	**39.5**	22.9	29.0	**48.1**	**31.0**	**37.7**	**58.4**	36.2	44.7	**37.1***
J&L	BL	43.2	39.4	41.2	19.0	31.1	23.6	15.0	17.1	16.0	26.9
	SRL	**58.3**	**47.2**	**52.1**	**60.8**	**35.6**	**45.0**	**58.8**	**30.6**	**40.3**	**45.8***

Table 4: Results argument classification (gold predicates)

piled comparative keyphrases from (Jindal and Liu, 2006a) in addition to the POS tags.

Table 2 shows the result of our experiments. Our method significantly outperforms both baselines in all datasets. The generally low recall values are mainly a result of the wide variety of predicates that are used to express comparisons (see Discussion).

Results on Arguments. To get results indepent of the errors introduced by the relatively low performance on predicate identification, we use annotated predicates (gold predicates) as a starting point for the argument experiments. All results drop about 10% when system predicates are used.

As a *baseline (BL)* for argument identification and classification, we use some heuristics based on the characteristics of our data. Most entities are (pro)nouns and most predicates are positive, so we classify the first noun or pronoun before the predicate as entity+ (entity 1 for J&L) and the first noun or pronoun after the predicate as a entity- (entity 2). If the predicate is a comparative adjective, we classify the predicate itself as aspect, because this type of annotation is very frequent in the JDPA data. For other predicates except nouns and verbs, we classify the direct head of the predicate as aspect.

Table 3 shows the results for argument identification, the results for argument classification can be seen in Table 4. Our system outperforms the baseline for all datasets. The differences are significant except for the cameras dataset. In general, the numbers are low. We will discuss some reasons for this in the next section.

5 Discussion

Sparseness. There are many ways to express a comparison and the size of the available training data is relatively small. This strongly influences the recall of our system as many predicates and arguments occur only once. As we can see in Table 1, 60% of the predicates in the cameras dataset occur only once. In contrast, only 12 predicates occur ten times or more. The trends are similar in the other datasets. This particularly affects verbs and nouns, where many colloquial expressions are used ("hammers", "pwns", "go head to head with", "put X to the sword", ...).

Argument identification and classification would benefit from generalizing over the many different product identifiers like "EOS 5D" or "D200". We want to try to use a Named Entity Recognition system trained on this type of entities for this purpose.

Sentiment Relevance. The following examples show a problem that is typical for sentiment analysis and responsible for many false positive predicates:

(2) a. "Relatively [**lower**]$_{\mathrm{A}}$ noise at higher ISO ..."
b. "... but [**higher**]$_{\mathrm{A}}$ then [Sony]$_{\mathrm{E+}}$"

Although "higher" often expresses a comparison like in sentence 2b, in sentence 2a it only describes a camera setting and should not be extracted as a comparative predicate. There has been considerable work in the areas of subjectivity classification (Wilson and Wiebe, 2003) and the related sentiment relevance (Scheible and Schütze, 2013) which we will try to use to detect such irrelevant, "descriptive" uses of comparative words.

Linguistic anchoring. In contrast to SRL, the task of comparison detection in reviews is a relatively new task without universally recognized definitions and annotation schemes. The annotators of the corpora had a lot of freedom in their choice of linguistic anchoring of the predicates and arguments. Consider these examples from the cameras dataset:

(3) a. "[**Lighter**]$_{\mathrm{A}}$ in weight compared to the [others]$_{\mathrm{E-}}$."
b. "... [its]$_{\mathrm{E+}}$ [better]$_{\mathrm{A}}$ and faster **compared** vs the [SB800 flash]$_{\mathrm{E-}}$ as well."
c. "... this camera's [screen]$_{\mathrm{E+}}$ is [**smaller**]$_{\mathrm{A}}$ than the [ones]$_{\mathrm{E-}}$ on some competing models ..."

Sentences 3a and 3b show a situation where two words are used to express the same comparison and it is unclear which one to chose as a predicate. The decision is left to the individual annotators.

There is some variety of annotations on arguments as well. In the JDPA data, a comparative adjective is often annotated as aspect, sometimes even when there is an alternative, e.g., "weight" in sentence 3a. Also, for a phrase like "its screen", we find "screen" annotated as the aspect (sentence 1a) or an entity (sentence 3c) – and both have their merit. We want to further study how different linguistic anchorings of comparisons effect classification performance.

Equative comparisons. As we can see from the confusion matrix of our system, the distinction between entity+ and entity- is very difficult to learn. In graded comparisons, the distinction is informative, but sentiment information would be needed for the correct assignment. There are also some problematic cases where the ranking cannot be inferred without the broader context, e.g., sentence 1d.

A more annotation-related problem concerns equative comparisons, i.e., both entities are rated as equal. The difference between entity+ and entity- is meaningless in this case. In the JDPA corpus, entities still have to be annotated as either entity+ or entity- and the annotation guidelines allow the annotator to choose freely. As a result, the data is noisy, for the same predicate sometimes entity- is before the predicate, sometimes entity+. If we eliminate this noise by always assigning the entities in order of surface position, we see a gain in macro averaged F1-measure for all systems of about 2% (cameras) to 4% (cars).

6 Conclusions

We presented a pilot experiment on using an SRL-inspired approach to detect comparisons (comparative predicate, entity+, entity-, aspect) in user generated content. We re-trained an existing SRL system on data that is labeled with comparative predicates and arguments. Even without feature engineering or major adaptions, our approach outperforms the baselines in three datasets in every task. This is an encouraging result for a linguistically grounded modeling approach to comparison detection.

For future work, we plan to include features that have been tailored specifically to the task of detecting product comparisons. To address the inherent diversity of expressions typical for user generated content, we want to employ generalization techniques, e.g., to detect product names. We also want to further study the different possible linguistic anchorings of comparisons and their effect on classification performance. Studies of this kind may also inform future data annotation efforts in that certain ways of anchoring the elements of a comparison linguistically may be more helpful than others. We also believe that the explicit modeling of different types (equative, superlative, non-equal gradable) of comparisons will have a positive effect on performance.

Acknowledgments

The work reported in this paper was supported by a Nuance Foundation Grant.

References

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual Semantic Role Labeling. In *Proceedings of CoNLL '09 Shared Task*, pages 43–48.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING '10*, pages 89–97.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of COLING '08*, pages 241–248.

Feng Hou and Guo-hui Li. 2008. Mining Chinese comparative sentences by semantic role labeling. In *Proceedings of ICMLC '08*, pages 2563–2568.

Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In *Proceedings of SIGIR '06*, pages 244–251.

Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In *Proceedings of AAAI '06*, pages 1331–1336.

Christopher Kennedy. 1999. *Projecting the Adjective: The Syntax and Semantics of Gradability and Comparison*. Outstanding Dissertations in Linguistics. Garland Pub.

Jason S. Kessler, Miriam Eckert, Lyndsay Clark, and Nicolas Nicolov. 2010. The 2010 ICWSM JDPA Sentiment Corpus for the Automotive Domain. In *Proceedings of ICWSM-DWC '10*.

Friederike Moltmann. 1992. *Coordination and Comparatives*. Ph.D. thesis, Massachusetts Institute of Technology.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses – an introduction*. Wiley & Sons.

Christian Scheible and Hinrich Schütze. 2013. Sentiment relevance. In *Proceedings of ACL '13*, pages 954–963.

Theresa Wilson and Janyce Wiebe. 2003. Annotating opinions in the world press. In *Proceedings of SIGdial '03*, pages 13–22.

Kaiquan Xu, Stephen Shaoyi Liao, Jiexun Li, and Yuxia Song. 2011. Mining comparative opinions from customer reviews for competitive intelligence. *Decis. Support Syst.*, 50(4):743–754, March.

Seon Yang and Youngjoong Ko. 2009. Extracting comparative sentences from Korean text documents using comparative lexical patterns and machine learning techniques. In *Proceedings of the ACL-IJCNLP '09*, pages 153–156.

Seon Yang and Youngjoong Ko. 2011a. Extracting comparative entities and predicates from texts using comparative type classification. In *Proceedings of HLT '11*, pages 1636–1644.

Seon Yang and Youngjoong Ko. 2011b. Finding relevant features for Korean comparative sentence extraction. *Pattern Recogn. Lett.*, 32(2):293–296, January.

A multi-Teraflop Constituency Parser using GPUs

John Canny **David Hall** **Dan Klein**
UC Berkeley
Berkeley, CA, 94720
canny@berkeley.edu, dlwh,klein@cs.berkeley.edu

Abstract

Constituency parsing with rich grammars remains a computational challenge. Graphics Processing Units (GPUs) have previously been used to accelerate CKY chart evaluation, but gains over CPU parsers were modest. In this paper, we describe a collection of new techniques that enable chart evaluation at close to the GPU's practical maximum speed (a Teraflop), or around a half-trillion rule evaluations per second. Net parser performance on a 4-GPU system is over 1 thousand length-30 sentences/second (1 trillion rules/sec), and 400 general sentences/second for the Berkeley Parser Grammar. The techniques we introduce include grammar compilation, recursive symbol blocking, and cache-sharing.

1 Introduction

Constituency parsing with high accuracy (e.g. latent variable) grammars remains a computational challenge. The $O(Gs^3)$ complexity of full CKY parsing for a grammar with G rules and sentence length s, is daunting. Even with a host of pruning heuristics, the high cost of constituency parsing limits its uses. The most recent Berkeley latent variable grammar for instance, has 1.7 million rules and requires about a billion rule evaluations for inside scoring of a single length-30 sentence. GPUs have previously been used to accelerate CKY evaluation, but gains over CPU parsers were modest. e.g. in Yi et al. (2011) a GPU parser is described for the Berkeley Parser grammar which achieves 5 sentences per second on the first 1000 sentences of Penn Treebank section 22 Marcus et al. (1993), which is comparable with the best CPU parsers Petrov and Klein (2007). Our parser achieves 120 sentences/second per GPU for this sentence set, and over 250 sentences/sec on length ≤ 30 sentences. These results use a Berkeley Grammar approximately twice as big as Yi et al. (2011), an apparent 50x improvement. On a 4-GPU system, we achieve 1000 sentences/sec for length ≤ 30 sentences. This is 2 orders of magnitude faster than CPU implementations that rely heavily on pruning, and 5 orders of magnitude faster than full CKY evaluation on a CPU.

Key to these results is a collection of new techniques that enable GPU parsing at close to the GPU's practical maximum speed (a Teraflop for recent GPUs), or around a half-trillion rule evaluations per second. The techniques are:

1. Grammar compilation, which allows register-to-register code for application of grammar rules. This gives an order of magnitude (10x) speedup over alternative approaches that use shared memory.
2. Symbol/rule blocking of the grammar to respect register, constant and instruction cache limits. This is precondition for 1 above, and the details of the partitioning have a big (> 4x) effect on performance.
3. Sub-block partitioning to distribute rules across the stream processors of the GPU and allow L2 cache acceleration. A factor of 2 improvement.

The code generated by our parser comes close to the theoretical limits of the GPU. 80% of grammar rules

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1898–1907,
Seattle, Washington, USA, 18-21 October 2013.

are evaluated using a single-cycle register-to-register instruction.

2 GPU Design Principles

In this paper, we focus on the architecture of recent NVIDIA® GPUs, though many of the principles we describe here can be applied to other GPUs (e.g. those made by AMD®.) The current NVIDIA® Kepler™ series GPU contains between 2 and 16 "stream processors" or SMX's which share an L2 cache interfacing to the GPUs main memory Anonymous (2013). The SMXs in turn comprise 192 cores which share a memory which is partitioned into "shared memory" and L1 cache. Shared memory supports relatively fast communication between threads in an SMX. Communication between SMXs has to pass through slower main memory.

The execution of instructions within SMXs is virtualized and pipelined - i.e. it is not a simple task to count processors, although there are nominally 192 in the Kepler™ series. Register storage is not attached to cores, instead registers are associated in blocks of 63 or 255 (depending on Kepler™ sub-architecture) with running threads. Because of this, it is usually easier for the programmer to think of the SMXes as 1024 thread processors. These 1024 threads are grouped into 32 groups of 32 threads called *warps*. Each warp of threads shares a program counter and executes code in lock-step. However, execution is not SIMD - all threads do not execute all instructions. When the warp encounters a branching instruction, all branches that are satisfied by some thread will be executed in sequence. Each thread only executes the instructions for its own branch, and idles for the others. NVIDIA® calls this model SIMT (Single Instruction, Multiple Threads). Execution of diverging branches by a warp is called *warp divergence*. While it simplifies programming, warp divergence understandably hurts performance and our first goal is to avoid it.

GPUs are generally optimized for single-precision floating point arithmetic in support of rendering and simulation. Table 1 shows instruction throughput (number of instructions that are executed per cycle on each SMX). The Kepler™ series has two architectural sub-generations (3.0 and 3.5) with significant differences in double-precision support. Data from Anonymous (2012) and NVIDIA (2012).

Instruction type	Architecture	
	3.0	3.5
Shared memory word access	32	32
FP arithmetic +,-,*,FMA	192	192
DP arithmetic +,-,*,FMA	8	64
Integer +,-	160	160
Integer *,FMA	32	32
float sin, exp, log,...	32	32

Table 1: Instructions per cycle per SMX in generation 3.0 and 3.5 Kepler™ devices

In the table, FP is floating point, DP is double precision, and FMA is a single-cycle floating-piont fused multiply-add used in most matrix and vector operations ($A \leftarrow A + B * C$). Note next that floating point (single precision) operations are extremely fast and there is an FPU for each of the 192 processors. Double precision floating point is 3x slower on high-end 3.5 GPUS, and much slower (24x) on the commodity 3.0 machines. While integer addition is fast, integer multiply is much slower. Perhaps most surprising is the speed of single-precision transcendental function evaluation, log, exp, sin, cos, tan, etc., which are as fast as shared memory accesses or integer multiplication, and which amount to a quarter-trillion transcendental evaluations per second on a GTX-680/K10.

PCFG grammar evaluation nominally requires two multiplications and an addition per rule (section 4) which can be written:

$$S_{ij,m} = \sum_{k=1\ldots j;\ n,p\in Q} S_{ik,n} S_{(k+1)j,p} c_{mnp} \qquad (1)$$

i.e. the CKY node scores are sums of products of pairs of scores and a weight. This suggests that at least in principle, it's possible to achieve a trillion rule evaluations per second on a 680 or K10 device, using a * and an FMA operation for each rule. That assumes we are doing register-to-register operations however. If we worked through shared memory (first line of the table), we would be limited to about 80 billion evaluations/sec, 20 times slower. The analysis underscores that high performance for parsing on a GPU is really a challenge of data movement. We next review the different storage types and their

bandwidths, since prudent use of, and movement between storage types is the key to performance.

2.1 Memory Types and Speeds

There are six types of storage on the GPU which matter for us. For each type, we give the capacity and aggregate bandwidth on a typical device (a GTX-680 or K10 running at 1GHz).

Register files These are virtualized and associated with threads rather than processors. 256kB per SMX. Each thread in architecture 3.0 devices can access 63 32-bit registers, or 255 registers for 3.5 devices. Aggregate bandwidth 40 TB/s.

Shared memory/L1 cache is shared by all threads in an SMX. 64kB per SMX partitioned into shared memory and cache functions. Aggregate bandwidth about 1 TB/s.

Constant memory Each SMX has a 48kB read/only cache separate from the L1 cache. It can store grammar constants and has much higher bandwidth than shared memory. Broadast bandwidth 13 TB/s.

Instruction cache is 8 KB per SMX. Aggregate bandwidth 13 TB/s.

L2 cache is 0.5-1.5 MB, shared between all SMXs. Aggregate bandwidth 500 GB/s.

Global memory is 2-6GB typically, and is shared by all SMXs. GPUs use a particularly fast form of SDRAM (compared to CPUs) but it is still much slower than the other memory types above. Aggregate bandwidth about 160 GB/s.

There is one more very important principle: Coalesced main memory access. From the above it can be seen that main memory access is much slower than other memories and can easily bottleneck the calculations. The figure above (160 GB/s) for main memory access assumes such access is *coalesced*. Each thread in the GPU has a thread and a block number which determines where it runs on the hardware. Consecutively-numbered threads should access consecutive main memory locations for fast memory access.

These parameters suggest a set of design principles for peak performance:

1. Maximize use of registers for symbol scores, and minimize use of shared memory (in fact we will not use it at all).
2. Maximize use of constant memory for rule weights, and minimize use of shared memory.
3. Partition the rule set into blocks that respect the limits on number of registers, constant memory (needed for grammar rules probabilities) and instruction cache limits.
4. Minimize main memory access and use L2 cache to speed it up.

Lets look in more detail at how to achieve this.

3 Anatomy of an Efficient GPU Parser

High performance on the GPU requires us to minimize code divergence. This suggests that we do not use a lexicalized grammar or a grammar that is sensitive to the position of a span within the sentence. These kinds of grammars—while highly accurate—have irregular memory access patterns that conflict with SIMD execution. Instead, an unlexicalized approach like that of Johnson (2011) or Klein and Manning (2003), or a latent variable approach like that of Matsuzaki et al. (2005) or Petrov et al. (2006) are more appropriate. We opt for the latter kind: latent variable grammars are fairly small, and their accuracies rival lexicalized approaches.

Our GPU-ized inside algorithm maintains two data structures: parse charts that store scores for each labeled span, as usual, and a "workspace" that is used to actually perform the updates of the inside algorithm. Schematically, this memory layout is represented in Figure 1. A queue is maintained CPU-side that enqueues work items of the form (s, p, l, r), where s is a sentence, and p, l, and r specify the index in the parse chart for parent, left child, and right child, respectively. The outer loop proceeds in increasing span length (or height of parent node scores to be computed). Next the algorithm iterates over the available sentences. Then it iterates over the parent nodes at the current length in that sentences, and finally over all split points for the current parent node. In each case, work items are sent to the queue with that span for all possible split points.

When the queue is full—or when there are no more work items of that length—the queue is flushed

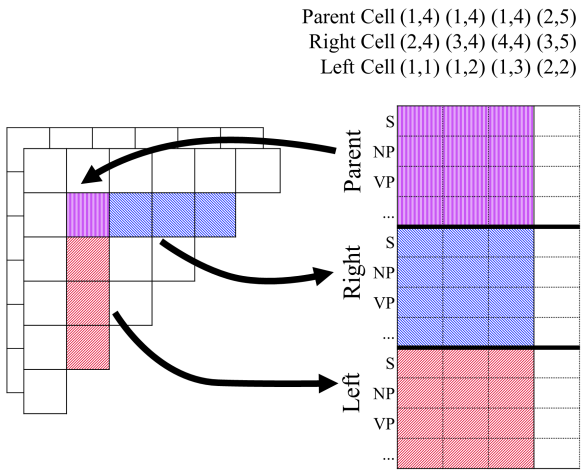


Figure 1: The architecture of the system. Parse charts are stored in triangular arrays laid out consecutively in memory. Scores for left and right children are transposed and copied into the "workspace" array, and the inside updates are calculated for the parent. Scores are then pushed back to the appropriate cell in the parse charts, maxing them with scores that are already there. Transposition ensures that reads and writes are coalesced.

to the GPU, which executes three steps. First, the scores for each left and right child are copied into the corresponding column in the workspace. Then inside updates are applied in parallel for all cells to get parent scores. Then parents are entered back to their appropriate cells in the parse charts. This is typically a many-to one atomic reduction (either a sum for probability scores, or a max for max-sum log probability scores). This process repeats until all span lengths have been processed.

3.1 The Inside Updates

The high-level goal of our parser is to use SIMD parallelism to evaluate the same rule across many spans (1024 threads are currently used to process 8192 spans in each kernel). This approaches allows us to satisfy the GPU performance desiderata from the previous section. As discussed in section 5 each GPU kernel actually processes a small subset of the symbols and rules for the grammar, and kernels are executed in sequence until the entire grammar has been processed. Each thread iterates over the rules in the same order, reading in symbols from the left child and right child arrays in main memory as necessary.

The two-dimensional work arrays must be stored in "symbol-major" order for this to work. That is, the parent VP for one work item is stored next to the parent VP for the next work item, while the VP symbol for the first work item is stored on the next "row" of the work array. The reason the workspace cells are stored in "symbol-major" order is to maximize coalesced access: each thread in the SMX accesses the same symbol for a different work item in parallel, and those work items are in consecutive memory locations.

3.2 The Copy-transpose Operations

Unlike the workspace arrays, the arrays for the parse charts are stored in "span-major" order, transposed from how they are stored in the workspace arrays. That is, for a given span, the NP symbol is next to the same span's VP symbol (for example). This order accelerates both symbol loading and Viterbi search later on. It requires a transpose-copy instead of "non-transposed" copy to move from chart to workspace arrays and back again, but note that a non-transposed copy (or direct access to the chart by the GPU compute kernel) would probably be slower. The reason is that any linear ordering of cells in the triangle table will produce short segments (less than 32 words and often less than 16) of consecutive memory locations. This will lead to many non-coalesced memory accesses. By contrast the span-major representation always uses vectors whose lengths equals the number of symbols (500-1000), and these can be accessed almost entirely with coalesced operations. The copy-transpose operations are quite efficient (the transpose itself is much faster than the I/O), and come close to the 160 GB/s GPU main memory limit.

The reverse copy-transpose (from parent workspace cells to chart) is typically many-to-one, since parent scores derive from multiple splits. They are implemented using atomic reduce operations (either atomic sum or atomic max) to ensure data consistency.

At the heart of our approach is the use of grammar compilation and symbol/rule blocking, described next.

4 Grammar Compilation

Each rule in a probabilistic context-free grammar can be evaluated with an update of the form:

$$S_{ij,m} = \sum_{k=1...j;\ n,p \in Q} S_{ik,n} S_{(k+1)j,p} c_{mnp} \quad (2)$$

where $S_{ij,m}$ is the score for symbol m as a generator of the span of words from position i to j in the input sentence, c_{mnp} is the probability that symbol m generates the binary symbol pair n, p, and Q is the set of symbols. The scores will be stored in a CKY chart indexed by the span ij and the symbol m.

To evaluate (2) as fast as possible, we want to use register variables which are limited in number. The location indices i, j, k can be moved outside the GPU kernel to reduce the variable count. We use symbols P, L and R for respectively the score of the parent, left child and right child in the CKY chart. Then the core relation in (2) can be written as:

$$P_m = \sum_{n,p \in Q} L_n R_p c_{mnp} \quad (3)$$

In the Kepler™ architecture, register arguments are non-indexed, i.e. one cannot access register 3 as an array variable R[i] with i=3[1]. So in order to use register storage for maximum speed, we must open-code the grammar. Symbols like L_3, R_{17} are encoded as variables $L003$ and $R017$, and each rule must appear as a line of C code:

```
P043 += L003*R017*0.023123f;
P019 += L012*R123*6.21354e-7f;
 :        :       :       :
```

Open-coding the grammar likely has a host of performance advantages. It allows both compiler and hardware to "see" what arguments are coming and schedule the operations earlier than a "grammar as data" approach. Note that we show here the sum-product code for computing inner/outer symbol probabilities. For Viterbi parse extraction we replace +,* with max,+ and work on log scores.

L and R variables must be loaded from main memory, while P-values are initialized to zero and then atomically combined (sum or max) with P-values in memory. Loads are performed as late as possible, that is, a load instruction will immediately precede the first use of a symbol:

```
float R031 = right[tid+65*stride];
P001 += L001*R031*1.338202e-001f;
```

where tid is the thread ID plus an offset, and stride is the row dimension of the workspace (typically 8192), and right is the main memory array of right symbol scores. Similarly, atomic updates to P-values occur as early as possible, right after the last update to a value:

```
G020 += L041*R008*6.202160e-001f;
atomicAdd(&par[tid+6*stride],G020);
```

These load/store strategies minimize the active life of each variable and allow reuse of register variables for symbols whose lifetimes do not overlap. This will be critical to successful blocking, described in the next section.

4.1 Common subexpressions

One interesting discovery made by the compiler was that the same L,R pair is repeated in several rules. In hindsight, this is obvious because the symbols in this grammar are splits of base symbols, and so splits of the parent symbol will be involved in rules with each pair of L,R splits. The compiler recognized this by turning the L,R pair into a common subexpression in a register. i.e. the compiler converts

```
P008 += L041*R008*6.200769e-001f;
P009 += L041*R008*6.201930e-001f;
P010 += L041*R008*6.202160e-001f;
```

into

```
float LRtmp = L041*R008;
P008 += LRtmp*6.200769e-001f;
P009 += LRtmp*6.201930e-001f;
P010 += LRtmp*6.202160e-001f;
```

and inspection of the resulting assembly code shows that each rule is compiled into a single fused multiply-add of LRtmp and a value from constant memory into the P symbol register. This allows grammar evaluation to approach the theoretical Gflop limit of the GPU. For this to occur, the rules need to be sorted with matching L,R pairs consecutive. The compiler does not discover this constraint otherwise or reorder instructions to make it possible.

[1] Even if indexing were possible, it is extremely unlikely that such accesses could complete in a single cycle

4.2 Exploiting L2 cache

Finally, we have to generate code to evaluate distinct minor cube rulesets on each of the 8 SMXes concurrently in order to benefit from the L2 cache, as described in the next section. CUDA™ (NVIDIA's GPU programming Platform) does not allow direct control of SMX target, but we can achieve this by running the kernel as 8 thread blocks and then testing the block ID within the kernel and dispatching to one of 8 blocks of rules. The CUDA™ scheduler will execute each thread block on a different SMX which gives the desired distribution of code.

5 Symbol and Rule Blocking

The grammar formula (3) is very sparse. i.e. most productions are impossible and most c_{mnp} are zero. For the Berkeley grammar used here, only 0.2% of potential rules occur. Normally this would be bad news for performance because it suggests low variable re-use. However, the update relation is a *tensor* rather than a matrix product. The re-use rate is determined by the number of rules in which a particular symbol occurs, which is actually very high (more than 1000 on average).

The number of symbols is about 1100 in this grammar, and only a fraction can be stored in a thread's register set at one time (which is either 63 or 255 registers). To compute all productions we will need to break the calculation into smaller groups of variables that can fit in the available register space.

We can visualize this geometrically in figure 2. The vectors of symbols P, L and R form the leading edges of this cube. The cube will be partitioned into smaller subcubes indexed by subsets of those symbols, and containing all the rules that apply between those symbols. The partitioning is chosen so that the symbols in that subset can fit into available register storage. In addition, the partitioning is chosen to induce the same number of rules in each cube - otherwise different code paths in the kernel will run longer than others, and reduce overall performance. This figure is a simplification - in order to balance the number of rules in each subcube, the partitioning is not uniform in number of symbols as the figure suggests.

As can be seen in figure 2, cube partitioning has two levels. The original P-L-R cube is first partitioned into "major" cubes, which are then partitioned into "minor" cubes (2x2x2 in the figure). A major cube holds the symbols and rules that are executed in a single GPU kernel. The minor cubes in a major cube hold the symbols and rules that are executed in a particular SMX. For the GTX-680 or K10 with 8 SMXs, this allows different SMXs to concurrently work on different 2x2x2 subcubes in a major cube. This arrangement substantially reduces main memory bandwidth through the L2 cache (which is shared between SMXes). Each symbol in a major cube will be loaded just once from main memory, but loaded into (up to) 4 different SMXes through the L2 cache. Subcube division for caching in our experiments roughly doubled the kernel's speed.

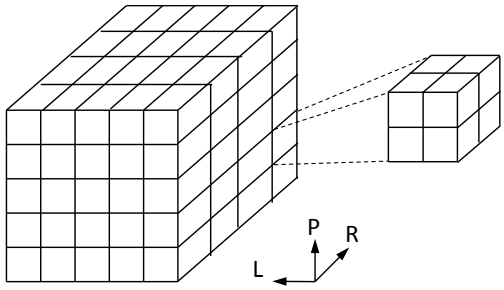


Figure 2: Partition of the cube of symbol combinations into major subcubes (left) and minor subcubes (right).

However, simple partitioning will not work. e.g. if we blocked into groups of 20 P, L, R symbols (in order to fit into 60 registers), we would need 1100/20 = 55 blocks along each edge, and a total of $55^3 \approx 160,000$ cells. Each symbol would need to be loaded $55^2 = 3025$ times, there would be almost no symbol re-use. Throughput would be limited by main memory speed to about 100 Gflops, an order of magnitude slower than our target. Instead, we use a rule partitioning scheme that creates as small a symbol footprint as possible in each cube. We use a spectral method to do this.

Before describing the spectral method we mention an optimization that drops the symbol count by 2. Symbols are either terminal or non-terminal, and in the Berkeley latent variable grammar there are roughly equal numbers of them (503 non-terminals and 631 terminals). All binary rules involve a non-terminal parent. L and R symbols may be either terminal or non-terminal, so there are 4 distinct types

of rules depending on the L, R, types. We handle each of these cases with a different kernel, which rougly halves the number of rules along each edge (it is either 503 or 631 along each edge). Furthermore, these kernels are called in different contexts, and a different number of times. e.g. XX (L, R, both non-terminal) kernels are called $O(s^3)$ times for sentences of length s because both L, R children can occur at every position in the chart. XT and TX kernels (with one terminal and one non-terminal symbol) are called only $O(s^2)$ times since one of L or R must be at the base of the chart. Finally TT kernels (both L and R are terminals) will be called $O(s)$ times. Performance is therefore dominated by the XX kernel.

5.1 Spectral Partitioning

We explored a number of partitioning schemes for both symbol and rule partitioning. In the end we settled on a spectral symbol partitioning scheme. Each symbol is a node in the graph to be partitioned. Each node is assigned a feature vector designed to match it to other nodes with similar symbols occuring in many rules. There was considerable evolution of this feature set to improve partitioning. In the end the vector for a particular P symbol is $a = (a_1, 0.1 * a_2, 0.1 * a_3)$ where a_1 is a vector whose elements are indexed by L, R pairs and whose values represent the number of rules involving both those symbols (and the parent symbol P), a_2 encodes L symbols and counts the number of rules containing that L symbol and P, and a_3 encodes the R symbols and counts rules containing that R symbol and P. This feature vector produces a high similarity between P symbols that exactly share many L,R pairs and lower similarity for shared L and R.

A spectral clustering/partitioning algorithm approximately minimizes the total edge weight of graph cuts. In our case, the total weight of a cut is to first order the product of the number of L,R pairs that occur on each side of the cut, and to second order the count of individual L and R pairs that span the cut. Let S and T be the counts for a particular LR pair or feature, then we are trying to minimize the product S*T while keeping the sum S+T, which is the total occurences of the feature on both sides of the partition, constant. Such a product is minimized when one of S or T is zero. Since many symbols are involved, this typically does not happen to an individual symbol, but this heuristic is successful at making the individual symbol or LR pair distributions across the cuts as unbalanced as possible. i.e. one side of the cut has very few instances of a given symbol. The number of instances of a symbol is an upper bound on the number of subcells in which than symbol occurs, and therefore on the number of times it needs to be loaded from memory. Repeating this operation recursively to produce a 3d cell decomposition also concentrates each symbol in relatively few cells, and so tends to reduce the total register count per cell.

In a bit more detail, from the vectors a above we construct a matrix A whose columns are the feature vectors for each P symbol. Next we construct the symmetric normalized Laplacian L for the adjacency matrix $A^T A$. We then compute the eigendecomposition of L, and extract the eigenvector corresponding to the second-smallest eigenvalue. Each node in the graph is assign a real weight from the corresponding element of this eigenvector. We sort by these weights, and partition the symbols using this sort order. We tried both recursive binary partitioning, and partitioning into k intervals using the original sort order, and obtained better results with the latter.

Partitioning is applied in order P, L, R to generate the major cubes of the rule/symbol partition, and then again to generate minor cubes. This partitioning is far more efficient than a naive partitioning. The XX ruleset for our Berkeley grammar has about 343,000 rules over a 503^3 cube of non-terminal symbols. The optimal PxLxR cube decomposition (optimal in net kernel throughput) for this ruleset was 6x2x2 for major cubes, and then 2x2x2 for minor cubes. This requires 6x2x2=24 GPU kernels, each of which encodes 2x2x2=8 code blocks (recall that each of the 8 SMXs executes a different code block from the same kernel)[2]. Most importantly the reload rate (the mean number of major cells containing a given symbol, or the mean number of times a symbol needs to be reloaded from main memory) drops to about 6 (vs. 3000 for naive partitioning). This is very significant. Each symbol is used on average $343,000/501 \approx 6000$ times overall by the XX ker-

[2]This cube decomposition also respects the constant cache and instruction cache limits

nel. Dropping the reload factor to 6 means that for every 1 main memory load of a symbol, there are approximately 1000 register or L2 cache reuses. A little further calculation shows that L2 cache items are used a little more than twice, so the register reuse rate within kernel code blocks is close to 500 on average. This is what allows teraflop range speeds.

Note that while the maximum number of registers per thread in the GTX-680 or K10 is 63, the average number of variables per minor cube is over 80 for our best-performing kernel, showing a number of variables have non-overlapping lifetimes. Sorting rules lexicographically by (L,R,P) does a good job of minimizing variable lifetime overlap. However the CUDA™ compiler reorders variables anyway with slightly worse performance on average (there seems to be no way around this, other than generating assembly code directly).

6 GPU Viterbi Parse Extraction

In sequential programs for chart generation, it is possible to compute and save a pointer to the best split point and score at each node in the chart. However, here the scores at each node are computed with fine-grained parallelism. The best split point and score cannot be computed until all scores are available. Thus there is a separate Viterbi step after chart scoring.

The gap between GPU and CPU performance is large enough that CPU Viterbi search was a bottleneck, even though it requires asymptotically less work ($O(Gs^2)$ worst case, $O(Gs)$ typical) vs $O(Gs^3)$ to compute the CKY scores. Therefore we wrote a non-recusive GPU-based Viterbi search. Current GPUs support "high-level" recursion, but there is no stack in the SMX. A recursive program must create software stack space in either shared memory or main memory which serious performance impact on small function calls. Instead, we use an iterative version of Viterbi parse extraction which uses pre-allocated array storage to store its output, and such that the partially-complete output array encodes all the information the algorithm needs to proceed - i.e. the output array is also the algorithm's work queue.

Ignoring unaries for the moment, a binary parse tree for a sentence of length n has $2n-1$ nodes, including preterminals, internal nodes, and the root. We can uniquely represent a tree as an array with $2n-1$ elements. In this representation, each index corresponds to a node in prefix (depth-first) order. For example, the root is always at position 0, and the second node will correspond to the root's left child. If this second node has a left child, it will be the third node, otherwise the third node will be the second's right sibling.

We can uniquely identify the topology of the tree by storing the "width" of each node in this array, where the width is the number of words governed by that constituent. For a node at position p, its left child will always be at $p+1$, and its right child will always be at $p + 2 \cdot w_\ell$, where w_ℓ is the width of the left child. The symbol for each node can obviously be stored with the height. For unaries, we require exactly one unary rule per node, with the possibility that it is the identity rule, and so we store two nodes: one for the "pre-unary" symbol, and one for the "post-unary." (Identity unary transitions are removed in post-processing.)

Algorithm 1 Non-recursive Viterbi implementation. The algorithm proceeds left-to-right in depth-first order along the array representing the tree.

```
Input: Sentence length n, parse chart V[i,j]
Output: Array tree of size 2·n−2
  tree[0].preunary ← ROOT
  tree[0].width ← n
  i ← 0                    ▷ Current leftmost position for span
  for p ← 0 to 2·n−2 do
      j ← i + tree[p].width      ▷ Rightmost position
      postu ← BestUnary(V, tree[p].preunary, i, j)
      tree[p].postunary ← parent
      if tree[p].width = 1 then
          i ← i + 1
      else
          lc, rc, k ← BestBinary(V, parent, i, j)
          tree[p + 1].preunary ← lc
          tree[p + 1].width ← k − i
          tree[p + 2·(k−i)].width ← j - k
          tree[p + 2·(k−i)].preunary ← rc
      end if
  end for
```

Armed with this representation, we are ready to describe algorithm 1. The algorithm proceeds in

left-to-right order along the array. First, the symbol of the root is recorded. Then, for each node in the tree, we search for the best unary rule continuing it. If the node is a terminal, then no more nodes can contain the current word, and so we advance the position of the left most child. Otherwise, if the node is a non-terminal, we then find its left and right children, entering their respective symbols and widths into the array representing the tree.

The GPU implementation follows the algorithm outline above although is somewhat technical. Each parse tree is handled by a separate thread block (thread blocks are groups of threads that can communicate through shared memory, and run on a single SMX). Each thread block includes a number of threads which are used to rapidly (in partly parallel fashion) iterate through rulesets and symbol vectors for the BestBinary and BestUnary operations using coalesced memory accesses. Each thread block first loads the complete set of L and R scores for the current split being explored. Recall that these are in consecutive memory locations using the "span-major" ordering, so these loads are coalesced. Then the thread block parallel-iterates through the rules for the current parent symbol, which will be in a contiguous block of memory since the rules are sorted by parent symbol, and again are coalesced. The thread block therefore needs storage for all the L, R symbol scores and in addition working storage proportional to the number of threads (to hold the best child symbol and its score from each thread). The number of threads is chosen to maximize speed: too few will cause each thread to do more work and to run more slowly. Too many will limit the number of thread blocks (since the total threads concurrently running on an SMX is 1024) that can run concurrently. We found 128 to be optimum.

With these techniques, Viterbi search consumes approximately 1% of the parser's running time. Its throughput is around 10 Gflops, and it is 50-100x faster than a CPU reference implementation.

7 Experiments

The parser was tested in an desktop computer with one Intel E5-2650 processor, 64 GB ram, and 2 GTX-690 dual GPUs (effectively 4 GTX-680 GPUs). The high-level parser code is written in a matrix library in the Scala language, which access GPU code through JNI and using the JCUDA wrapper library for CUDA™.

XX-kernel throughput was 900 Gflops per GPU for sum-product calculation (which uses a single FMA for most rules) and 700 Gflops per GPU for max-sum calculations (which requires two instructions for most rules). Net parser throughput including max-sum CKY evaluation, Viterbi scoring traspose-copy etc was between 500 and 600 gigaflops per GPU, or about 2 teraflops total. Parsing max-length-30 sentences from the Penn Treebank test set ran at 250 sentences/sec per GPU, or 1000 sentences/sec total. General sentences were parsed at about half this rate, 120 sentences/sec per GPU, or 480 sentences/sec for the system.

8 Conclusions and Future Work

We described a new approach to GPU constituency parsing with surprisingly fast performance, close to the theoretical limits of the GPU and similar to dense matrix multiplication which achieves the devices highest practical throughput. The resulting parser parses 1000 length-30 sentences per second in a 4-GPU computer. The parser has immediate application to parsing and eventually to parser training. The two highest-priority extensions are:

Addition of pruning: coarse-to-fine score pruning should be applicable to our GPU design as it is to CPU parsers. GPU pruning will not be as granular as CPU pruning and is unlikely to yield as large speedups (4-5 orders of magnitude are common for CPU parser pruning). But on the other hand, we hardly need speedups that large, and 1-2 orders of magnitude would be very useful.

Direct generation of assembly code. Currently our code generator produces ($>$ 1.7 million lines, about same as the number of rules) C source code which must be compiled into GPU binary code. While it takes only 8 seconds to generate the source code, it takes more than an hour to compile it. The compiler evidently applies a number of optimizations that we cannot disable, and this takes time. This is an obstacle to e.g. using this framework to train a parser where there would be frequent updates to the grammar. However, since symbol variables correspond almost one-to-one with registers (modulo

lifetime overlap and reuse, which our code generator is slightly better at than the compiler), there is no reason for our code generator not to generate assembly code directly. Presumably assembly code is much faster to translate into kernel modules than C source, and hopefully this will lead to much faster kernel generation.

8.1 Code Release

The code will be released under a BSD-style open source license once its dependencies are fully integrated. Pre- and Final releases will be here `https://github.com/jcanny/BIDParse`

References

Anonymous. 2012. CUDA C PROGRAMMING GUIDE. Technical Report PG-02829-001-v5.0. Included with CUDA™ Toolkit.

Anonymous. 2013. NVIDIA's next generation CUDA compute architecture: Kepler™ GK110. Technical report. Included with CUDA™ Tootkit.

Mark Johnson. 2011. Parsing in parallel on multiple cores and gpus. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 29–37, Canberra, Australia, December.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, Michigan, June. Association for Computational Linguistics.

NVIDIA. 2012. private communication.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Youngmin Yi, Chao-Yue Lai, Slav Petrov, and Kurt Keutzer. 2011. Efficient parallel cky parsing on gpus. In *Proceedings of the 2011 Conference on Parsing Technologies*, Dublin, Ireland, October.

Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing

Angeliki Lazaridou, Eva Maria Vecchi and **Marco Baroni**
Center for Mind/Brain Sciences
University of Trento, Italy
first.last@unitn.it

Abstract

In this work, we argue that measures that have been shown to quantify the degree of semantic plausibility of phrases, as obtained from their compositionally-derived distributional semantic representations, can resolve syntactic ambiguities. We exploit this idea to choose the correct parsing of NPs (e.g., (*live fish*) *transporter* rather than *live* (*fish transporter*)). We show that our plausibility cues outperform a strong baseline and significantly improve performance when used in combination with state-of-the-art features.

1 Introduction

Live fish transporter: A transporter of live fish or rather a fish transporter that is not dead? While our intuition, based on the meaning of this phrase, prefers the former interpretation, the Stanford parser, which lacks semantic features, incorrectly predicts the latter as the correct parse.[1] The correct syntactic parsing of sentences is clearly steered by semantic information (as formal syntacticians have pointed out at least since Fillmore (1968)), and consequently the semantic plausibility of alternative parses can provide crucial evidence about their validity.

An emerging line of parsing research capitalizes on the advances of compositional distributional semantics (Baroni and Zamparelli, 2010; Guevara, 2010; Mitchell and Lapata, 2010; Socher et al., 2012). Information related to compositionally-derived distributional representations of phrases is integrated at various stages of the parsing process to improve overall performance.[2] We are aware of two very recent studies exploiting the semantic information provided by distributional models to resolve syntactic ambiguity: Socher et al. (2013) and Le et al. (2013).

Socher et al. (2013) present a recursive neural network architecture which jointly learns semantic representations and syntactic categories of phrases. By annotating syntactic categories with their distributional representation, the method emulates lexicalized approaches (Collins, 2003) and captures similarity more flexibly than solutions based on hard clustering (Klein and Manning, 2003; Petrov et al., 2006). Thus, their approach mainly aims at improving parsing by capturing a richer, data-driven categorial structure.

On the other hand, Le et al. (2013) work with the output of the parser. Their hypothesis is that parses that lead to less semantically plausible interpretations will be penalized by a reranker that looks at the composed semantic representation of the parse. Their method achieves an improvement of 0.2% in F-score. However, as the authors also remark, because of their experimental setup, they cannot conclude that the improvement is truly due to the semantic composition component, a crucial issue that is deferred to further investigation.

This work aims at corroborating the hypothesis that the semantic plausibility of a phrase can indeed determine its correct parsing. We develop a system based on simple and intuitive measures, ex-

[1] http://nlp.stanford.edu:8080/parser/index.jsp

[2] Distributional representations approximate word and phrase meaning by vectors that record the contexts in which they are likely to appear in corpora; for a review see, e.g., Turney and Pantel (2010).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1908–1913,
Seattle, Washington, USA, 18-21 October 2013.

Type of NP	#	Example
A (N N)	1296	*local phone company*
(A N) N	343	*crude oil sector*
N (N N)	164	*miracle home run*
(N N) N	424	*blood pressure medicine*
Total	2227	-

Table 1: NP dataset

tracted from the compositional distributional representations of phrases, that have been shown to correlate with semantic plausibility (Vecchi et al., 2011).

We develop a controlled experimental setup, focusing on a single syntactic category, that is, noun phrases (NP), where our task can be formalized as (left or right) bracketing. Unlike previous work, we compare our compositional semantic component against features based on n-gram statistics, which can arguably also capture some semantic information in terms of frequent occurrences of meaningful phrases. Inspired by previous literature demonstrating the power of metrics based on Pointwise Mutual Information (PMI) in NP bracketing (Nakov and Hearst, 2005; Pitler et al., 2010; Vadas and Curran, 2011), we test an approach exploiting PMI features, and show that plausibility features relying on composed representations can significantly boost accuracy over PMI.

2 Setup

Noun phrase dataset To construct our dataset, we used the Penn TreeBank (Marcus et al., 1993), which we enriched with the annotation provided by Vadas and Curran (2007a), since the original treebank does not distinguish different structures inside the NPs and always marks them as right bracketed, e.g., *local (phone company)* but also *blood (pressure medicine)*. We focus on NPs formed by three elements, where the first can be an adjective (A) or a noun (N), the other two are nouns. Table 1 summarizes the characteristics of the dataset.[3]

Distributional semantic space As our source corpus we use the concatenation of ukWaC, the English Wikipedia (2009 dump) and the BNC, with a total of about 2.8 billion tokens.[4] We collect co-occurrence statistics for the top 8K Ns and 4K As, plus any other word from our NP dataset that was below this rank. Our context elements are composed of the top 10K content words (adjectives, adverbs, nouns and verbs). We use a standard bag-of-words approach, counting within-sentence collocates for every target word. We apply (non-negative) Pointwise Mutual Information as weighting scheme and dimensionality reduction using Non-negative Matrix Factorization, setting the number of reduced-space dimensions to 300.[5]

Composition functions We experiment with various composition functions, chosen among those sensitive to internal structure (Baroni and Zamparelli, 2010; Guevara, 2010; Mitchell and Lapata, 2010), namely dilation (**dil**), weighted additive (**wadd**), lexical function (**lexfunc**) and full additive (**fulladd**).[6] For model implementation and (unsupervised) estimation, we rely on the freely available *DISSECT* toolkit (Dinu et al., 2013).[7] For all methods, vectors were normalized before composing, both in training and in generation. Table 2 presents a summary description of the composition methods we used.

Following previous literature (Mitchell and Lapata, 2010), and the general intuition that adjectival modification is quite a different process from noun combination (Gagné and Spalding, 2009; McNally, 2013), we learn different parameters for noun-noun (NN) and adjective-noun (AN) phrases. As an example of the learned parameters, for the **wadd** model the ratio of parameters w_1 and w_2 is 1:2 for ANs, whereas for NNs it is almost 1:1, confirming the intuition that a non-head noun plays a stronger role in composition than an adjective modifier.

[3]The dataset is available from: `http://clic.cimec.unitn.it/composes`

[4]`http://wacky.sslmit.unibo.it`, `http://en.wikipedia.org`, `http://www.natcorp.ox.ac.uk`

[5]For tuning the parameters of the semantic space, we computed the correlation of cosines produced with a variety of parameter settings (SVD/NMF/no reduction, PMI/Local MI/raw counts/log transform, 150 to 300 dimensions in steps of 50) with the word pair similarity ratings in the MEN dataset: `http://clic.cimec.unitn.it/~elia.bruni/MEN`

[6]We do not consider the popular multiplicative model, as it produces identical representations for NPs irrespective of their internal structure.

[7]`http://clic.cimec.unitn.it/composes/toolkit/`

Model	Composition function	Parameters
wadd	$w_1\vec{u} + w_2\vec{v}$	w_1, w_2
dil	$\|\|\vec{u}\|\|_2^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u}$	λ
fulladd	$W_1\vec{u} + W_2\vec{v}$	$W_1, W_2 \in \mathbf{R}^{m\times m}$
lexfunc	$A_u\vec{v}$	$A_u \in \mathbf{R}^{m\times m}$

Table 2: Composition functions of inputs (u, v).

Recursive composition In this study we also experiment with recursive composition; to the best of our knowledge, this is the first time that these composition functions have been explicitly used in this manner. For example, given the left bracketed NP *(blood pressure) medicine*, we want to obtain its compositional semantic representation, $\overrightarrow{blood\ pressure\ medicine}$. First, *basic composition* is applied, in which $\overrightarrow{blood}$ and $\overrightarrow{pressure}$ are combined with one of the composition functions. Following that, we apply *recursive composition*; the output of basic composition, i.e., $\overrightarrow{blood\ pressure}$, is fed to the function again to be composed with the representation of $\overrightarrow{medicine}$.

The latter step is straightforward for all composition functions except **lexfunc** applied to left-bracketed NPs, where the first step should return a matrix representing the left constituent (*blood pressure* in the running example). To cope with this nuisance, we apply the **lexfunc** method to *basic composition* only, while recursive representations are derived by summing (e.g., $\overrightarrow{blood\ pressure}$ is obtained by multiplying the *blood* matrix by the *pressure* vector, and it is then summed to $\overrightarrow{medicine}$).

3 Experiments

Semantic plausibility measures We use measures of semantic plausibility computed on composed semantic representations introduced by Vecchi et al. (2011). The rationale is that the correct (wrong) bracketing will lead to semantically more (less) plausible phrases. Thus, a measure able to discriminate semantically plausible from implausible phrases should also indicate the most likely parse. Considering, for example, the alternative parses of *miracle home run*, we observe that *home run* is a more semantically plausible phrase than *miracle home*. Furthermore, we might often refer to a baseball player's miracle home run, but we doubt that even a miracle home can run! Given the composed representation of an AN (or NN), Vecchi et al. (2011) define the following measures:

- *Density*, quantified as the average cosine of a phrase with its (top 10) nearest neighbors, captures the intuition that a deviant phrase should be isolated in the semantic space.
- *Cosine of phrase and head N* aims to capture the fact that the meaning of a deviant AN (or NN) will tend to diverge from the meaning of the head noun.
- *Vector length* should capture anomalous vectors.

Since length, as already observed by Vecchi et al., is strongly affected by independent factors such as input vector normalization and the estimation procedure, we introduce *entropy* as a measure of vector quality. The intuition is that meaningless vectors, whose dimensions contain mostly noise, should have high entropy.

NP Parsing as Classification Parsing NPs consisting of three elements can be treated as binary classification; given *blood pressure medicine*, we predict whether it is left- (*(blood pressure) medicine*) or right-bracketed (*blood (pressure medicine)*).

We conduct experiments using an SVM with Radial Basis Function kernel as implemented in the *scikit-learn* toolkit.[8] Our dataset is split into 10 folds in which the ratio between the two classes is kept constant. We tune the SVM complexity parameter *C* on the first fold and we report accuracy results on the remaining nine folds after cross-validation.

Features Given a composition function $\boldsymbol{f}$, we define the following feature sets, illustrated with the usual *blood pressure medicine* example, which are used to build different classifiers:

- $\boldsymbol{f_{basic}}$ consists of the semantic plausibility measures described above computed for the two-word phrases resulting from alternative bracketings, i.e., 3 measures for each bracketing, evaluated on *blood pressure* and *pressure medicine* respectively, for a total of 6 features.
- $\boldsymbol{f_{rec}}$ contains 6 features computed on the vectors resulting from the recursive compositions

[8] http://scikit-learn.org/

Features	Accuracy
$right$	65.6
pos	77.3
$lexfunc_{basic}$	74.6
$lexfunc_{rec}$	74.0
$lexfunc_{plausibility}$	76.2
$wadd_{basic}$	75.9
$wadd_{rec}$	78.2
$wadd_{plausibility}$	78.7
pmi	81.2
pmi+$lexfunc_{plausibility}$	82.9
pmi+$wadd_{plausibility}$	**85.6**

Table 3: Evaluation of feature sets from Section 3

(blood pressure) medicine and *blood (pressure medicine)*.

- $f_{plausibility}$ concatenates f_{basic} and f_{rec}.
- pmi contains the PMI scores extracted from our corpus for *blood pressure* and *pressure medicine*.[9]
- pmi + $f_{plausibility}$ concatenates pmi and $f_{plausibility}$.

Baseline Model Given the skewed bracketing distribution in our dataset, we implement the following majority baselines: *a*) $right$ classifies all phrases as right-bracketed; *b*) pos classifies NNN as left-bracketed (Lauer, 1995), ANN as right-bracketed.

4 Results and Discussion

Table 3 omits results for **dil** and **fulladd** since they were outperformed by the $right$ baseline. That **wadd**- and **lexfunc**-based plausibility features perform well above this baseline is encouraging, since it represents the typical default behaviour of parsers for NPs, although note that these features perform comparably to the pos baseline, which would be quite simple to embed in a parser (for English, at least). For both models, using both basic and recursive features leads to a boost in performance over basic features alone. Note that recursive features (f_{rec}) achieve at least equal or better performance than basic ones (f_{basic}). We expect indeed that in many cases the asymmetry in plausibility will be sharper when considering the whole NP rather than its sub-parts; a *pressure medicine* is still a conceivable concept, but *blood (pressure medicine)* makes no sense whatsoever. Finally, **wadd** outperforms both the more informative baseline pos and **lexfunc**. The difference between **wadd** and **lexfunc** is significant ($p < 0.05$)[10] only when they are trained with recursive composition features, probably due to our suboptimal adaptation of the latter to recursive composition (see Section 2).

The pmi approach outperforms the best plausibility-based feature set $wadd_{plausibility}$. However, the two make only a small proportion of common errors (29% of the total $wadd_{plausibility}$ errors, 32% for pmi), suggesting that they are complementary. Indeed the pmi + $wadd_{plausibility}$ combination significantly outperforms pmi alone ($p < 0.001$), indicating that plausibility features can improve NP bracketing on top of the powerful PMI-based approach. The same effect can also be observed in the combination of pmi + $lexfunc_{plausibility}$, which again significantly outperforms pmi alone ($p < 0.05$). This behaviour further suggests that the different types of errors are not a result of the parameters or type of composition applied, but rather highlights fundamental differences in the kind of information that PMI and composition models are able to capture.

One hypothesis is that compositional models are more robust for low-frequency NPs, for which PMI estimates will be less accurate; results on those low-frequency trigrams only (20% of the NP dataset, operationalized as those consisting of bigrams with frequency ≤ 100) revealed indeed that $wadd_{plausibility}$ performed 8.1% better in terms of accuracy than pmi.

5 Conclusion

Our pilot study showed that semantic plausibility, as measured on compositional distributional representations, can improve syntactic parsing of NPs. Our results further suggest that state-of-the-art PMI features and the ones extracted from compositional representations are complementary, and thus, when combined, can lead to significantly better results. Besides paving the way to a more general integration

[9] Several approaches to computing PMI for these purposes have been proposed in the literature including the *dependency model* (Lauer, 1995) and the *adjacency model* (Marcus, 1980). We implement the latter since it has been shown to perform better (Vadas and Curran, 2007b) on NPs extracted from Penn TreeBank.

[10] Significance values are based on t-tests.

of compositional distributional semantics in syntactic parsing, the proposed methodology provides a new way to evaluate composition functions.

The relatively simple-minded **wadd** approach outperformed more complex models such as **lexfunc**. We plan to experiment next with more linguistically motivated ways to adapt the latter to recursive composition, including hybrid methods where ANs and NNs are treated differently. We would also like to consider more sophisticated semantic plausibility measures (e.g., supervised ones), and apply them to other ambiguous syntactic constructions.

6 Acknowledgments

We thank Georgiana Dinu and German Kruszewski for helpful discussions and the reviewers for useful feedback. This research was supported by the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

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

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. DISSECT: DIStributional SEmantics Composition Toolkit. In *Proceedings of the System Demonstrations of ACL 2013*, Sofia, Bulgaria.

Charles Fillmore. 1968. The case for case. In Emmon Bach and Robert Harms, editors, *Universals in Linguistic Theory*, pages 1–89. Holt, Rinehart and Winston, New York.

Christina Gagné and Thomas Spalding. 2009. Constituent integration during the processing of compound words: Does it involve the use of relational structures? *Journal of Memory and Language*, 60:20–35.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430. Association for Computational Linguistics.

Mark Lauer. 1995. Corpus statistics meet the noun compound: Some empirical results. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, pages 47–54, Cambridge, MA.

Phong Le, Willem Zuidema, and Remko Scha. 2013. Learning from errors: Using vector-based compositional semantics for parse reranking. In *Proceedings of the ACL 2013 Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Mitchell P Marcus. 1980. *Theory of syntactic recognition for natural languages*. MIT press.

Louise McNally. 2013. Modification. In Maria Aloni and Paul Dekker, editors, *Cambridge Handbook of Semantics*. Cambridge University Press, Cambridge, UK. In press.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL*, pages 17–24, Stroudsburg, PA, USA.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*, pages 433–440, Stroudsburg, PA, USA.

Emily Pitler, Shane Bergsma, Dekang Lin, and Kenneth Church. 2010. Using web-scale n-grams to improve base NP parsing performance. In *Proceedings of the COLING*, pages 886–894, Beijing, China.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*, Sofia, Bulgaria.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

David Vadas and James Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *Proceedings of ACL*, pages 240–247, Prague, Czech Republic.

David Vadas and James R Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *Proceedings of the PACLING*, pages 104–112.

David Vadas and James R. Curran. 2011. Parsing noun phrases in the penn treebank. *Comput. Linguist.*, 37(4):753–809.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (Linear) maps of the impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the ACL Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, OR.

Learning Distributions over Logical Forms for Referring Expression Generation

Nicholas FitzGerald **Yoav Artzi** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{nfitz,yoav,lsz}@cs.washington.edu

Abstract

We present a new approach to referring expression generation, casting it as a density estimation problem where the goal is to learn distributions over logical expressions identifying sets of objects in the world. Despite an extremely large space of possible expressions, we demonstrate effective learning of a globally normalized log-linear distribution. This learning is enabled by a new, multi-stage approximate inference technique that uses a pruning model to construct only the most likely logical forms. We train and evaluate the approach on a new corpus of references to sets of visual objects. Experiments show the approach is able to learn accurate models, which generate over 87% of the expressions people used. Additionally, on the previously studied special case of single object reference, we show a 35% relative error reduction over previous state of the art.

1 Introduction

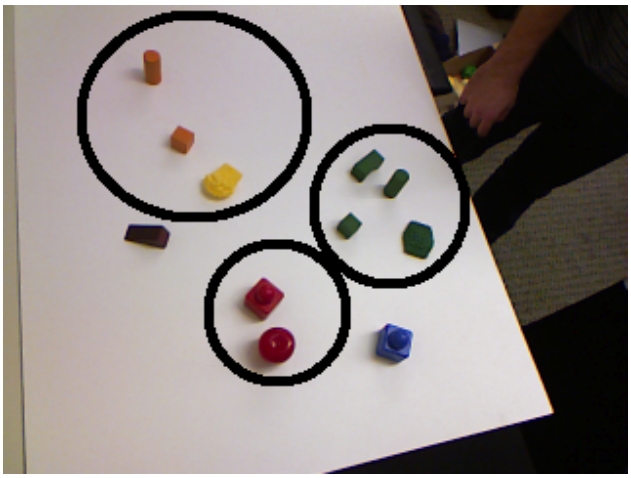
Understanding and generating natural language requires reasoning over a large space of possible meanings; while many statements might achieve the same goal in a certain situation, some are more likely to be used than others. In this paper, we model these preferences by learning distributions over situated meaning use.

We focus on the task of referring expression generation (REG), where the goal is to produce an expression which uniquely identifies a pre-defined object or set of objects in an environment. In practice, many such expressions can be produced. Figure 1 shows referring expressions provided by human subjects for a set of objects (Figure 1a), demonstrating variation in utterances (Figure 1b) and their corresponding meaning representations (Figure 1c). Although nearly a third of the people simply listed the colors of the desired objects, many other strategies were also used and no single option dominated. Learning to model such variation would enable systems to better anticipate what people are likely to say and avoid repetition during generation, by producing appropriately varied utterances themselves. With these goals in mind, we cast REG as a density estimation problem, where the goal is to learn a distribution over logical forms.

Learning such distributions is challenging. For a target set of objects, the number of logical forms that can be used to describe it grows combinatorially with the number of observable properties, such as color and shape. However, only a tiny fraction of these possibilities are ever actually used by people. We must learn to efficiently find these few, and accurately estimate their associated likelihoods.

We demonstrate effective learning of a globally normalized log-linear distribution with features to account for context dependence and communicative goals. We use a stochastic gradient descent algorithm, where the key challenge is the need to compute feature expectations over all possible logical forms. For that purpose, we present a multi-stage inference algorithm, which progressively constructs meaning representations with increasing complexity, and learns a pruning model to retain only those that are likely to lead to high probability expressions. This approach allows us to consider a large

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1914–1925,
Seattle, Washington, USA, 18-21 October 2013.



(a)

The green, red, orange and yellow toys.	(1)
The green, red, yellow, and orange objects.	(1)
The red, green, yellow and orange toys.	(1)
The red, yellow, orange and green objects.	(1)
All the green, red, yellow and orange toys.	(1)
All the yellow, orange, red and green objects.	(1)
All the pieces that are not blue or brown.	(2)
All items that are not brown or blue.	(2)
All items that are not brown or blue.	(2)
Everything that is not brown or blue.	(3)
Everything that is not purple or blue.	(3)
All but the black and blue ones.	(4)
Any toy but the blue and brown toys.	(4)
Everything that is green, red, orange or yellow.	(5)
All objects that are not triangular or blue.	(6)
Everything that is not blue or a wedge.	(7)
Everything that is not a brown or blue toy.	(8)
All but the blue piece and brown wedge.	(9)
Everything except the brown wedge and the blue object.	(10)
All pieces but the blue piece and brown triangle shape.	(11)

(b)

$\hat{P}(z\|S,G)$	z	
0.30	$\iota(\lambda x.(yellow(x) \vee orange(x) \vee red(x) \vee green(x)) \wedge object(x) \wedge plu(x))$	(1)
0.15	$\iota(\lambda x.\neg(brown(x) \vee blue(x)) \wedge object(x) \wedge plu(x))$	(2)
0.10	$Every(\lambda x.\neg(brown(x) \vee blue(x)) \wedge object(x) \wedge sg(x))$	(3)
0.10	$Every(\lambda x.object(x) \wedge sg(x)) \setminus [\iota(\lambda x.(blue(x) \vee brown(x)) \wedge object(x) \wedge plu(x)]$	(4)
0.05	$Every(\lambda x.(yellow(x) \vee orange(x) \vee red(x) \vee green(x)) \wedge object(x) \wedge sg(x))$	(5)
0.05	$\iota(\lambda x.(triangle(x) \vee blue(x)) \wedge object(x) \wedge plu(x))$	(6)
0.05	$Every(\lambda x.object(x) \wedge sg(x) \neg(blue(x) \vee equal(x, \mathcal{A}(\lambda y.triangle(y) \wedge sg(y)))))$	(7)
0.05	$Every(\lambda x.object(x) \wedge sg(x) \wedge \neg equal(x, \mathcal{A}(\lambda y.(brown(y) \vee blue(y)) \wedge object(y) \wedge sg(y))))$	(8)
0.05	$Every(\lambda x.object(x) \wedge sg(x)) \setminus [\iota(\lambda x.(blue(x) \wedge object(x) \wedge sg(x)) \vee (brown(x) \wedge triangle(x) \wedge sg(x))]$	(9)
0.05	$Every(\lambda x.object(x) \wedge sg(x)) \setminus [\iota(\lambda x.brown(x) \wedge triangle(x) \wedge sg(x)) \cup \iota(\lambda y.blue(y) \wedge object(y) \wedge sg(x))]$	(10)
0.05	$\iota(\lambda x.object(x) \wedge plu(x)) \setminus [\iota(\lambda x.(blue(x) \wedge object(x) \wedge sg(x)) \vee (brown(x) \wedge triangle(x) \wedge object(x) \wedge sg(x))]$	(11)

(c)

Figure 1: An example scene from our object selection dataset. Figure 1a shows the image shown to subjects on Amazon Mechanical Turk. The target set G is the circled objects. Figure 1b shows the 20 sentences provided as responses. Figure 1c shows the empirical distribution $\hat{P}(z|G,S)$ for this scene, estimated by labeling the sentences in Figure 1b. The correspondence between a sentence in 1b and its labeled logical expression in 1c is indicated by the number in parentheses. Section 5.1 presents a discussion of the space of possible logical forms.

set of possible meanings, while maintaining computational tractability.

To represent meaning we build on previous approaches that use lambda calculus (Carpenter, 1997; Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2013b). We extend these techniques by modeling the types of plurality and coordination that are prominent in expressions which refer to sets.

We also present a new corpus for the task of referring expression generation.[1] While most previous REG data focused on naming single objects, to the best of our knowledge, this is the first corpus with sufficient coverage for learning to name sets of objects. Experiments demonstrate highly accurate learned models, able to generate over 87% of the expressions people used. On the previously studied special case of single object reference, we achieve state-of-the-art performance, with over 35% relative error reduction over previous state of the art (Mitchell et al., 2013).

2 Related Work

Referring expression generation has been extensively studied in the natural language generation

[1]The corpus was collected using Amazon Mechanical Turk and is available on the authors' websites.

community, dating as far back as SHRDLU (Winograd, 1972). Most work has built on variations of the Incremental Algorithm (Dale and Reiter, 1995), a deterministic algorithm for naming single objects that constructs conjunctive logical expressions. REG systems are used in generation pipelines (Dale and Reiter, 2000) and are also commonly designed to be cognitively plausible, for example by following Gricean maxims (Grice, 1975). Krahmer and van Deemter (2012) and van Deemter et al. (2012a) survey recent literature on REG.

Different approaches have been proposed for generating referring expressions for sets of objects. Van Deemter (2002) extended the Incremental Algorithm to allow disjunction and negation, enabling reference to sets. Further work attempted to resolve the unnaturally long expressions which could be generated by this approach (Gardent, 2002; Horacek, 2004; Gatt and van Deemter, 2007). Later, description logic was used to name sets (Areces et al., 2008; Ren et al., 2010). All of these algorithms are manually engineered and deterministic.

In practice, human utterances are surprisingly varied, loosely following the Gricean ideals (van Deemter et al., 2012b). Much recent work in REG has identified the importance of modeling the variation observed in human-generated referring expressions (Viethen and Dale, 2010; Viethen et al., 2013; van Deemter et al., 2012b; Mitchell et al., 2013), and some approaches have applied machine-learning techniques to single-object references (Viethen and Dale, 2010; Mitchell et al., 2011a,b). Recently, Mitchell et al. (2013) introduced a probabilistic approach for conjunctive descriptions of single objects, which will provide a comparison baseline for experiments in Section 8. To the best of our knowledge, this paper presents the first learned probabilistic model for referring expressions defining sets, and is the first effort to treat REG as a density estimation problem.

REG is related to *content selection*, which has been studied for generating text from databases (Konstas and Lapata, 2012), event streams (Chen et al., 2010), images (Berg et al., 2012; Zitnick and Parikh, 2013), and text (Barzilay and Lapata, 2005; Carenini et al., 2006). However, most approaches to this problem output bags of concepts, while we construct full logical expressions, allowing our approach to capture complex relations between attributes.

Finally, our approach to modeling meaning using lambda calculus is related to a number of approaches that used similar logical representation in various domains, including database query interfaces (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005, 2007), natural language instructions (Chen and Mooney, 2011; Matuszek et al., 2012b; Kim and Mooney, 2012; Artzi and Zettlemoyer, 2013b), event streams (Liang et al., 2009; Chen et al., 2010), and visual descriptions (Matuszek et al., 2012a; Krishnamurthy and Kollar, 2013). Our use of logical forms follows this line of work, while extending it to handle plurality and coordination, as described in Section 4.1. In addition, lambda calculus was shown to enable effective natural language generation from logical forms (White and Rajkumar, 2009; Lu and Ng, 2011). If combined with these approaches, our approach would allow the creation of a complete REG pipeline.

3 Technical Overview

Task Let $\mathcal{Z}$ be a set of logical expressions that select a target set of objects G in a world state S, as formally defined in Section 5.1. We aim to learn a probability distribution $P(z \mid S, G)$, with $z \in \mathcal{Z}$.

For example, in the referring expressions domain we work with, the state $S = \{o_1, \ldots, o_n\}$ is a set of n objects o_i. Each o_i has three properties: color, shape and type. The target set $G \subseteq S$ is the subset of objects to be described. Figure 1a shows an example scene. The world state S includes the 11 objects in the image, where each object is assigned color (*yellow, green* ...), shape (*cube, cylinder* ...) and type (*broccoli, apple* ...). The target set G contains the circled objects. Our task is to predict a distribution which closely matches the empirical distribution in Figure 1c.

Model and Inference We model $P(z|S, G)$ as a globally normalized log-linear model, using features of the logical form z, and its execution with respect to S and G. Since enumerating all $z \in \mathcal{Z}$ is intractable, we develop an approximate inference algorithm which constructs a high quality candidate set, using a learned pruning model. Section 5.2 describes the globally scored log-linear model. Sec-

tion 5.3 presents a detailed description of the inference procedure.

Learning We use stochastic gradient descent to learn both the global scoring model and the explicit pruning model, as described in section 6. Our data consists of human-generated referring expressions, gathered from Amazon Mechanical Turk. These sentences are automatically labelled with logical forms with a learned semantic parser, providing a stand-in for manually labeled data (see Section 7).

Evaluation Our goal is to output a distribution that closely matches the distribution that would be produced by humans. We therefore evaluate our model with gold standard labeling of crowd-sourced referring expressions, which are treated as samples from the implicit distribution we are trying to model. The data and evaluation procedure are described in Section 7. The results are presented in Section 8.

4 Modeling Referring Expressions

4.1 Semantic Modeling

Our semantic modeling approach uses simply-typed lambda-calculus following previous work (Carpenter, 1997; Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2013b), extending it in one important way: we treat *sets* of objects as a primitive type, rather than individuals. This allows us to model plurality, cardinality, and coordination for the language observed in our data, and is further motivated by recent cognitive science evidence that sets and their properties are represented as single units in human cognition (Scontras et al., 2012).

Plurals Traditionally, noun phrases are identified with the entity-type $\mathtt{e}$ and pick out individual objects (Carpenter, 1997). This makes it difficult to interpret plural noun-phrases which pick out a set of objects, like "The red cubes". Previous approaches would map this sentence to the same logical expression as the singular "The red cube", ignoring the semantic distinction encoded by the plural.

Instead, we define the primitive entity $\mathtt{e}$ to range over sets of objects. $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions are therefore functions from sets to a truth-value. These are used in two ways, modeling both *distributive* and *collective* predicates (cf. Stone, 2000):

1. *Distributive* predicates are $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions which will return $true$ if every individual in the set has a given property. For example, the expression $\lambda x.red(x)$ will be true for all sets which contain only objects for which the value red is *true*.
2. *Collective* predicates are $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions which indicate a property of the set itself. For example, in the phrase "the two cubes", "two" corresponds to the expression $\lambda x.cardinality_2(x)$ which will return $true$ only for sets which have exactly two members.

We define semantic plurality in terms of two special *collective* predicates: sg for singular and plu for plural. For examples, "cube" is interpreted as $\lambda x.cube(x) \wedge sg(x)$, whereas "cubes" is interpreted as $\lambda x.cube(x) \wedge plu(x)$. The sg predicate returns $true$ only for singleton sets. The plu predicate returns $true$ for sets that contain two or more objects.

We also model three kinds of determiners, functional-type $\langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{e} \rangle$-type expressions which select a single set from the power-set represented by their $\langle \mathtt{e}, \mathtt{t} \rangle$-type argument. The definite determiner "the" is modeled with the predicate ι, which resolves to the maximal set amongst those licensed by its argument. The determiner $Every$ only accepts $\langle \mathtt{e}, \mathtt{t} \rangle$-type arguments that define singleton sets (i.e. the argument includes the sg predicate) and returns a set containing the union of these singletons. For example, although "red cube" is a singular expression, "Every red cube" refers to a set. Finally, the indefinite determiner "a" is modeled with the logical constant $\mathcal{A}$, which picks a singleton set by implicitly introducing an existential quantifier (Artzi and Zettlemoyer, 2013b).[2]

Coordination Two types of coordination are prominent in set descriptions. The first is attribute coordination, which is typically modeled with the boolean operators: $\wedge$ for conjunction and $\vee$ for disjunction. For example, the phrase "the red cubes and green rectangle" involves a disjunction that joins two conjunctive expressions, both within the scope of the definite determiner: $\iota(\lambda x.(red(x) \wedge cube(x) \wedge plu(x)) \vee (green(x) \wedge rectangle(x) \wedge sg(x)))$.

[2] This treatment of the indefinite determiner is related to generalized skolem terms as described by Steedman (2011).

The second kind of coordination, a new addition of this work, occurs when two sets are coordinated. This can either be set union ($\cup$) as in the phrase "The cubes and the rectangle" ($\iota(\lambda x.cube(x) \wedge plu(x)) \cup \iota(\lambda x.rectangle(x) \wedge sg(x))$), or set difference ($\setminus$) as in the phrase "All blocks except the green cube": ($\iota(\lambda x.object(x) \wedge plu(x)) \setminus \iota(\lambda x.green(x) \wedge cube(x) \wedge sg(x))$).

4.2 Visual Domain

Objects in our scenes are labeled with attribute values for four attribute types: `color` (7 values, such as *red*, *green*), `shape` (9 values, such as *cube*, *sphere*), `type` (16 values, such as *broccoli*, *apple*) and a special `object` property, which is *true* for all objects. The special `object` property captures the role of descriptions that are true for all objects, such as "toy" or "item". Each of these 33 attribute values corresponds to an $\langle \mathtt{e}, \mathtt{t} \rangle$-type predicate.

5 Model and Inference

In this section, we describe our approach to modeling the probability $P(z \mid S, G)$ of a logical form $z \in \mathcal{Z}$ that names a set of objects G in a world S, as defined in Section 3. We first define $\mathcal{Z}$ (Section 5.1), and then present the distribution (Section 5.2) and an approximate inference approach that makes use of a learned pruning model (Section 5.3).

5.1 Space of Possible Meanings

The set $\mathcal{Z}$ defines logical expressions that we will consider for picking the target set G in state S. In general, we can construct infinitely many such expressions. For example, every $z \in \mathcal{Z}$ can be trivially extended to form a new candidate z' for $\mathcal{Z}$ by adding a true clause to any conjunct it contains. However, the vast majority of such expressions are overly complex and redundant, and would never be used in practice as a referring expression.

To avoid this explosion, we limit the type and complexity of the logical expressions that are included in $\mathcal{Z}$. We consider only `e`-type expressions, since they name sets, and furthermore only include expressions that name the desired target set G.[3] We also limit the overall complexity of each $z \in \mathcal{Z}$, to contain not more than M logical constants.

To achieve these constraints, we define an inductive procedure for enumerating $\mathcal{Z}$, in order of complexity. We first define $\mathcal{A}_j$ to be the set of all `e`- and $\langle \mathtt{e}, \mathtt{t} \rangle$-type expressions that contain exactly j logical constants. Figure 2 presents five rules that can be used to construct $\mathcal{A}_j$ by induction, for $j = 1, \ldots, \infty$, by repeatedly adding new constants to expressions in $\mathcal{A}_{j'}$ for $j' < j$. Intuitively, $\mathcal{A}_j$ is the set of all complexity j expressions that can be used as subexpressions for higher complexity entires in our final set $\mathcal{Z}$. Next, we define $\mathcal{Z}_j$ to be the `e`-type expressions in $\mathcal{A}_j$ that name the correct set G. And, finally, $\mathcal{Z} = \cup_{j=1 \ldots M} \mathcal{Z}_j$ of all correct expressions up to a maximum complexity of M.

This construction allows for a finite $\mathcal{Z}$ with good

- $p : \langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{e} \rangle, e_1 : \langle \mathtt{e}, \mathtt{t} \rangle \rightarrow p(e_1) : \mathtt{e}$

  e.g.

$p = \iota : \langle \langle \mathtt{e}, \mathtt{t} \rangle, \mathtt{e} \rangle$
$e_1 = \lambda x.cube(x) \wedge sg(x) : \langle \mathtt{e}, \mathtt{t} \rangle$
$\iota(\lambda x.cube(x) \wedge sg(x)) : \mathtt{e}$

- $p : \langle \mathtt{t}, \mathtt{t} \rangle, e_1 : \langle \mathtt{e}, \mathtt{t} \rangle \rightarrow \lambda x.p(e_1(x)) : \langle \mathtt{e}, \mathtt{t} \rangle$

  e.g.

$p = \neg : \langle \mathtt{t}, \mathtt{t} \rangle$
$e_1 = \lambda x.red(x) : \langle \mathtt{e}, \mathtt{t} \rangle$
$\lambda x. \neg(red(x)) : \langle \mathtt{e}, \mathtt{t} \rangle$

- $p : \langle \mathtt{e}, \langle \mathtt{e}, \mathtt{t} \rangle \rangle, e_1 : \mathtt{e} \rightarrow \lambda x.(p(x))(e_1)$

  e.g.

$p = equal : \langle \mathtt{e}, \langle \mathtt{e}, \mathtt{t} \rangle \rangle$
$e_1 = \mathcal{A}(\lambda y.cube(y) \wedge sg(y)) : \mathtt{e}$
$\lambda x.equal(x, \mathcal{A}(\lambda y.cube(y) \wedge sg(y)))$

- $p : \langle \mathtt{e}, \langle \mathtt{e}, \mathtt{e} \rangle \rangle, e_1 : \mathtt{e}, e_2 : \mathtt{e} \rightarrow (p(e_1))(e_2) : \mathtt{e}$

  e.g.

$p = \setminus : \langle \mathtt{e}, \langle \mathtt{e}, \mathtt{e} \rangle \rangle$
$e_1 = \iota(\lambda x.cube(x) \wedge plu(x)) : \mathtt{e}$
$e_2 = Every(\lambda x.object(x) \wedge sg(x)) : \mathtt{e}$
$Every(\lambda x.object(x) \wedge sg(x)) \setminus$ $\iota(\lambda x.cube(x) \wedge plu(x)) : \mathtt{e}$

- $p : \langle \mathtt{t}, \langle \mathtt{t}, \mathtt{t} \rangle \rangle, e_1 : \langle \mathtt{e}, \mathtt{t} \rangle, e_2 \langle \mathtt{e}, \mathtt{t} \rangle \rightarrow$
 $\lambda x.(p(e_1(x)))(e_2(x)) : \langle \mathtt{e}, \mathtt{t} \rangle$

  e.g.

$p = \wedge : \langle \mathtt{t}, \langle \mathtt{t}, \mathtt{t} \rangle \rangle$
$e_1 = \lambda x.red(x) : \langle \mathtt{e}, \mathtt{t} \rangle$
$e_2 = \lambda x.cube(x) : \langle \mathtt{e}, \mathtt{t} \rangle$
$\lambda x.red(x) \wedge cube(x) : \mathtt{e}$

Figure 2: The five rules used during generation. Each rule is a template which takes a predicate $p : t$ of type t and one or two arguments $e_i : t_i$, with type t_i. The output is the logical expression after the arrow $\rightarrow$, constructed using the inputs as shown.

[3] We do not attempt to model underspecified or otherwise incorrect expressions, although our model could handle this by considering all `e`-type expressions.

empirical coverage, as we will see in the experiments in Section 8. However, $\mathcal{Z}$ is still prohibitively large for the maximum complexities used in practise (for example $M = 20$). Section 5.3 presents an approach for learning models to prune $\mathcal{Z}$, while still achieving good empirical coverage.

5.2 Global Model

Given a finite $\mathcal{Z}$, we can now define our desired globally normalized log-linear model, conditioned on the state S and set of target objects G:

$$P_G(z \mid S, G; \theta) = \frac{1}{C} e^{\theta \cdot \phi(z,S,G)} \quad (1)$$

where $\theta \in \mathbb{R}^n$ is a parameter vector, $\phi(z, S, G) \in \mathbb{R}^n$ is a feature function and C is the normalization constant. Section 5.4 defines the features we use.

5.3 Pruning $\mathcal{Z}$

As motivated in Section 5.1, the key challenge for our global model in Equation 1 is that the set $\mathcal{Z}$ is too large to be explicitly enumerated. Instead, we designed an approach for learning to approximate $\mathcal{Z}$ with a subset of the highly likely entries, and use this subset as a proxy for $\mathcal{Z}$ during inference.

More specifically, we define a binary distribution that is used to classify whether each $a \in \mathcal{A}_j$ is likely to be used as a sub-expression in $\mathcal{Z}$, and prune each $\mathcal{A}_j$ to keep only the top k most likely entries. This distribution is a logistic regression model:

$$P_j(a \mid S, G; \pi_j) = \frac{e^{\pi_j \cdot \phi(a,S,G)}}{1 + e^{\pi_j \cdot \phi(a,S,G)}} \quad (2)$$

with features $\phi(a, S, G) \in \mathbb{R}^n$ and parameters $\pi_j \in \mathbb{R}^n$. This distribution uses the same features as the global model presented in Equation 1, which we describe in Section 5.4.

Together, the pruning model and global model define the distribution $\hat{P}(z \mid G, S; \theta, \Pi)$ over $z \in \mathcal{Z}$, conditioned on the world state S and target set G, and parameterized by both the parameters θ of the global model and the parameters $\Pi = \{\pi_1, \ldots, \pi_M\}$ of the pruning models.

5.4 Features

We use three kinds of features: logical expression structure features, situated features and a complexity feature. All features but the complexity feature are shared between the global model in Equation 1 and the pruning model in Equation 2. In order to avoid overly specific features, the attribute value predicates in the logical expressions are replaced with their attribute type (ie. $red \rightarrow color$). In addition, the special constants sg and plu are ignored when computing features.

In the following description of our features, all examples are computed for the logical expression $\iota(\lambda x.red(x) \wedge object(x) \wedge plu(x))$, with respect to the scene and target set in Figure 1a.

Structure Features We use binary features that account for the presence of certain structures in the logical form, allowing the model to learn common usage patterns.

- **Head Predicate** - indicator for use of a logical constant as a head predicate in every subexpression of the expression. A head predicate is the top-level operator of an expression. For example, the head predicate of the expression "$\lambda x.red(x) \wedge object(x)$" is "$\wedge$" and the head of $\lambda x.red(x)$ is red. For our running example, the head features are ι, $\wedge$, $color$, $object$.

- **Head-Predicate Bigrams and Trigrams** - head-predicate bigrams are defined to be the head predicate of a logical form, and the head predicate of one of its children. Trigrams are similarly defined. E.g. bigrams: $[\iota, \wedge]$, $[\wedge, color]$, $[\wedge, object]$, and trigrams: $[\iota, \wedge, red]$, $[\iota, \wedge, object]$.

- **Conjunction Duplicate** - this feature fires if a conjunctive expression contains duplicate subexpressions amongst its children.

- **Coordination Children** - this feature set indicates the presence of a coordination subexpression ($\wedge$, $\vee$, $\cup$ or $\setminus$) and the head expressions of all pairs and triples of its child expressions. E.g. $[\wedge; red, object]$.

Situated Features These features take into account the evaluation of the logical form z with respect to the state S and target set G. They capture common patterns between the target set G and the object groups named by subexpressions of z.

- **Head Predicate and Coverage** - this feature set indicates the head predicate of every sub-expression of the logical form, combined with a comparison between the execution of the sub-expression and the target set G. The possible values for this comparison (which we call the "coverage" of the expression with respect to G) are: EQUAL, SUBSET (SUB), SUPERSET (SPR), DISJOINT, ALL, EMPTY and OTHER. E.g. $[\iota, \texttt{SUB}]$, $[\wedge, \texttt{SUB}]$, $[color, \texttt{SUB}]$, $[object, \texttt{ALL}]$

- **Coordination Child Coverage** - this feature set indicates the head-predicate of a coordination subexpression, combined with the coverage of all pairs and triples of its child expressions. E.g. $[\wedge; \texttt{SUB}, \texttt{ALL}]$.

- **Coordination Child Relative Coverage** - this feature set indicates, for every pair of child subexpressions of coordination expressions in the logical form, the coverage of the child subexpressions relative to each other. The possible relative coverage values are: SUB-SPR, DISJOINT, OTHER. E.g. $[\wedge; \texttt{SUB-SPR}]$.

Complexity Features We use a single real-numbered feature to account for the complexity of the logical form. We define the complexity of a logical form to be the number of logical constants used. Our running example has a complexity of 4. This feature is only used in the global model, since the pruning model always considers logical expressions of fixed complexity.

6 Learning

Figure 3 presents the complete learning algorithm. The algorithm is online, using stochastic gradient descent updates for both the globally scored density estimation model and the learned pruning model. The algorithm assumes a dataset of the form $\{(Z_i, S_i, G_i) : i = 1 \dots n\}$ where each example scene includes a list of logical expressions Z_i, a world state S_i, and a target set of objects, G_i, which will be identified by the resulting logical expressions. The output is learned parameters for both the globally scored density estimation model θ, and for the learned pruning models Π.

Inputs: Training set $\{(Z_i, S_i, G_i) : i = 1 \dots n\}$, where Z_i is a list of logical forms, S_i is a world state, and G_i is a target set of objects. Number of iterations T. Learning rate α_0. Decay parameter c. Complexity threshold M, as described in Section 5.3.

Definitions: Let $\hat{P}(z \mid G_i, S_i; \theta, \Pi)$ be the predicted global probability from Equation 1. Let $\hat{P}_j(z \mid G_i, S_i; \pi_j)$ be the predicted pruning probability from Equation 2. Let $\hat{\mathcal{A}}_j$ be the set of all complexity-M logical expressions, after pruning (see Section 5.1). Let $SUB(j, z)$ be all complexity-j sub-expressions of logical expression z. Let $Q_i(z \mid S_i, G_i)$ be the empirical probability over $z \in \mathcal{Z}$, estimated from Z_i. Finally, let $\phi_i(z)$ be a shorthand for the feature function $\phi(z, S_i, G_i)$ as defined in Section 5.4.

Algorithm:

Initialize $\theta \leftarrow \vec{0}$, $\pi_j \leftarrow \vec{0}$ for $j = 1 \dots M$

For $t = 1 \dots T, i = 1 \dots n$:

Step 1: (Update Global Model)

a. Compute the stochastic gradient:
$\Delta_\theta \leftarrow E_{Q_i(z|S_i,G_i)}[\phi_i(z)] - E_{\hat{P}(z|G_i,S_i;\theta,\Pi)}[\phi_i(z)]$

b. Update the parameters:
$\gamma \leftarrow \frac{\alpha_0}{1+c\times\tau}$ where $\tau = i + t \times n$
$\theta \leftarrow \theta + \gamma\Delta_\theta$

Step 2: (Update Pruning Model)

For $j = 1 \dots M$

a. Construct a set of positive and negative examples:
$\mathcal{D}^+ \leftarrow \bigcup_{z \in Z_i} SUB(j, z)$.
$\mathcal{D}^- \leftarrow \hat{\mathcal{A}}_j \setminus \mathcal{D}^+$

b. Compute mini-batch stochastic gradient, normalizing for data skew:
$\Delta_{\pi_j} \leftarrow \frac{1}{|\mathcal{D}^+|}\sum_{z\in\mathcal{D}^+}(1 - P_j(z \mid S_i, G_i; \pi_j))\phi_i(z)$
$\quad - \frac{1}{|\mathcal{D}^-|}\sum_{z\in\mathcal{D}^-} P_j(z \mid S_i, G; \pi_j)\phi_i(z)$

c. Update complexity-j pruning parameters:
$\pi_j \leftarrow \pi_j + \gamma\Delta_{\pi_j}$

Output: θ and $\Pi = [\pi_1, \dots, \pi_M]$

Figure 3: The learning algorithm.

6.1 Global Model Updates

The parameters θ of the globally scored density estimation model are trained to maximize the log-likelihood of the data:

$$O_i = \log \prod_{z \in Z_i} P_G(z \mid S_i, G_i) \tag{3}$$

Taking the derivative of this objective with respect to θ yields the gradient in Step 1a of Figure 3. The marginals, $E_{\hat{P}(z|G_i,S_i;\theta,\Pi)}(\phi_i(z))$, are computed over the approximate finite subset of $\mathcal{Z}$ constructed with the inference procedure described in Section 5.3.

6.2 Pruning Model Updates

To update each of the M pruning models, we first construct a set of positive and negative examples (Step 2a). The positive examples, $\mathcal{D}^+$, include those sub-expressions which should be in the beam - these are all complexity j sub-expressions of logical expressions in Z_i. The negative examples, $\mathcal{D}^-$, include all complexity-j expressions constructed during beam search, minus those which are in $\mathcal{D}^+$. The gradient (Set 2b) is a binary mini-batch gradient, normalized to correct for data skew.

7 Experimental Setup

Data Collection Our dataset consists of 118 images, taken with a Microsoft Kinect camera. These are the same images used by Matuszek et al. (2012a), but we create multiple prompts for each image by circling different objects, giving 269 scenes in total. These scenes were shown to workers on Amazon Mechanical Turk[4] who were asked to imagine giving instructions to a robot and complete the sentence "Please pick up _____" in reference to the circled objects. Twenty referring expressions were collected for each scene, a total of 5380 expressions.

From this data, 43 scenes (860 expressions) were held-out for use in a test set. Of the remaining scenes, the sentences of 30 were labeled with logical forms. 10 of these scenes (200 expressions) are used as a labeled initialization set, and 20 are used as a development test set (400 expressions). A small number of expressions ($\sim$5%) from the labeled initial set were discarded, either because they did not correctly name the target set, or because they used very rare attributes (such as texture, or location) to name the target objects.

Surrogate Labeling To avoid hand labeling the large majority of the scenes, we label the data with a learned semantic parser (Zettlemoyer and Collins, 2005). We created a hand-made lexicon for the entire training set, which greatly simplifies the learning problem, and learned the parameters of the parsing moder on the 10-scene initialization set. The weights were then further tuned using semi-supervised techniques (Artzi and Zettlemoyer, 2011, 2013b) on the data to be labeled. Testing on the development set shows that this parser achieves roughly 95% precision and 70% recall.

Using this parser, we label the sentences in our training set. We only use scenes where at least 15 sentences were successfully parsed. This gives a training set of 141 scenes (2587 expressions). Combining the automatically labeled training set with the hand-labelled initialization, development and held-out data, our labelled corpus totals 3938 labeled expressions. By contrast, the popular TUNA furniture sub corpus (Gatt et al., 2007) contains 856 descriptions of 20 scenes, and although some of these refer to sets, these sets contain two objects at most.

Framework Our experiments were implemented using the University of Washington Semantic Parsing Framework (Artzi and Zettlemoyer, 2013a).

Hyperparameters Our inference procedure requires two hyperparameters: M, the maximum complexity threshold, and k, the beam size. In practice, we set these to the highest possible values which still allow for training to complete in a reasonable amount of time (under 12 hours). M is set to 20, which is sufficient to cover 99.5% of the observed expressions. The beam-size k is 100 for the first three complexity levels, and 50 thereafter.

For learning, we use the following hyperparameters, which were tuned on the development set: learning rate $\alpha_0 = .25$, decay rate $c = .02$, number of epochs $T = 10$.

Evaluation Metrics Evaluation metrics used in REG research have assumed a system that produces a single output. Our goal is to achieve a distribution over logical forms that closely matches the distribution observed from human subjects. Therefore, we compare our learned model to the labeled test data with mean absolute error:

$$MAE = \frac{1}{2n}\sum_{i=1}^{n}\sum_{z \in \mathcal{Z}} |P(z \mid S_i, G_i) - Q(z \mid S_i, G_i)|$$

where Q is the empirical distribution observed in the training data. MAE measures the total probability mass which is assigned differently in the predicted distribution than in the empirical distribution. We use MAE as opposed to KL divergence or data likelihood as both of these measures are uninformative when the support of the two distributions differ.

[4] http://www.mturk.com

–	MAE	$\%_{dup}$	$\%_{uniq}$	Top1
VOA	39.7	98.2	92.5	**72.7**
GenX	**25.8** (5.0)	**100** (0)	**100** (0)	**72.7** (0)

Table 1: Single object referring expression generation results. Our approach (GenX) is compared to the approach from Mitchell et al. (2013) (VOA). Standard deviation over five shuffles of training set is reported in parentheses.

This metric is quite strict; small differences in the estimated probabilities over a large number of logical expressions can result in a large error, even if the relative ordering is quite similar. Therefore, we report the percentage of observed logical expressions which the model produces, either giving credit multiple times for duplicates ($\%_{dup}$) or counting each unique logical expression in a scene once ($\%_{uniq}$). Put another way, $\%_{dup}$ counts logical expression tokens, whereas $\%_{uniq}$ counts types. We also report the proportion of scenes where the most likely logical expression according to the model matched the most common one in the data (Top1).

Single Object Baseline In order to compare our method against the state of the art for generating referring expressions for single objects, we use the subset of our corpus where the target set is a single object. This sub-corpus consists of 44 scenes for training and 11 held out for testing.

For comparison we re-implemented the probabilistic Visual Objects Algorithm (VOA) of Mitchell et al. (2013). We refer the readers to the original paper for details of the approach. The parameters of the model were tuned on the training data: the prior likelihood estimates for each of the four attribute types (α_{att}) were estimated as the relative frequency of each attribute in the data. We pick the ordering of attributes and the length penalty, λ, from the cross-product of all possible $4!$ orderings and all integers on the range of [1,10], choosing the setting which results in the lowest average absolute error (AAE) on the training set. This process resulted in the following parameter settings: $\alpha_{color} = .916$, $\alpha_{shape} = .586$, $\alpha_{type} = .094$, $\alpha_{object} = .506$, AP ordering = $[type, shape, object, color]$, $\lambda = 4$. Inference was done using 10,000 samples per scene.

–	MAE	$\%_{dup}$	$\%_{uniq}$	Top1
Full GenX	**54.3** (4.5)	**87.4** (0.6)	**72.9** (1.1)	**52.6** (8.3)
NoPrune	71.8 (2.5)	42.2 (2.7)	16.1 (1.7)	40.0 (5.0)
NoCOV	87.0 (6.7)	26.0 (3.7)	11.2 (2.1)	14.9 (9.7)
NoSTRUC	60.2 (1.7)	79.6 (0.3)	61.9 (0.5)	44.6 (4.5)
HeadExpOnly	88.8 (6.4)	21.9 (8.6)	9.3 (3.5)	14.0 (7.9)

Table 2: Results on the complete corpus for the complete system (Full GenX), ablating the pruning model (NoPrune) and the different features: without coverage features (NoCOV), without structure features (NoSTRUC) and using only the logical expression HeadExp features (HeadExpOnly). Standard deviation over five runs is shown in parentheses.

8 Results

We report results on both the single-object subset of our data and the full dataset. Since our approach is online, and therefore sensitive to data ordering, we average results over 5 different runs with randomly ordered data, and report variance.

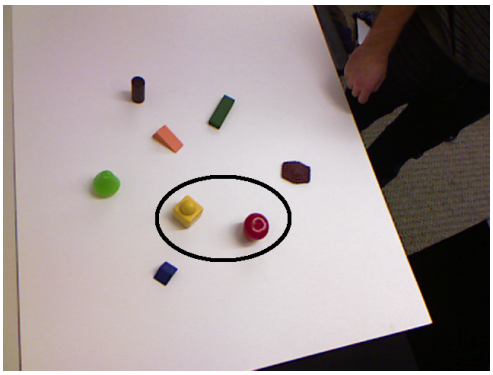
Single Objects Table 1 shows the different metrics for generating referring expression for single objects only. Our approach outperforms VOA (Mitchell et al., 2013) on all metrics, including an average of approximately 35% relative reduction in MAE. In addition, unlike VOA, our system (GenX) produces every logical expression used to refer to single objects in our dataset, including a small number which use negation and equality.

Object Sets Table 2 lists results on the full dataset. Our learned pruning approach produces an average 72.9% of the unique logical expressions used present in our dataset – over 87% when these counts are weighted by their frequency. The globally scored model achieves a mean absolute error of 54.3, and correctly assigns the highest probability to the most likely expression over 52% of the time.

Also shown in Table 2 are results obtained when elements of our approach are ablated. Using the global model for pruning instead of an explicitly trained model causes a large drop in performance, demonstrating that our global model is inappropri-

Q	$\hat{P}$	z
.750	.320	$\iota(\lambda x.object(x) \wedge (yellow(x) \vee red(x)))$
	.114	$\iota(\lambda x.lego(x)) \cup \iota(\lambda x.red(x) \wedge apple(x))$
	.114	$\iota(\lambda x.yellow(x) \wedge lego(x))) \cup \iota(\lambda x.apple(x))$
	.044	$\iota(\lambda x.lego(x) \vee (red(x) \wedge apple(x)))$
	.044	$\iota(\lambda x.(yellow(x) \wedge lego(x)) \vee apple(x))$
	.036	$\iota(\lambda x.lego(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
	.026	$\iota(\lambda x.red(x) \wedge lego(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
.050	.021	$\iota(\lambda x.(lego(x) \wedge yellow(x)) \vee (red(x) \wedge apple(x)))$
	.017	$\iota(\lambda x.(lego(x) \wedge yellow(x)) \vee (red(x) \wedge sphere(x)))$
	.014	$\iota(\lambda x.yellow(x) \wedge lego(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
.100	.010	$\iota(\lambda x.yellow(x) \wedge object(x)) \cup \iota(\lambda x.apple(x))$
.050	.007	$\iota(\lambda x.yellow(x) \wedge object(x)) \cup \iota(\lambda x.red(x) \wedge sphere(x))$
.050	.005	$\iota(\lambda x.yellow(x) \wedge object(x)) \cup \iota(\lambda x.red(x) \wedge object(x))$

(a)



(b)

Figure 4: Example output of our system for the scene on the right. We show the top 10 expressions (z) from the predicted distribution ($\hat{P}$) compared to the empirical distribution estimated from our labeled data (Q). The bottom section shows the predicted probability of the three expressions which were not in the top 10 of the predicted distribution. Although the mean absolute error (MAE) of $\hat{P}$ and Q is 63.8, $\hat{P}$ covers all of the entries in Q in the correct relative order and also fills in many other plausible candidates.

ate for pruning. We also ablate subsets of our features, demonstrating that the coverage and structural features are both crucial for performance.

Qualitatively, we found the learned distributions were often higher quality than the seemingly high mean absolute error would imply. Figure 4 shows an example output where the absolute error of the predicted distribution was 63.8. Much of the error can be attributed to probability mass assigned to logical expressions which, although not observed in our test data, are reasonable referring expressions. This might be due to the fact that our estimate of the empirical distribution comes from a fairly small sample (20), or other factors which we do not model that make these expressions less likely.

9 Conclusion

In this paper, we modeled REG as a density-estimation problem. We demonstrated that we can learn to produce distributions over logical referring expressions using a globally normalized model. Key to the approach was the use of a learned pruning model to define the space of logical expression that are explicitly enumerated during inference. Experiments demonstrate state-of-the-art performance on single object reference and the first results for learning to name sets of objects, correctly recovering over 87% of the observed logical forms.

This approach suggests several directions for future work. Lambda-calculus meaning representations can be designed for many semantic phenomena, such as spatial relations, superlatives, and graded properties, that are not common in our data. Collecting new datasets would allow us to study the extent to which the approach would scale to domains with such phenomena.

Although the focus of this paper is on REG, the approach is also applicable to learning distributions over logical meaning representations for many other tasks. Such learned models could provide a range of possible inputs for systems that map logical expressions to sentences (White and Rajkumar, 2009; Lu and Ng, 2011), and could also provide a valuable prior on the logical forms constructed by semantic parsers in grounded settings (Artzi and Zettlemoyer, 2013b; Matuszek et al., 2012a).

Acknowledgements

This research was supported in part by the Intel Science and Technology Center for Pervasive Computing, by DARPA under the DEFT program through the AFRL (FA8750-13-2-0019) and the CSSG (N11AP20020), the ARO (W911NF-12-1-0197), and the NSF (IIS-1115966). The authors wish to thank Margaret Mitchell, Mark Yatskar, Anthony Fader, Kenton Lee, Eunsol Choi, Gabriel Schubiner, Leila Zilles, Adrienne Wang, and the anonymous reviewers for their helpful comments.

References

Areces, C., Koller, A., and Striegnitz, K. (2008). Referring expressions as formulas of description logic. In *Proceedings of the International Natural Language Generation Conference*.

Artzi, Y. and Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. (2013a). UW SPF: The University of Washington Semantic Parsing Framework.

Artzi, Y. and Zettlemoyer, L. (2013b). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Barzilay, R. and Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Berg, A. C., Berg, T. L., Daume, H., Dodge, J., Goyal, A., Han, X., Mensch, A., Mitchell, M., Sood, A., Stratos, K., et al. (2012). Understanding and predicting importance in images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Carenini, G., Ng, R. T., and Pauls, A. (2006). Multi-document summarization of evaluative text. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

Carpenter, B. (1997). *Type-Logical Semantics*. The MIT Press.

Chen, D., Kim, J., and Mooney, R. (2010). Training a multilingual sportscaster: using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37(1):397–436.

Chen, D. and Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Dale, R. and Reiter, E. (1995). Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19:233–264.

Dale, R. and Reiter, E. (2000). *Building natural language generation systems*. Cambridge University Press.

Gardent, C. (2002). Generating minimal definite descriptions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Gatt, A. and van Deemter, K. (2007). Incremental generation of plural descriptions: Similarity and partitioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Gatt, A., Van Der Sluis, I., and Van Deemter, K. (2007). Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the European Workshop on Natural Language Generation*.

Grice, H. P. (1975). Logic and conversation. *1975*, pages 41–58.

Horacek, H. (2004). On referring to sets of objects naturally. In *Natural Language Generation*, pages 70–79. Springer.

Kim, J. and Mooney, R. J. (2012). Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Konstas, I. and Lapata, M. (2012). Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Krahmer, E. and van Deemter, K. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1(2):193–206.

Liang, P., Jordan, M., and Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Lu, W. and Ng, H. T. (2011). A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012a). A joint model of language and perception for grounded attribute learning. *Proceedings of the International Conference on Machine Learning*.

Matuszek, C., Herbst, E., Zettlemoyer, L. S., and Fox, D. (2012b). Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.

Mitchell, M., van Deemter, K., and Reiter, E. (2011a). Applying machine learning to the choice of size modifiers. In *Proceedings of the PRE-CogSci Workshop*.

Mitchell, M., Van Deemter, K., and Reiter, E. (2011b). Two approaches for generating size modifiers. In *Proceedings of the European Workshop on Natural Language Generation*.

Mitchell, M., van Deemter, K., and Reiter, E. (2013). Generating expressions that refer to visible objects. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*.

Ren, Y., Van Deemter, K., and Pan, J. Z. (2010). Charting the potential of description logic for the generation of referring expressions. In *Proceedings of the International Natural Language Generation Conference*.

Scontras, G., Graff, P., and Goodman, N. D. (2012). Comparing pluralities. *Cognition*, 123(1):190–197.

Steedman, M. (2011). *Taking Scope*. The MIT Press.

Stone, M. (2000). On identifying sets. In *Proceedings of the International Conference on Natural Language Generation*.

van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28:37–52.

van Deemter, K., Gatt, A., Sluis, I. v. d., and Power, R. (2012a). Generation of referring expressions: Assessing the incremental algorithm. *Cognitive Science*, 36(5):799–836.

van Deemter, K., Gatt, A., van Gompel, R. P., and Krahmer, E. (2012b). Toward a computational psycholinguistics of reference production. *Topics in Cognitive Science*, 4(2):166–183.

Viethen, J. and Dale, R. (2010). Speaker-dependent variation in content selection for referring expression generation. In *Proceedings of the Australasian Language Technology Workshop*.

Viethen, J., Mitchell, M., and Krahmer, E. (2013). Graphs and spatial relations in the generation of referring expressions. In *Proceedings of the European Workshop on Natural Language Generation*.

White, M. and Rajkumar, R. (2009). Perceptron reranking for ccg realization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zitnick, C. L. and Parikh, D. (2013). Bringing semantics into focus using visual abstraction. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Learning to rank lexical substitutions

György Szarvas[1]
Amazon Inc.

szarvasg@amazon.com

Róbert Busa-Fekete[2] **Eyke Hüllermeier**
University of Marburg
Hans-Meerwein-Str., 35032 Marburg, Germany
busarobi@mathematik.uni-marburg.de
eyke@mathematik.uni-marburg.de

Abstract

The problem to replace a word with a synonym that fits well in its sentential context is known as the lexical substitution task. In this paper, we tackle this task as a supervised ranking problem. Given a dataset of target words, their sentential contexts and the potential substitutions for the target words, the goal is to train a model that accurately ranks the candidate substitutions based on their contextual fitness. As a key contribution, we customize and evaluate several learning-to-rank models to the lexical substitution task, including classification-based and regression-based approaches. On two datasets widely used for lexical substitution, our best models significantly advance the state-of-the-art.

1 Introduction

The task to generate lexical substitutions in context (McCarthy and Navigli, 2007), i.e., to replace words in a sentence without changing its meaning, has become an increasingly popular research topic. This task is used, e.g. to evaluate semantic models with regard to their accuracy in modeling word meaning in context (Erk and Padó, 2010). Moreover, it provides a basis of NLP applications in many fields, including linguistic steganography (Topkara et al., 2006; Chang and Clark, 2010), semantic text similarity (Agirre et al., 2012) and plagiarism detection (Gipp et al., 2011). While closely related to WSD, lexical substitution does not rely on explicitly defined sense inventories (Dagan et al., 2006): the possible substitutions reflect all conceivable senses of the word, and the correct sense has to be ascertained to provide an accurate substitution.

While a few lexical sample datasets (McCarthy and Navigli, 2007; Biemann, 2012) with human-provided substitutions exist and can be used to evaluate different lexical paraphrasing approaches, a practically useful system must also be able to rephrase unseen words, i.e., any word for which a list of synonyms is provided. Correspondingly, unsupervised and knowledge-based approaches that are not directly dependent on any training material, prevailed in the SemEval 2007 shared task on English Lexical Substitution and dominated follow-up work. The only supervised approach is limited to the combination of several knowledge-based lexical substitution models based on different underlying lexicons (Sinha and Mihalcea, 2009).[3]

A recent work by Szarvas et al. (2013) describes a tailor-made supervised system based on delexicalized features that – unlike earlier supervised approaches, and similar to unsupervised and knowledge-based methods proposed for this task – is able to generalize to an open vocabulary. For each target word to paraphrase, they first compute a set of substitution candidates using WordNet: all synonyms from all of the target word's WordNet synsets, together with the words from synsets in *similar to, entailment* and *also see* relation to these synsets are considered as potential substitutions. Each candidate then constitutes a training (or test)

[1]Work was done while working at RGAI of the Hungarian Acad. Sci. and University of Szeged.

[2] R. Busa-Fekete is on leave from the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged.

[3]Another notable example for supervised lexical substitution is Biemann (2012), but this is a lexical sample system applicable only to the target words of the training datasets.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1926–1932,
Seattle, Washington, USA, 18-21 October 2013.

example, and these instances are characterized using non-lexical features from heterogeneous evidence such as lexical-semantic resources and distributional similarity, n-gram counts and shallow syntactic features computed on large, unannotated background corpora. The goal is then i) to predict how well a particular candidate fits in the original context, and ii) given these predictions for each of the candidates, to correctly order the elements of the candidate set according to their contextual fitness. That is, a model is successful if it prioritizes plausible substitutions ahead of less likely synonyms (given the context). This model is able to generate paraphrases for target words not contained in the training material. This favorable property is achieved using only such features (e.g. local n-gram frequencies in context) that are meaningfully comparable across the different target words and candidate substitutions they are computed from. More importantly, their model also provides superior ranking results compared to state of the art unsupervised and knowledge based approaches and therefore it defines the current state of the art for open vocabulary lexical substitution.

Motivated by the findings of Szarvas et al. (2013), we address lexical substitution as a supervised learning problem, and go beyond their approach from a methodological point of view. Our experiments show that the performance on the lexical substitution task is strongly influenced by the way in which this task is formalized as a machine learning problem (i.e., as binary or multi-class classification or regression) and by the learning method used to solve this problem. As a result, we are able to report the best performances on this task for two standard datasets.

2 Related work

Previous approaches to lexical substitution often seek to automatically generate a set of candidate substitutions for each target word first, and to rank the elements of this set of candidates afterward (Hassan et al., 2007; Giuliano et al., 2007; Martinez et al., 2007; Yuret, 2007). Alternatively, the candidate set can be defined by all human-suggested substitutions for the given target word in all of its contexts; then, the focus is just on the ranking problem (Erk and Padó, 2010; Thater et al., 2010; Dinu and Lapata, 2010; Thater et al., 2011). While only the former approach qualifies as a full-fledged substitution system for arbitrary, previously unseen target words, the latter simplifies the comparison of semantic ranking models, as the ranking step is not burdened with the shortcomings of automatically generated substitution candidates.

As mentioned before, Szarvas et al. (2013) recently formalized the lexical substitution problem as a supervised learning task, using delexicalized features. This non-lexical feature representation makes different target word/substitution pairs in different contexts[4] directly comparable. Thus, it becomes possible to learn an all-words system that is applicable to unseen words, using supervised methods, which provides superior ranking accuracy to unsupervised and knowledge based models.

In this work, we build on the problem formulation and the features proposed by Szarvas et al. (2013) while largely extending their machine learning methodology. We customize and experiment with several different learning-to-rank models, which are better tailored for this task. As our experiments show, this contribution leads to further significant improvements in modeling the semantics of a text and in end-system accuracy.

3 Datasets and experimental setup

Here we introduce the datasets, experimental setup and evaluation measures used in our experiments. Since space restrictions prohibit a comprehensive exposition, we only provide the most essential information and refer to Szarvas et al. (2013), whose experimental setup we adopted, for further details.

Datasets. We use two prominent datasets for lexical substitution. The *LexSub dataset* introduced in the Lexical Substitution task at Semeval 2007 (McCarthy and Navigli, 2007)[5] contains 2002 sentences for a total of 201 target words (from *all parts of speech*), and lexical substitutions assigned (to each target word and sentence pair) by 5 native speaker annotators. The second dataset, *TWSI* (Biemann, 2012)[6], consists of 24,647 sentences for a total of 1,012 target *nouns*, and lexical substitu-

[4] E.g., *bright* substituted with *intelligent* in *"He was bright and independent and proud"* and *side* for *part* in *"Find someone who can compose the biblical side"*.

[5] http://nlp.cs.swarthmore.edu/semeval/tasks/task10/data.shtml

[6] http://www.ukp.tu-darmstadt.de/data/lexical-resources/twsi-lexical-substitutions/

tions for each target word in context resulting from a crowdsourced annotation process.

For each sentence in each dataset, the annotators provided as many substitutions for the target word as they found appropriate in the context. Each substitution is then labeled by the number of annotators who listed that word as a good lexical substitution.

Experimental setup and Evaluation. On both datasets, we conduct experiments using a 10-fold cross validation process, and evaluate all learning algorithms on the same train/test splits. The datasets are randomly split into 10 equal-sized folds on the *target word* level, such that all examples for a particular target word fall into *either the training or the test set, but never both*. This way, we make sure to evaluate the models on target words *not seen during training*, thereby mimicking an open vocabulary paraphrasing system: at testing time, paraphrases are ranked for unseen target words, similarly as the models would rank paraphrases for any words (not necessarily contained in the dataset). For algorithms with tunable parameters, we further divide the training sets into a training and a validation part to find the best parameter settings. For evaluation, we use Generalized Average Precision (GAP) (Kishida, 2005) and Precision at 1 (P@1), i.e., the percentage of correct paraphrases at rank 1.

Features. In all experiments, we used the features described in Szarvas et al. (2013), implemented precisely as proposed by the original work.

Each $(sentence, target\,word, substitution)$ triplet represents an instance, and the feature values are computed from the sentence context, the target word and the substitution word. The features used fall into four major categories.

The most important features describe the syntagmatic coherence of the substitution in context, measured as local n-gram frequencies obtained from web data. The frequency for a 1-5gram context with the substitution word is computed and normalized with respect to either 1) the frequency of the original context (with the target word) or 2) the sum of frequencies observed for all possible substitutions. A third feature computes similar frequencies for the substitution *and* the target word observed in the local context (as part of a conjunctive phrase).

A second group of features describe the (non-positional, i.e. non-local) distributional similarity of the target and its candidate substitution in terms of sentence level co-occurrence statistics collected from newspaper texts: 1) How many words from the sentence appear in the top 1000 salient words listed for the candidate substitution in a distributional thesaurus, 2) how similar the top K salient words lists are for the candidate and the target word, 3) how similar the 2nd order distributional profiles are for candidate and target, etc. All these features are carefully normalized so that values compare well accross different words and contexts.

Another set of features capture the properties of the target and candidate word in WordNet, such as their 1) number of senses, 2) how frequent senses are synonymous and 3) the lowest common ancestor (and all synsets up) for the candidate and target word in the WordNet hierarchy (represented as a nominal feature, by the ID of these synsets).

Lastly a group of features capture shallow syntactic patterns of the target word and its local context in the form of 1) part of speech patterns (trigrams) in a sliding window around the target word using main POS categories, i.e. only the first letter of the Penn Treebank codes, and 2) the detailed POS code of the candidate word assigned by a POS tagger.

We omit a mathematically precise description of these features for space reasons and refer the reader to Szarvas et al. (2013) for a more formal and detailed description of the feature functions. Importantly, these delexicalized features are numerically comparable across the different target words and candidate substitutions they are computed from. This property enables the models to generalize over the words in the datasets and thus enables a supervised, all-words lexical substitution system.

4 Learning-to-Rank methods

Machine learning methods for ranking are traditionally classified into three categories. In the *pointwise* approach, a model is trained that maps instances (in this case candidate substitutions in a context) to scores indicating their relevance or fitness; to this end, one typically applies standard regression techniques, which essentially look at individual instances in isolation (i.e., independent of any other instances in the training or test set). To predict a ranking of a set of query instances, these are simply sorted by their predicted scores (Li et al., 2007).

The *pairwise* approach trains models that are able to compare pairs of instances. By marking such a pair as positive if the first instance is preferred to the second one, and as negative otherwise, the problem can formally be reduced to a binary classification task (Freund et al., 2003). Finally, in the *listwise* approach, tailor-made learning methods are used that directly optimize the ranking performance with respect to a global evaluation metric, i.e., a measure that evaluates the ranking of a complete set of query instances (Valizadegan et al., 2009).

Below we give a brief overview of the methods included in our experiments. We used the implementations provided by the MultiBoost (Benbouzid et al., 2012), RankSVM and RankLib packages.[7] For a detailed description, we refer to the original literature.

4.1 MAXENT

The ranking model proposed by Szarvas et al. (2013) was used as a baseline. This is a pointwise approach based on a maximum entropy classifier, in which the ranking task is cast as a binary classification problem, namely to discriminate good ($label > 0$) from bad substitutions. The actual label values for good substitutions were used for weighting the training examples. The underlying MaxEnt model was trained until convergence, i.e., there was no hyperparameter to be tuned. For a new target/substitution pair, the classifier delivers an estimation of the posterior probability for being a good substitution. The ranking is then produced by sorting the candidates in decreasing order according to this probability.

4.2 EXPENS

EXPENS (Busa-Fekete et al., 2013) is a pointwise method with listwise meta-learning step that exploits an ensemble of multi-class classifiers. It consists of three steps. First, ADABOOST.MH (Schapire and Singer, 1999) classifiers with several different weak learners (Busa-Fekete et al., 2011; Kégl and Busa-Fekete, 2009) are trained to predict the level of relevance (quality) of a substitution (i.e., the number of annotators who proposed the candidate for that particular context). Second, the classifiers are calibrated to obtain an accurate posterior distribution; to this end, several calibration techniques, such as Platt scaling (Platt, 2000), are used to obtain a diverse pool of calibrated classifiers. Note that this step takes advantage of the ordinal structure of the underlying scale of relevance levels, which is an important difference to MAXENT. Third, the posteriors of these calibrated classifiers are additively combined, with the weight of each model being exponentially proportional to its GAP score (on the validation set). This method has two hyperparameters: the number of boosting iterations T and the scaling factor in the exponential weighting scheme c. We select T and c from the intervals $[100, 2000]$ and $[0, 100]$, with step sizes 100 and 10, respectively.

4.3 RANKBOOST

RANKBOOST (Freund et al., 2003) is a pairwise boosting approach. The objective function is the rank loss (as opposed to ADABOOST, which optimizes the exponential loss). In each boosting iteration, the weak classifier is chosen by maximizing the weighted rank loss. For the weak learner, we used the decision stump described in (Freund et al., 2003), which is able to optimize the rank loss in an efficient way. The only hyperparameter of RANKBOOST to be tuned is the number of iterations that we selected from the interval $[1, 1000]$.

4.4 RANKSVM

RANKSVM (Joachims, 2006) is a pairwise method based on support vector machines, which formulates the ranking task as binary classification of pairs of instances. We used a linear kernel, because the optimization using non-linear kernels cannot be done in a reasonable time. The tolerance level of the optimization was set to 0.001 and the regularization parameter was validated in the interval $[10^{-6}, 10^{4}]$ with a logarithmically increasing step size.

4.5 LAMBDAMART

LAMBDAMART (Wu et al., 2010) is a listwise method based on the gradient boosted regression trees by Friedman (1999). The ordinal labels are learned directly by the boosted regression trees whose parameters are tuned by using a gradient-based optimization method. The gradient of parameters is calculated based on the evaluation metric used (in this case GAP). We tuned the number of boosting

[7] RankLib is available at `http://people.cs.umass.edu/~vdang/ranklib.html`. We extended the implementation of the LAMBDAMART algorithm in this package to compute the gradients of and optimize for the GAP measure.

Database	LexSub		TWSI	
Candidates	WN	Gold	WN	Gold
	GAP			
MaxEnt	43.8	52.4	36.6	47.2
ExpEns	44.3	**53.5**	**37.8**	**49.7**
RankBoost	44.0	51.4	37.0	47.8
RankSVM	43.3	51.8	35.5	45.2
LambdaMART	**45.5**	**55.0**	**37.8**	**50.1**
	P@1			
MaxEnt	40.2	57.7	32.4	49.5
ExpEns	39.8	**58.5**	**33.8**	**53.2**
RankBoost	**40.7**	55.2	33.1	50.8
RankSVM	40.3	51.7	33.2	45.1
LambdaMART	**40.8**	**60.2**	**33.1**	**53.6**

Table 1: GAP and p@1 values, with significant improvements over the performance of MaxEnt marked in bold.

System	GAP
Erk and Padó (2010)	38.6
Dinu and Lapata (2010)	42.9
Thater et al. (2010)	46.0
Thater et al. (2011)	51.7
Szarvas et al. (2013)	52.4
EXPENS	53.5
LAMBDAMART	55.0

Table 2: Comparison to previous studies (dataset *LexSub*, candidates *Gold*).

iterations in the interval $[10, 1000]$ and the number of tree leaves in $\{8, 16, 32\}$.

5 Results and discussion

Our results using the above learning methods are summarized in Table 1. As can be seen, the two methods that exploit the cardinal structure of the label set (relevance degrees), namely EXPENS and LAMBDAMART, consistently outperform the baseline taken from Szarvas et al. (2013) – the only exception is the p@1 score for EXPENS on the Semeval Lexical Substitution dataset and the candidate substitutions extracted from WordNet. The improvements are significant (using paired t-test, $p < 0.01$) for 3 out of 4 settings for EXPENS and in all settings for LAMBDAMART. In particular, the results of LAMBDAMART are so far the best scores that have been reported for the best studied setting, i.e. the LexSub dataset using substitution candidates taken from the gold standard (see Table 2).

We suppose that the relatively good results achieved by the LAMBDAMART and EXPENS methods are due to that, first, it seems crucial to properly model and exploit the ordinal nature of the annotations (number of annotators who suggested a given word as a good paraphrase) provided by the datasets. Second, the RANKBOOST and RANKSVM are less complex methods than the EXPENS and LAMBDAMART. The RANKSVM is the least complex method from the pool of learning-to-rank methods we applied, since it is a simple linear model. The RANKBOOST is a boosted decision *stump* where, in each boosting iteration, the stump is found by maximizing the weighted exponential rank loss. On the other hand, both the EXPENS and LAMBDAMART make use of *tree* learners in the ensemble classifier they produce. We believe that overfitting is not an issue in a learning task like the LexSub task: most features are relatively weak predictors on their own, and we can learn from a large number of data points (2000 sentences with an average set size of 20, about 40K data points for the smallest dataset and setting). Rather, as our results show, less complex models tend to underfit the data. Therefore we believe that more complex models can achieve a better performance, of course with an increased computational cost.

6 Conclusion and future work

In this paper, we customized and applied some relatively novel algorithms from the field of learning-to-rank for ranking lexical substitutions in context. In turn, we achieved significant improvements on the two prominent datasets for lexical substitution.

Our results indicate that an exploitation of the ordinal structure of the labels in the datasets can lead to considerable gains in terms of both ranking quality (GAP) and precision at 1 (p@1). This observation is supported both for the theoretically simpler pointwise learning approach and for the most powerful listwise approach. On the other hand, the pairwise methods that cannot naturally exploit this property, did not provide a consistent improvement over the baseline. In the future, we plan to investigate this finding in the context of other, similar ranking problems in Natural Language Processing.

Acknowledgment

This work was supported by the German Research Foundation (DFG) as part of the Priority Programme 1527.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada.

D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl. 2012. MultiBoost: a multi-purpose boosting package. *Journal of Machine Learning Research*, 13:549–553.

Chris Biemann. 2012. Creating a System for Lexical Substitutions from Scratch using Crowdsourcing. *Language Resources and Evaluation: Special Issue on Collaboratively Constructed Language Resources*, 46(2).

R. Busa-Fekete, B. Kégl, T. Éltető, and Gy. Szarvas. 2011. Ranking by calibrated AdaBoost. In *(JMLR W&CP)*, volume 14, pages 37–48.

R. Busa-Fekete, B. Kégl, T. Éltető, and Gy. Szarvas. 2013. Tune and mix: learning to rank using ensembles of calibrated multi-class classifiers. *Machine Learning*, 93(2–3):261–292.

Ching-Yun Chang and Stephen Clark. 2010. Practical linguistic steganography using contextual synonym substitution and vertex colour coding. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1194–1203, Cambridge, MA.

Ido Dagan, Oren Glickman, Alfio Gliozzo, Efrat Marmorshtein, and Carlo Strapparava. 2006. Direct word sense matching for lexical substitution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 449–456, Sydney, Australia.

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 92–97, Uppsala, Sweden.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

J. Friedman. 1999. Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University.

Bela Gipp, Norman Meuschke, and Joeran Beel. 2011. Comparative Evaluation of Text- and Citation-based Plagiarism Detection Approaches using GuttenPlag. In *Proceedings of 11th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'11)*, pages 255–258, Ottawa, Canada. ACM New York, NY, USA. Available at http://sciplore.org/pub/.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. 2007. FBK-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 145–148, Prague, Czech Republic.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. UNT: SubFinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 410–413, Prague, Czech Republic.

T. Joachims. 2006. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

B. Kégl and R. Busa-Fekete. 2009. Boosting products of base classifiers. In *International Conference on Machine Learning*, volume 26, pages 497–504, Montreal, Canada.

Kazuaki Kishida. 2005. *Property of Average Precision and Its Generalization: An Examination of Evaluation Indicator for Information Retrieval Experiments*. NII technical report. National Institute of Informatics.

P. Li, C. Burges, and Q. Wu. 2007. McRank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems*, volume 19, pages 897–904. The MIT Press.

David Martinez, Su Nam Kim, and Timothy Baldwin. 2007. MELB-MKB: Lexical substitution system based on relatives in context. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 237–240, Prague, Czech Republic.

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

J. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Ravi Sinha and Rada Mihalcea. 2009. Combining lexical resources for contextual synonym expansion. In *Proceedings of the International Conference RANLP-2009*, pages 404–410, Borovets, Bulgaria.

György Szarvas, Chris Biemann, and Iryna Gurevych. 2013. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, June.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957, Uppsala, Sweden.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing : IJCNLP 2011*, pages 1134–1143, Chiang Mai, Thailand. MP, ISSN 978-974-466-564-5.

Umut Topkara, Mercan Topkara, and Mikhail J. Atallah. 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, pages 164–174, New York, NY, USA. ACM.

H. Valizadegan, R. Jin, R. Zhang, and J. Mao. 2009. Learning to rank by optimizing NDCG measure. In *Advances in Neural Information Processing Systems 22*, pages 1883–1891.

Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao. 2010. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270.

Deniz Yuret. 2007. Ku: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 207–214, Prague, Czech Republic, June. Association for Computational Linguistics.

Identifying Manipulated Offerings on Review Portals

Jiwei Li
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jiweil@cs.cmu.edu

Myle Ott **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
myleott,cardie@cs.cornell.edu

Abstract

Recent work has developed supervised methods for detecting *deceptive opinion spam*—fake reviews written to sound authentic and deliberately mislead readers. And whereas past work has focused on identifying individual fake reviews, this paper aims to identify *offerings* (e.g., hotels) that contain fake reviews. We introduce a semi-supervised manifold ranking algorithm for this task, which relies on a small set of labeled individual reviews for training. Then, in the absence of gold standard labels (at an offering level), we introduce a novel evaluation procedure that ranks artificial instances of real offerings, where each artificial offering contains a known number of injected deceptive reviews. Experiments on a novel dataset of hotel reviews show that the proposed method outperforms state-of-art learning baselines.

1 Introduction

Consumers increasingly rely on user-generated online reviews when making purchase decisions (Cone, 2011; Ipsos, 2012). Unfortunately, the ease of posting content to the Web, potentially anonymously, combined with the public's trust and growing reliance on opinions and other information found online, create opportunities and incentives for unscrupulous businesses to post *deceptive opinion spam*—fraudulent or fictitious reviews that are deliberately written to sound authentic, in order to deceive the reader (Ott et al, 2011).

Unlike other kinds of spam, such as Web (Martinez-Romo and Araujo, 2009; Castillo et al, 2006) and e-mail spam (Chirita et al, 2005), recent work has found that deceptive opinion spam is neither easily ignored nor easily identified by human readers (Ott et al, 2011). Accordingly, there is growing interest in developing automatic, usually learning-based, methods to help users identify deceptive opinion spam (see Section 2). Even in fully-supervised settings, however, automatic methods are imperfect at identifying individual deceptive reviews, and erroneously labeling genuine reviews as deceptive may frustrate and alienate honest reviewers.

An alternative approach, not yet considered in previous work, is to instead identify those product or service offerings where fake reviews appear with high probability. For example, a hotel manager may post fake positive reviews to promote their own hotel, or fake negative reviews to demote a competitor's hotel. In both cases, rather than identifying these deceptive reviews individually, it may be preferable to identify the *manipulated offering* (i.e., the hotel) so that review portal operators, such as TripAdvisor or Yelp, can further investigate the situation without alienating users.[1]

Accordingly, this paper addresses the novel task of *identifying manipulated offerings*, which we frame as a ranking problem, where the goal is to rank offerings by the proportion of their reviews that are believed to be deceptive. We propose a novel three-layer graph model, based on manifold ranking (Zhou et al, 2003a; 2003b), to jointly model deceptive language at the offering-, review- and term-level. In particular, rather than treating reviews within the same offering as independent units, there is a reinforcing relationship between offerings and reviews.

[1] Manipulating online reviews may also have legal consequences. For example, the Federal Trade Commission (FTC) has updated their guidelines on the use of endorsements and testimonials in advertising to suggest that posting deceptive reviews may be unlawful in the United States (FTC, 2009).

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1933–1942,
Seattle, Washington, USA, 18-21 October 2013.

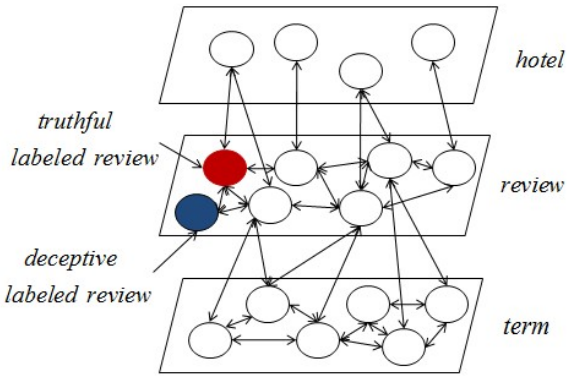


Figure 1: Mutual Reinforcement Graph Model for Hotel Ranking using the Manifold-Ranking Method

Our manifold ranking approach is semi-supervised in that it *requires no supervisory information at the offering level*; rather, it requires only a small amount of labeled data at a review level. Intuitively, and as depicted in Figure 1 for hotel offerings, we represent hotels, reviews and terms as nodes in a graph, where each hotel is connected to its reviews, and each review, in turn, is connected to the terms used within it. The influence of labeled data is propagated along the graph to unlabeled data, such that a hotel is considered more deceptive if it is heavily linked with other deceptive reviews, and a review, in turn, is more deceptive if it is generated by a deceptive hotel.

The success of our semi-supervised approach further depends on the ability to learn patterns of truthful and deceptive reviews that generalize across reviews of different offerings. This is challenging, because reviews often contain offering-specific vocabulary. For example, reviews of hotels in Los Angeles are more likely to include keywords such as "beach", "sea", "sunshine" or "LA", while reviews of Juneau hotels may contain "glacier", "Juneau", "bear" or "aurora borealis." A hotel review might also mention the hotel's restaurant or bar by name.

Unfortunately, it is unclear how important (or detrimental) offering-specific features are when deciding whether a review is fake. Accordingly, we *propose a dimensionality-reduction approach*, based on Latent Dirichlet Allocation (LDA) (Blei et al, 2003), *to obtain a vector representation of reviews for the ranking algorithm that generalizes across reviews of different offerings*. Specifically, we train an LDA-based topic model to view each review as a mixture of aspect-, city-, hotel- and review-specific topics (see Section 6). We then reduce the dimensionality of our data (i.e., labeled and unlabeled reviews) by replacing each review term vector with a vector that corresponds to its term distribution over just its aspect-specific topics, i.e., excluding city-, hotel- and review-specific topics. We find that, compared to models trained either on the full vocabulary, or trained on standard LDA document-topic vectors, this representation allows our models to generalize better across reviews of different offerings.

We evaluate our approach on the task of identifying (ranking) manipulated hotels. In particular, in the absence of gold standard offering-level labels, we introduce a *novel evaluation procedure for this task*, in which we rank numerous *versions* of each hotel, where each hotel version contains a different number of injected, known deceptive reviews. Thus, we expect hotel versions with larger proportions of deceptive reviews to be ranked higher than those with smaller proportions.

For labeled training data, we use the Ott et al. (2011) dataset of 800 *positive* (5-star) reviews of 20 Chicago hotels (400 deceptive and 400 truthful). For evaluation, *we construct a new* FOUR-CITIES *dataset*, containing 40 deceptive and 40 truthful reviews for each of eight hotels in four different cities (640 reviews total), following the procedure outlined in Ott et al. (2011). We find that our manifold ranking approach outperforms several state-of-the-art learning baselines on this task, including transductive Support Vector Regression. We additionally apply our approach to a large-scale collection of real-world reviews from TripAdvisor and explore the resulting ranking.

In the sections below, we discuss related work (Section 2) and describe the datasets used in this work (Section 3), the dimensionality-reduction approach for representing reviews (Section 4), and the semi-supervised manifold ranking approach (Section 5). We then evaluate the methods *quantitatively* (Sections 6 and 7) and *qualitatively* (Section 8).

2 Related Work

A number of recent approaches have focused on identifying individual fake reviews or users who post

fake reviews. For example, Jindal and Liu (2008) train machine learning classifiers to identify duplicate (or near duplicate) reviews. Yoo and Gretzel (2009) gathered 40 truthful and 42 deceptive hotel reviews and manually compare the psychologically relevant linguistic differences between them. Lim et al. (2010) propose an approach based on abnormal user behavior to predict spam users, without using any textual features. Ott et al. (2011) solicit deceptive reviews from workers on Amazon Mechanical Turk, and built a dataset containing 400 deceptive and 400 truthful reviews, which they use to train and evaluate supervised SVM classifiers. Ott et al. (2012) expand upon this work to estimate prevalences of deception in a review community. Mukherjee et al. (2012) study spam produced by groups of fake reviewers. Li et al. (2013) use topic models to detect differences between deceptive and truthful topic-word distributions. In contrast, in this work we aim to identify fake reviews at an offering level.[2]

LDA Topic Models. LDA topic models (Blei et al, 2003) have been employed for many NLP tasks in recent years. Here, we build on earlier work that uses topic models to (a) separate background information from information discussing the various "aspects" of products (e.g., Chemudugunta et al. (2007)) and (b) identify different levels of information (e.g., user-specific, location-specific, time-specific) (Ramage et al., 2009).

Manifold Ranking Algorithm. The manifold-ranking method (Zhou et al, 2003a; Zhou et al, 2003b) is a mutual reinforcement ranking approach initially proposed to rank data points along their underlying manifold structure. It has been widely used in many different ranking applications, such as summarization (Wan et al, 2007; Wan and Yang, 2007).

3 Dataset

In this paper, we train all of our models using the CHICAGO dataset of Ott et al (2011), which contains 20 deceptive and 20 truthful reviews from each of 20 Chicago hotels (800 reviews total). This dataset is unique in that it contains known (*gold standard*) deceptive reviews, solicited through Amazon Mechanical Turk, and is publicly-available.[3]

City	Hotels
Chicago	W Chicago, Palomar Chicago
New York	Hotel Pennsylvania, Waldorf Astoria
Los Angeles	Sheraton Gateway, The Westin Los Angeles Airport
Houston	Magnolia Hotel, Crowne Plaza Houston River Oaks

Table 1: Details of our FOUR-CITIES evaluation data.

Unfortunately, the CHICAGO dataset is limited, both in size (800 reviews) and scope, in that it only contains reviews of hotels in one city: Chicago. Accordingly, in order to perform a more realistic evaluation for our task, we construct a new dataset, FOUR-CITIES, that contains 40 deceptive and 40 truthful reviews from each of eight hotels in four different cities (640 reviews total).

We build the FOUR-CITIES dataset using the same procedure as Ott et al (2011), by creating and dividing 320 Mechanical Turk jobs, called *Human-Intelligence Tasks* (HITs), evenly across eight of the most popular hotels in our four chosen cities (see Table 1). Each HIT presents a worker with the name of a hotel and a link to the hotel's website. Workers are asked to imagine that they work for the marketing department of the hotel and that their boss has asked them to write a fake positive review, as if they were a customer, to be posted on a travel review website. Each worker is allowed to submit a single review, and is paid $1 for an acceptable submission.

Finally, we augment our deceptive FOUR-CITIES reviews with a matching set of truthful reviews from TripAdvisor by randomly sampling 40 positive (5-star) reviews for each of the eight chosen hotels. While we cannot know for sure that the sampled reviews are truthful, previous work has suggested that rates of deception among popular hotels is likely to be low (Jindal and Liu, 2008; Lim et al, 2010).

4 Topic Models for Dimensionality Reduction

As mentioned in the introduction, we want to learn patterns of truthful and deceptive reviews that apply

[2] Approaches for identifying individual fake reviews may be applied to our task, for example, by averaging the review-level predictions for an offering. This averaging approach is one of our baselines in Section 7.

[3] We use the dataset available at: `http://www.cs.cornell.edu/~myleott/op_spam`.

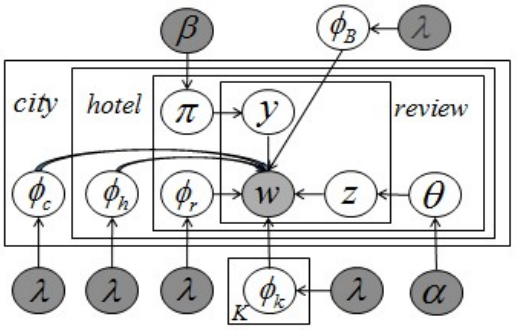


Figure 2: Graphical illustration of the RLDA topic model.

across hotels in different locations. This is challenging, however, because hotel reviews often contain specific information about the hotel or city, and it is unclear whether these features will generalize to reviews of other hotels.

We therefore investigate an LDA-based dimensionality-reduction approach (RLDA) to derive effective vector representations of reviews. Specifically, we model each document as a bag of words, generated from a mixture of: (a) "aspect" topics (that discuss various dimensions of the offering); (b) city-specific topics; (c) hotel-specific topics; (d) review-specific topics;[4] and (e) a background topic. We use this model to reduce the dimensionality of the review representation in our training and test sets, by replacing each review's term vector with a vector corresponding to the distribution over only the aspect-based topics, i.e., we exclude city, hotel and review-specific topics, as well as the background topic.

Below we present specific details of our model (Sections 4.1 and 4.2). The effectiveness of our dimensionality-reduction approach will be directly evaluated in Section 6, by comparing the performance of various classifiers trained either on the full vocabulary, or on our reduced feature representation.

4.1 RLDA Model Details

The plate diagram and generative story for our model are given in Figures 2 and 3, respectively. Our model has a similar general structure to standard LDA, but with additional machinery to handle different levels of information. In particular, in order to model K aspects in a collection of R reviews,

[4]These will be terms used in just a small number of reviews.

- Draw $\phi_B \sim Dir(\lambda)$
- For each aspect $z = 1, 2, ..., K$: draw $\phi^z \sim Dir(\lambda)$
- For each city $c = 1, 2, ..., C$: draw $\phi^c \sim Dir(\lambda)$
- For each hotel $h = 1, 2, ..., H$: draw $\phi^h \sim Dir(\lambda)$
- For each review r:
  - Draw $\pi^r \sim Dir(\beta)$
  - Draw $\phi^r \sim Dir(\lambda)$
  - Draw $\theta^r \sim Dir(\alpha)$
  - For each word w in d:
    - Draw $y_w \sim Multi(\pi_r)$
    - If $y_w = 0$:
      - Draw $z_w \sim Multi(\theta)$
      - Draw $w \sim Multi(\phi^{z_w})$
    - If $y_w = 1$: draw $w \sim Multi(\phi_B)$
    - If $y_w = 2$: draw $w \sim Multi(\phi_d)$
    - If $y_w = 3$: draw $w \sim Multi(\phi_h)$
    - If $y_w = 4$: draw $w \sim Multi(\phi_c)$

Figure 3: Generative story for the RLDA topic model.

of H hotels, in C cities, we first draw multinomial word distributions corresponding to: the background topic, ϕ_B; aspect topics, ϕ_k for $k \in [1, K]$; review-specific topics, ϕ_r for $r \in [1, R]$; hotel-specific topics, ϕ_h for $h \in [1, H]$; and city-specific topics, ϕ_c for $c \in [1, C]$. Then, for each word w in review R, we sample a switch variable, $y \in [0, 4]$, indicating whether w comes from one of the aspect topics ($y = 0$), or the background topic ($y = 1$), review-specific topic ($y = 2$), hotel-specific topic ($y = 3$) or city-specific topic ($y = 4$). If the word comes from one of the aspect topics, then we further sample the specific aspect topic, $z_w \in [1, K]$. Finally, we generate the word, w, from the corresponding ϕ.

4.2 Inference for RLDA

Given the review collection, our goal is to find the most likely assignment y_w (and z_w if $y_w = 0$) for each word, w, in each review. We perform inference using Gibbs sampling. It is relatively straightforward to derive Gibbs sampling equations that allow joint sampling of the z_w and y_w latent variables for each word token w:

$$P(y_w = 0, Z_w = k) = \frac{N^a_{r,-w} + \beta}{N_{r,-w} + 5\beta} \times \frac{C^k_{r,-w} + \alpha}{\sum_k C^k_{r,-w} + K\alpha} \times \frac{E^w_k + \lambda}{\sum_w E^w_k + V\lambda},$$

$$P(y_w = m, m = 1, 2, 3, 4) = \frac{N^m_{r,-w} + \beta}{N_{r,-w} + 5\beta} \times \frac{E^w_m + \lambda}{\sum_w E^w_m + V\lambda},$$

Note that the subscript $-w$ indicates that the count for word token w is excluded. Also, N_r

denotes the number of words in review r and $N^a_{r,-w}, N^1_{r,-w}, N^2_{r,-w}, N^3_{r,-w}, N^4_{r,-w}$ are the number of words in review r assigned to the aspect, background, review-specific, hotel-specific and city-specific topics, respectively, excluding the current word. $C^k_{r,-w}$ denotes the number of words in review r assigned to aspect topic k. $E^w_k, E^w_1, E^w_2, E^w_3, E^w_4$ denote the number of times that the word w is assigned to aspect k, the background topic, review-specific topic r, hotel-specific topic h, and city-specific topic c, respectively. We set hyperparameter α to 1, β to 0.5, λ to 0.01. We run 200 iterations of Gibbs sampling until the topic distribution stabilizes. After each iteration in Gibbs sampling, we obtain:

$$\pi^i_r = \frac{N^i_r + \beta}{\sum_i N^i_r + 5\beta} \quad \theta^k_r = \frac{C^k_r + \alpha}{\sum_k C^k_r + K\alpha}$$
$$\phi^w_z = \frac{E^w_z + \lambda}{\sum_w E^w_z + V\lambda} \quad \phi^w_m = \frac{E^w_m + \lambda}{\sum_w E^w_m + V\lambda} \tag{1}$$

Finally, at the end of Gibbs sampling, we filter out background, document-specific, hotel-specific and city-specific information, by replacing each document's term vector with a $1 \times K$ aspect-topic vector, $\vec{G_r} = \langle \theta^1_r, \theta^2_r, \cdots, \theta^K_r \rangle$.

5 Manifold Ranking for Hotels

In this section, we describe our ranking algorithm — based on manifold ranking (Zhou et al, 2003a; Zhou et al, 2003b) — that tries to jointly model deceptive language at the hotel-, review- and term-level.

5.1 Graph Construction

We use a three-layer (hotel layer, review layer and term layer) mutual reinforcement model (see Figure 1). Formally, we represent our three-layer graph as $G = \langle V_H, V_R, V_T, E_{HR}, E_{RR}, E_{RT}, E_{TT} \rangle$, where $V_H = \{H_i\}_{i=1}^{i=N_H}$, $V_R = \{R\}_{i=1}^{i=H_R}$ and $V_T = \{T_i\}_{i=1}^{i=V}$ correspond to the set of hotels, reviews and terms respectively. E_{HR}, E_{RR} and E_{RT} respectively denote the edges between hotels and reviews, reviews and reviews and reviews and terms. Each edge is associated with a weight that denotes the similarity between two nodes.

Let $sim(H_i, R_j)$, where $H_i \in V_H$ and $R_j \in V_R$, denote the edge weight between hotel H_i and review R_j, calculated as follows:

$$sim(H_i, R_j) = \begin{cases} 1 & if \quad R_i \in H_j \\ 0 & if \quad R_i \notin H_j \end{cases} \tag{2}$$

Then we get row normalized matrices $D_{HR} \in \mathbb{R}^{N_H \times N_R}$ and $D_{RH} \in \mathbb{R}^{N_R \times N_H}$ as follows:

$$D_{HR}(i,j) = \frac{sim(H_i, R_j)}{\sum_{i'} sim(H_{i'}, R_j)}$$
$$D_{RH}(i,j) = \frac{sim(H_i, R_j)}{\sum_{j'} sim(H_i, R_{j'})} \tag{3}$$

As described in Section 4.2, each review is represented with a $1 \times K$ aspect vector G_r after filtering undesired information. The edge weight between two reviews is then the cosine similarity, $sim(R_i, R_j)$, between two reviews and can be calculated as follows:

$$sim(R_i, R_j) = \frac{\sum_{t=1}^{t=K} G^t_i \cdot G^t_j}{\sqrt{\sum_{t=1}^{t=K} G^{t2}_i} \cdot \sqrt{\sum_{t=1}^{t=K} G^{t2}_j}} \tag{4}$$

Since the normalization process will make the review-to-review relation matrix asymmetric, we adopt the following strategy: let P denote the similarity matrix between reviews, where $P(i,j) = sim(R_i, R_j)$ and M denotes the diagonal matrix with (i,i)-element equal to the sum of the i^{th} row of SIM. The normalized matrix between reviews $D_{RR} \in \mathbb{R}^{N_R \times N_R}$ is calculated as follows:

$$D_{RR} = M^{-\frac{1}{2}} \cdot P \cdot M^{-\frac{1}{2}} \tag{5}$$

$sim(R_i, w_j)$ denotes the similarity between review R_i and term w_j and is the conditional probability of word w_j given review R_i. If $w_j \in R_j$, $sim(R_i, w_j)$ is calculated according to Eq. (6) by integrating out latent parameters θ and π. Else if $w_j \notin R_j$, $sim(R_i, w_j) = 0$.

$$\begin{aligned} sim(R_i, w_j) &= \sum_{k=1}^{k=K} p(z=k|r_i) \times p(w_j|z=k) \\ &\quad + \sum_{t \in \{B,h,c,d\}} p(w_j|y_i=t) p(y_i=t|r_i) \\ &= \pi^{(a)}_d \sum_{k=1}^{k=K} \theta^z_d \cdot \phi^{(w_j)}_z + \sum_{t \in \{B,h,c,d\}} \pi^{(t)}_d \phi^{(w_j)}_t \end{aligned} \tag{6}$$

Similar to Eq. (3), we further get the normalized matrix $D_{RT} \in \mathbb{R}^{H_R \times V}$ and $D_{TR} \in \mathbb{R}^{V \times H_R}$.

Similarity between terms $sim(w_i, w_j)$ is given by the WordNet path-similarity,[5] normalized to create the matrix D_{VV}.

[5]Path-similarity is based on the shortest path that connects the senses in the "is-a" (hypernym/hyponym) taxonomy. See `http://nltk.googlecode.com/svn/trunk/doc/howto/wordnet.html`.

Input: The hotel set V_D, review set V_R, term set V_T, normalized transition probability matrix $D_{HR}, D_{RR}, D_{RH}, D_{RT}, D_{TT}, D_{TR}$.
Output: the ranking vectors S_R, S_H, S_T.
Begin:

1. Initialization: set the score labeled reviews to $+1$ or -1 and other unlabeled reviews 0: $S_R^0 = [+1, ..., +1, -1, ..., -1, 0, ..., 0]$. Set S_H^0 and S_T^0 to 0. Normalize the score vector.
2. update S_R^k, S_H^k and S_T^k according to Eq. (7).
3. normalize S_R^k, S_H^k and S_T^k.
4. fix the score of labeled reviews to $+1$ and -1. Go to step (2) until convergence.

Figure 4: Semi-Supervised Reinforcement Ranking.

5.2 Reinforcement Ranking Based on the Manifold Method

Based on the set of labeled reviews, nodes for truthful reviews (positive) are initialized with a high score (1) and nodes for deceptive reviews, a low score (-1). Given the weighted graph, our task is to assign a score to the each hotel, each term, and the remaining unlabeled reviews. Let S_H, S_R and S_T denote the ranking scores of hotels, reviews and terms, which are updated during each iteration as follows until convergence[6]:

$$\begin{cases} S_H^{k+1} = D_{HR} \cdot S_R^k \\ S_R^{k+1} = \epsilon_1 D_{RR} \cdot S_R^k + \epsilon_2 D_{RH} \cdot S_H^k + \epsilon_3 D_{RT} \cdot S_t^k \\ S_T^{k+1} = \epsilon_4 D_{TT} \cdot S_T^k + \epsilon_5 D_{TR} \cdot S_R^k \end{cases} \tag{7}$$

where $\epsilon_1 + \epsilon_2 + \epsilon_3 = 1$ and $\epsilon_4 + \epsilon_5 = 1$. (The score of labeled reviews will be fixed to $+1$ or -1.)

6 Learning Generalizable Classifiers

In Section 4, we introduced RLDA to filter out review-, hotel- and city-specific information from our vector-based review representation. Here, we will directly evaluate the effectiveness of RLDA by comparing the performance of binary deceptive vs. truthful classifiers trained on three feature sets: (a) the full vocabulary, encoded as unigrams and bigrams (N-GRAMS); (b) a reduced-dimensionality feature space, based on standard LDA (Blei et al, 2003); and (c) a reduced-dimensionality feature space, based on our proposed revised LDA approach (RLDA).

We compare two kinds of classifiers, which are trained on only the labeled CHICAGO dataset and tested on the FOUR-CITIES dataset. First, we use SVM^{light} (Joachims, 1999) to train linear SVM classifiers, which have been shown to perform well in related work (Ott et al, 2011). Second, we train a two-layer manifold classifier, which is a simplified version of the model presented in Section 5. In this model, the graph consists of only review and term layers, and the score of a labeled review is fixed to 1 or -1 in each iteration. After convergence, reviews with scores greater than 0 are classified as truthful, and less than 0 as deceptive.

Results and Discussion The results are shown in Table 2 and show the average accuracy and precision/recall w.r.t. the *truthful* (positive) class. We find that SVM and MANIFOLD are comparable in all six conditions, and not surprisingly, perform best when evaluated on reviews from the two Chicago hotels in our FOUR-CITIES data. However, the N-GRAM and LDA feature sets perform much worse than RDLA when evaluation is performed on reviews from the other three (non-Chicago) cities. This confirms that classifiers trained on n-gram features overfit to the training data (CHICAGO) and do not generalize well to reviews from other cities. In addition, the standard LDA-based method for dimensionality reduction is not sufficient for our specific task.

7 Identifying Manipulated Hotels

In this section, we evaluate the performance of our manifold ranking approach (see Section 5) on the task of identifying *manipulated hotels*.

Baselines. We consider several baseline ranking approaches to compare to our manifold ranking approach. Like the manifold ranking approach, the baselines also employ both the CHICAGO dataset (labeled) and FOUR-CITIES dataset (**without** labels).[7] For fair comparison, we use identical processing techniques for each approach. Topic number is set

[6] Convergence is achieved if the difference between ranking scores in two consecutive iterations is less than 0.00001.

[7] While we have not investigated the effects of unlabeled data in detail, providing additional unlabeled data (beyond the test set) boosts the manifold ranking performances reported below by 1-2%.

city	feature set	SVM			Manifold		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall
Chicago	N-GRAMS	0.831	0.844	0.818	0.835	0.844	0.825
	LDA	0.833	0.846	0.819	0.817	0.832	0.802
	RLDA	0.830	0.838	0.822	0.841	0.819	0.863
Non-Chicago	N-GRAMS	0.728	0.744	0.714	0.733	0.738	0.727
	LDA	0.714	0.696	0.732	0.728	0.715	0.741
	RLDA	0.791	0.799	0.780	0.801	0.787	0.815

Table 2: Binary classification results showing that n-gram features overfit to the CHICAGO training data. Results correspond to evaluation on reviews for the two Chicago hotels from FOUR-CITIES and non-Chicago FOUR-CITIES reviews (six hotels).

to five for all topic-model-based approaches. Each baseline makes review-level predictions and then ranks each hotel by the average of those predictions.

- **Review-SVR:** Uses linear Tranductive Support Vector Regression with unigram and bigram features, similar to Ott et al. (2011).
- **Review-SVR+LDA (R):** Similar to REVIEW-SVR but uses our revised LDA (RLDA) topic model for dimensionality reduction **(R)**.
- **Two-Layer Manifold (S):** A simplified version of our model where the hotel-level is removed from the graph. Dimensionality reduction is performed using standard LDA **(S)**.
- **Two-Layer Manifold (R):** Similar to TWO-LAYER MANIFOLD (S) but uses the revised LDA (RLDA) model for dimensionality reduction.
- **Three-layer Manifold (tf-idf):** Our three-layer manifold ranking model, except with each review represented as a TF-IDF term vector. Review similarity is calculated based on the cosine similarity between these vectors.

Evaluation Method. To evaluate ranking performance in the absence of a gold standard set of manipulated hotels, we rearrange the FOUR-CITIES test set of 40 truthful and 40 deceptive reviews for each of eight hotels: we create 41 *versions* of each hotel, where each hotel version contains a different number of injected deceptive reviews, ranging from 0 to 40. For example, the first version of a hotel will have 40 truthful and 0 deceptive reviews, the second version 39 truthful and 1 deceptive, and the 41st version 0 truthful and 40 deceptive. In total, we generate $41 \times 8 = 328$ versions of hotel reviews. We expect versions with larger proportions of deceptive reviews to receive lower scores by the ranking models (i.e., they are ranked higher/more deceptive).

Metrics. To qualitatively evaluate the ranking results, we use the Normalized Discounted Cumulative Gain (NDCG), which is commonly used to evaluate retrieval algorithms with respect to an ideal relevance-based ranking. In particular, NDCG rewards rankings with the most relevant results at the top positions (Liu, 2009), which is also our objective, namely, to rank versions that have higher proportions of deceptive reviews nearer to the top.

Let $R(m)$ denote the relevance score of m^{th} ranked hotel version. Then, NDCG_N is defined as:

$$NDCG_N = \frac{1}{IDCG_N} \sum_{m=1}^{m=N} \frac{2^{R(m)} - 1}{log_2(1+m)} \quad (8)$$

where $IDCG_N$ refers to discounted cumulative gain (DCG) of the ideal ranking of the top N results. We define the ideal ranking according to the proportion of deceptive reviews in different versions, and report NDCG scores for the Nth ranked hotel versions ($N = 8$ to 321), at intervals of 8 (to account for ties among the eight hotels).

Results and Discussion. NDCG results are shown in Figure 5. We observe that our approach (using 2, 5 or 10 topics) generally outperforms the other approaches. In particular, approaches that use our RLDA text representation (OUR APPROACH, TWO-LAYER MANIFOLD (R), and REVIEW-SVR+LDA (R)), which tries to remove city- and hotel-specific information, perform better than those that use the full vocabulary (REVIEW-SVR, TWO-LAYER MANIFOLD (S), and THREE-LAYER MANIFOLD (TF-IDF)). This further confirms that our RLDA dimensionality reduction technique allows models,

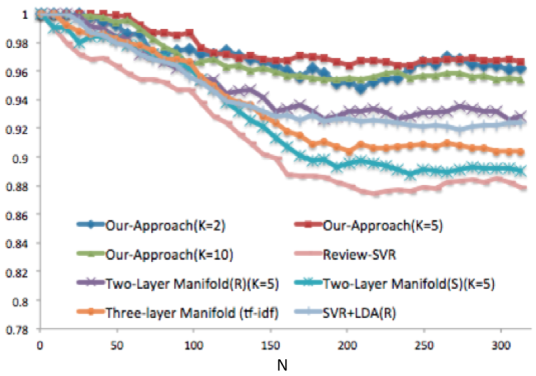


Figure 5: NDCG_N results for different approaches. K indicates the number of topics.

trained on limited data, to generalize to reviews of different hotels and in different locations. We also find that approaches that model a reinforcing relationship between hotels and their reviews are better than approaches that model reviews as independent units, e.g., TWO-LAYER MANIFOLD (R) vs. REVIEW-SVR+LDA and TWO-LAYER MANIFOLD (S) vs. REVIEW-SVR. This confirms our intuition that a hotel is more deceptive if it is connected with many deceptive reviews, and, in turn, a review is more deceptive if from a deceptive hotel.

8 Qualitative Evaluation

We now present qualitative evaluations for the RLDA topic model and the manifold ranking model.

Topic Quality. Table 3 gives the top words for four aspect topics and four city-specific topics in the RLDA topic model; Table 4 gives the highest and lowest ranking term weights in our three-layer manifold model. By comparing the first row of topics in Table 3, corresponding to aspect topics, to the top words in Table 4, we observe that the learned topics relate to truthful and deceptive classes. For example, Topics 1 and 4 share many terms with the top truthful terms in the manifold model, e.g., spatial terms, such as `location`, `floor` and `block`, and punctuation, such as `(`, `)`, and `$`. Similarly, Topics 2 and 7 share many terms with the top deceptive terms in the manifold model, e.g., `hotel`, `husband`, `wife`, `amazing`, `experience` and `recommend`. This makes sense, since topic models have been shown to produce discriminative topics on this data in previous work (Li et al., 2013).

Topic1	**Topic2**	**Topic4**	**Topic7**
location	hotel	(	hotel
$	stay	room	service
walk	staff	)	husband
night	restaurant	park	amazing
block	friendly	bed	will
floor	room	night	weekend
quiet	recommend	shower	friendly
nice	love	view	travel
lobby	excellent	minute	experience
breakfast	wife	pillow	friend
NYC	**Chicago**	**LA**	**Houston**
York	Chicago	los	Houston
ny	Michigan	Angeles	downtown
time	mile	la	Texas
square	tower	lax	cab
nyc	Illinois	shuttling	Westside
street	avenue	hollywood	center
empire	Rogers	plane	Northwest
Chinatown	river	morning	st
station	Burnham	California	museum
Wall	Goodman	downtown	mission

Table 3: Top words in topics extracted from RLDA topic model (see Section 4). The top row presents topic words from four aspect topics ($K = 10$) and the bottom row presents top words from four city-specific topics.

Deceptive		**Truthful**	
term	**score**	**term**	**score**
my	-1.063	$	0.964
visit	-0.944	location	0.922
we	-0.882	(	0.884
hotel	-0.863	)	0.884
husband	-0.828	bathroom	0.842
family	-0.824	floor	0.810
amazing	-0.782	breakfast	0.784
experience	-0.740	bar	0.762
recommend	-0.732	block	0.747
wife	-0.680	small	0.721
relax	-0.678	but	0.720
vacation	-0.651	walk	0.707
will	-0.651	lobby	0.707
friendly	-0.646	quiet	0.684

Table 4: Term scores from our ranking algorithm.

With respect to the second row in Table 4, containing top words from city-specific topics, we observe that each topic does contain primarily city-specific information. This helps to explain why removing terms associated with these topics resulted in a better vector representation for reviews.

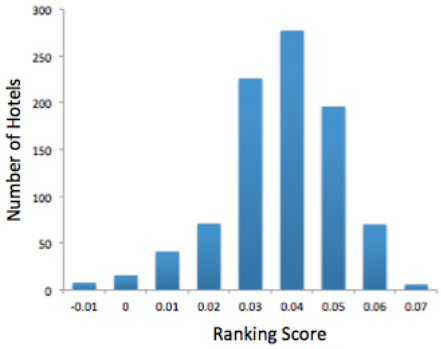


Figure 6: Hotel Ranking Distribution on TripAdvisor

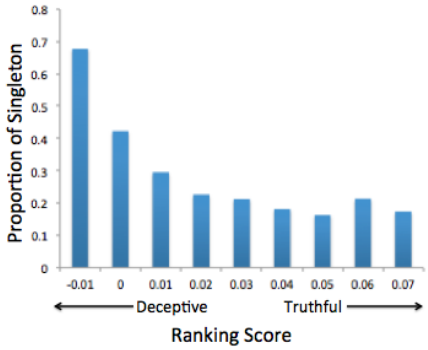


Figure 7: Proportion of Singletons vs. Hotel Ranking.

Real-world Evaluation. Finally, we apply our ranking model to a large-scale collection of real-world reviews from TripAdvisor. We crawl 878,561 reviews from 3,945 hotels in 25 US cities from TripAdvisor excluding all non-5-star reviews and removing hotels with fewer than 100 reviews. In the end, we collect 244,810 reviews from 838 hotels.

We apply our manifold ranking model and rank all 838 hotels. First, we present a histogram of the resulting manifold ranking scores in Figure 6. We observe that the distribution reaches a peak around 0.04, which in our quantitative evaluation (Section 7) corresponded to a hotel with 34 truthful and 6 deceptive reviews. These results suggest that the majority of reviews in TripAdvisor are truthful, in line with previous findings by Ott et al. (2011).

Next, we note that previous work has hypothesized that deceptive reviews are more likely to be posted by first-time review writers, or *singleton* reviewers (Ott et al, 2011; Wu et al, 2011). Accordingly, if this hypothesis were valid, then manipulated hotels would have an above-average proportion of singleton reviews. Figure 7 shows a histogram of the average proportion of singleton reviews, as a function of the ranking scores produced by our model. Noting that lower scores correspond to a higher predicted proportion of deceptive reviews, we observe that hotels that are ranked as being more deceptive by our model have much higher proportions of singleton reviews, on average, compared to hotels ranked as less deceptive.

9 Conclusion

We study the problem of identifying manipulated offerings on review portals and propose a novel three-layer graph model, based on manifold ranking for ranking offerings based on the proportion of reviews expected to be instances of deceptive opinion spam. Experimental results illustrate the effectiveness of our model over several learning-based baselines.

Acknowledgments

This work was supported in part by National Science Foundation Grant BCS-0904822, a DARPA Deft grant, as well as a gift from Google. We also thank the EMNLP reviewers for their helpful comments and advice.

References

David Blei, Ng Andrew and Michael Jordan. Latent Dirichlet allocation. 2003. In *Journal of Machine Learning Research*.

Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini and Sebastiano Vigna. A reference collection for web spam. In *ACM Sigir Forum*. 2006.

Paul-Alexandru Chirita, Jrg Diederich and Wolfgang Nejdl. MailRank: using ranking for spam detection. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 2005.

Cone. 2011 Online Influence Trend Tracker. http://www.coneinc.com/negative-reviews-online-reverse-purchase-decisions. August.

Yajuan Duan, Zhumin Chen, Furu Wei, Ming Zhou and Heung-Yeung Shum. Twitter Topic Summarization by Ranking Tweets Using Social Influence and Content Quality. In *Proceedings of 24th International Conference on Computational Linguistics* 2012.

Federal Trade Commission. Guides Concerning Use of Endorsements and Testimonials in Advertising. In *FTC 16 CFR Part 255*. 2009.

Socialogue: Five Stars? Thumbs Up? A+ or Just Average? URL:http://www.ipsos-na.com/news-polls/pressrelease.aspx?id=5929g

Nitin Jindal, and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. 2008.

Nitin Jindal, Bing Liu and Ee-Peng Lim. Finding Unusual Review Patterns Using Unexpected Rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*.2010.

Thorsten Joachims. Making large-scale support vector machine learning practical. In *Advances in kernel methods.*1999.

Fangtao Li, Minlie Huang, Yi Yang and Xiaoyan Zhu. Learning to identify review Spam. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. 2011.

Jiwei Li, Claire Cardie and Sujian Li. TopicSpam: a Topic-Model-Based Approach for Spam Detection. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguis- tics.* 2013.

Peng Li, Jing Jiang and Yinglin Wang. Generating templates of entity summaries with an entity-aspect model and pattern mining. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.* 2010.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting Product Review Spammers Using Rating Behavior. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. 2010.

Tieyan Liu. Learning to Rank for Information Retrieval. In *Foundations and Trends in Information Retrieval* 2009.

Arjun Mukherjee, Bing Liu and Natalie Glance. Spotting Fake Reviewer Groups in Consumer Reviews . In *Proceedings of the 21st international conference on World Wide Web.* 2012.

Juan Martinez-Romo and Lourdes Araujo. Web spam identification through language model analysis. In *Proceedings of the 5th international workshop on adversarial information retrieval on the web.* 2009.

Myle Ott, Claire Cardie and Jeffrey Hancock. Estimating the Prevalence of Deception in Online Review Communities. In *Proceedings of the 21st international conference on World Wide Web*. 2012.

Myle Ott, Yejin Choi, Claire Cardie and Jeffrey Hancock. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.* 2011.

Daniel Ramage, David Hall, Ramesh Nallapati and Christopher Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.* 2009.

Michal Rosen-zvi, Thomas Griffith, Mark Steyvers and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence.*2004.

Xiaojun Wan and Jianwu Yang. Multi-Document Summarization Using Cluster-Based Link Analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.* 2008.

Xiaojun Wan, Jianwu Yang and Jianguo Xiao Manifold-Ranking Based Topic-Focused Multi-Document Summarization. In *Proceedings of International Joint Conferences on Artificial Intelligence*,2007.

Guan Wang, Sihong Xie, Bing Liu and Philip Yu. Review Graph based Online Store Review Spammer Detection. In *Proceedings of International Conference of Data Mining.* 2011.

Guangyu Wu, Derek Greene and , Padraig Cunningham. Merging multiple criteria to identify suspicious reviews. In *Proceedings of the fourth ACM conference on Recommender systems.* 2011.

Kyung-Hyan Yoo and Ulrike Gretzel. Comparison of Deceptive and Truthful Travel Reviews. In *Information and Communication Technologies in Tourism.* 2009.

Dengyong Zhou, Olivier Bousquet, Thomas Navin and Jason Weston. Learning with local and global consistency. In *Proceedings of Advances in neural information processing systems.*2003.

Dengyong Zhou, Jason Weston, Arthur Gretton and Olivier Bousquet. Ranking on data manifolds. In *Proceedings of Advances in neural information processing systems.*2003.

Well-argued recommendation: adaptive models based on words in recommender systems

Julien Gaillard
University of Avignon
Agorantic
Avignon, France
julien.gaillard@univ-avignon.fr

Marc El-Beze
University of Avignon
Agorantic
Avignon, France
marc.elbeze@univ-avignon.fr

Eitan Altman
INRIA Sophia Antipolis
Agorantic
Sophia-Antipolis, France
eitan.altman@inria.fr

Emmanuel Ethis
University of Avignon
Agorantic
Avignon, France
emmanuel.ethis@univ-avignon.fr

Abstract

Recommendation systems (RS) take advantage of products and users information in order to propose items to consumers. Collaborative, content-based and a few hybrid RS have been developed in the past. In contrast, we propose a new domain-independent semantic RS. By providing textually well-argued recommendations, we aim to give more responsibility to the end user in his decision. The system includes a new similarity measure keeping up both the accuracy of rating predictions and coverage. We propose an innovative way to apply a fast adaptation scheme at a semantic level, providing recommendations and arguments in phase with the very recent past. We have performed several experiments on films data, providing textually well-argued recommendations.

1 Introduction

Recommender systems aim at suggesting appropriate items to users from a large catalog of products. Those systems are individually adapted by using a specific profile for each user and item, derived from the analysis of past ratings. The last decade has shown a historical change in the way we consume products. People are getting used to receive recommendations. Nevertheless, after a few bad recommendations, users will not be convinced anymore by the RS. Moreover, if these suggestions come without explanations, why people should trust it? Numbers and figures cannot talk to people.

To answer these key issues, we have designed a new semantic recommender sytem (SRS) including at least two innovative features:

- Argumentation: each recommendation relies on and comes along with a textual argumentation, providing the reasons that led to that recommendation.
- Fast adaptation: the system is updated in a continuous way, as each new review is posted.

In doing so, the system will be perceived as less intrusive thanks to well-chosen words and its failures will be smoothed over. It is therefore necessary to design a new generation of RS providing textually well-argued recommendations. This way, the end user will have more elements to make a well-informed choice. Moreover, the system parameters have to be dynamically and continuously updated, in order to provide recommendations and arguments in phase with the very recent past. To do so, we have adapted the algorithms we described in Gaillard (Gaillard et al., 2013), by including a semantic level, i.e words, terms and phrases as they are naturally expressed in reviews.

This paper is structured as follows. In the next section, we present the state of the art in recommendation systems and introduce some of the improvements we have made. Then, we present our approach and define the associated methods in section 3. We describe the evaluation protocol and how we have performed some experiments in section 4. Finally we report results including a comparison to a baseline in section 5.

2 Related work and choice of a baseline

We present here some methods used in the literature. Collaborative Filtering (CF) systems use logs

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1943–1947,
Seattle, Washington, USA, 18-21 October 2013.

of users, generally user ratings on items (Burke, 2007; Sarwar et al., 1998). In these systems, the following assumption is made: if user a and user b rate n items similarly, they will rate other items in the same way (Deshpande and Karypis., 2004). This technique has many well-known issues such as the "cold start" problem, i.e when new items or users appear, it is impossible to make a recommendation, due to the absence of rating data (Schein et al., 2002). Other limitations of RS are sparsity, scalability, overspecialization and domain-dependency problems.
In Content Based Filtering (CBF) systems, users are supposed to be independent (Mehta et al., 2008). Hence for a given user, recommendations rely only on items he previously rated.

Some RS incorporate semantic knowledge to improve quality. Generally, they apply a concept-based approach to enhance the user modeling stage and employ standard vocabularies and ontology resources. For instance, ePaper (scientific-paper recommender), computes the matching between the concepts constituting user interests and the concepts describing an item by using hierarchical relationships of domain concepts (Maidel et al., 2008). Codina and Ceccaroni (2010) propose to take advantage of semantics by using an interest-prediction method based on user ratings and browsing events.
However, none of them are actually based on the user opinion as it is expressed in natural language.

2.1 Similarity measures

Similarity measures are the keystone of RS (Herlocker et al., 2005). Resnick (1997) was one of the first to introduce the Pearson correlation coefficient to derive a similarity measure between two entities. Other similarity measures such as Jaccard and Cosine have been proposed (Meyer, 2012). Let S_u be the set of items rated by u, T_i the set of users who have rated item i, $r_{u,i}$ the rating of user u on item i and $\overline{r_x}$ the mean of x (user or item). PEA(i,j) stands for the Pearson similarity between items i and j and is computed as follows:

$$\frac{\sum_{u \in T_i \cap T_j}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in T_i \cap T_j}(r_{u,i} - \overline{r_i})^2 \sum_{u \in T_i \cap T_j}(r_{u,j} - \overline{r_j})^2}} \quad (1)$$

In the remainder, the Pearson similarity measure will be used as a baseline. The Manhattan Weighted and Corrected similarity (MWC), that we introduced in (Gaillard et al., 2013), will be used as a point of comparison as well[1]. Again, for none of them, textual content is taken into account.

2.2 Rating prediction

Let i be a given item and u a given user. We suppose the pair (u, i) is unique. Indeed, most of social networks do not allow multiple ratings by the same user for one item. In this framework, two rating prediction methods have to be defined: one user oriented and the other item oriented. *Sim* stands for some similarity function in the following formula.

$$rating(u,i) = \frac{\sum_{v \in T_i} Sim(u,v) \times r_{v,i}}{\sum_{v \in T_i} |Sim(u,v)|} \quad (2)$$

A symmetrical formula for items $rating(i,u)$ is derived from and combined with (2).

$$\hat{r}_{u,i} = \beta \times rating(u,i) + (1-\beta) \times rating(i,u) \quad (3)$$

3 Methods

In this section, we describe the methods we have used and propose some of the enhancements we have elaborated in our system. In formula (2), Sim can be replaced by several similarity such as Pearson, Cosine or MWC similarity (Tan et al., 2005). All these methods provide a measurement of the likeness between two objects. We then conclude if two users (or items) are "alike" or not. One has to define what "alike" should mean in this case. If two users rate the same movies with equals ratings, then these similarities will be maximal. However, they may have rated identically but for completely different reasons, making them not alike at all. Moreover, none of these similarity measures can express why two users or items are similar. This is due to the fact that they rely on ratings only.

3.1 New similarity based on words

We propose a new similarity method, taking into account words used by users in their past reviews about items. In the remainder, we call it the *Word Based Similarity* (WBS). Each user x (or item) has a vocabulary set V_x and each word w in it is associated

[1] Details on MWC can be found in supplementary material.

with a set of ratings $\mathbf{R}_{w,x}$ and an average usage rating $\overline{r_w}$. In order to balance the contribution of each word, we define a weight function F_w, mixing the well-known Inverse Document Frequency $IDF(w)$ with the variance σ_w^2. Common words and words w associated with very heterogenous ratings $\mathbf{R}_{w,x}$ (i.e a high variance) will have a smaller weight in the similarity. N_w is the number of items in which the word w appears. N_{tot} is the total number of items. D is the maximum difference between two ratings. Note that F_w has to be updated at each iteration.

$$F_w = -log\left(\frac{N_w}{N_{tot}}\right) \times \frac{1}{\sigma_w^2} \tag{4}$$

$$WBS(x,y) = \frac{\sum_{w \in V_x \cap V_y} (D - |\overline{r_{w,x}} - \overline{r_{w,y}}|) F_w}{D \times |V_x \cap V_y| \sum_{w \in V_x \cap V_y} F_w} \tag{5}$$

3.2 Adaptation

An adaptive framework proposed in (Gaillard et al., 2013) allows the system to have a dynamic adaptation along time, overcoming most of the drawbacks due to the cold-start. The authors have designed a dynamic process following the principle that every update (u, i) needs to be instantly taken into account by the system. Consequently, we have to update the σ_w^2 and IDF(w) at each iteration, for every word. Paying attention to avoid a whole re-estimation of these two variables, we derived an iterative relation for the two of them[2]. We thus reduced the complexity by one degree, keeping our system very well-fitted to dynamic adaptation.

3.3 Textual recommendation

The main innovative feature of our proposal is to predict what a user is going to write on an item we recommend. More precisely, we can tell the user why he is expected to like or dislike the recommended item. This is possible thanks to the new similarity measure we have introduced (WBS). Let us consider a user u and an item i. To keep it simple, the system takes into account what u has written on other items in the past and what other users have written on item i, by using WBS. The idea consists in extracting what elements of i have been liked or disliked by other users, and what u generally likes. At the intersection of these two pieces of information, we extract a set of matching words that we sort by relevance using F_w. Then, by taking into account the ratings associated with each word, we define two sub-sets P_w and N_w. P_w contains what user u is probably going to like in i and N_w what u may dislike. Finally, we provide the most relevant arguments contained in both P_w and N_w, and each of them is given in the context they have been used for item i. As an example, some outputs are shown in section 5.2.

[2]More details can be found in the supplementary material.

4 Evaluation criteria

We present here the evaluation protocol we designed. It should be noted that we are not able to make online experiments. Therefore, we can not measure the feedback on our recommendations. However, the cornerstone of recommender system is the accuracy of rating predictions (Herlocker et al., 2004). From this point of view, one could argue that the quality of a recommender engine could be assessed by its capacity to predict ratings. It is thus possible to evaluate our system comparing the prediction $\hat{r}_{u,i}$ for a given pair (u, i), with the actual real rating $r_{u,i}$.

The classical metrics[3] (Bell et al., 2007) *Root Mean Square Error* (RMSE) and *Mean Absolute Error* (MAE) will be used to evaluate our RS.

Last but not least, we make the following assumption: if WBS results are as good as MWC's, the words presented by the system to users as arguments are likely to be relevant.

5 Experiments

This work has been carried out in partnership with the website Vodkaster [4], a Cinema social network. Researchers have used other datasets such as the famous Netflix. Unfortunately, the latter does not include textual reviews. It is therefore strictly impossible to experiment a SRS on such a dataset.

5.1 Corpus

The corpus has been extracted from Vodkaster's database. Users post *micro-reviews* (MR) to express their opinion on a movie and rate it, within a

[3]Details on metrics are given in the supplementary material.

[4]www.vodkaster.com

140 characters Twitter-like length limit. We divided the corpus into three parts, chronologically sorted: training (Tr), development (D) and test (T). Note that in our experiments, the date is taken into account since we also work on dynamic adaptation.

	Tr	D	Tr+D	T
Size	55486	9892	65378	9729
Nb of Films	8414	3184	9130	3877
Nb of Users	1627	675	1855	706

Table 1: Statistics on the corpus

5.2 Results

Figure 1 compares four different methods: the classical Pearson (PEA) method that does not allow quick adaptation, the MWC method with and without quick adaptation MNA and ours (WBS). Within the confidence interval, in terms of accuracy,

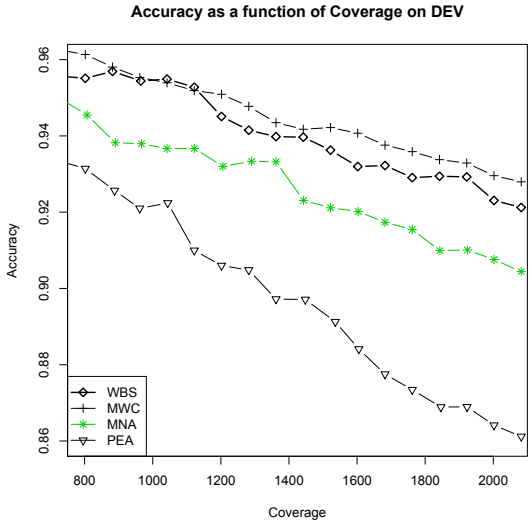


Figure 1: Evolution of accuracy as a function of coverage for PEA, MWC and WBS methods on D corpus.

the same performances are obtained by MWC and WBS. Both outperform[5] PEA and MNA. Our word based approach is thus able to offer the arguments feature without any loss of perfomances with respect to any others RS methods that we know of.

In Table 2, we set a constant coverage (2000 predictions) in order to be able to compare results obtained with different methods.

Corp.	Met.	RMSE	MAE	%Acc.	CI
D	PEA	0.99	0.76	86.41	1.49
E	MNA	0.93	0.72	90.75	1.26
V	MWC	0.89	0.69	92.95	1.12
	WBS	0.89	0.70	92.45	1.16
T	PEA	1.01	0.78	86.02	1.51
E	MNA	0.98	0.75	90.04	1.30
S	MWC	0.92	0.71	91.46	1.22
T	WBS	0.94	0.72	91.15	1.24

Table 2: Results with Pearson (PEA), MWC, MWC without Adaptation (MNA), WBS. CI is the radius confidence interval estimated in % on accuracy (Acc.).

MNA (MWC without adaptation) being better and more easily updated than Pearson (PEA), we have decided to use the adaptive framework only for MWC. Moreover, for Pearson dynamic adaptation, the updating algorithm complexity is increased by one degree.
We want to point out that the results are the same for both MWC and WBS methods, within a confidence interval (CI) radius of 1.16%. From a qualitative point of view, these results can be seen as an assessment of our approach based on words.

Example of outputs: The movie *Apocalypse Now* is recommended to user Theo6 with a rating prediction equal to 4.3. Why he might like: *some brillant moments* (0.99), *among the major masterpiece* (0.91), *Vietnam's hell* (0.8); dislike: *did not understand everything but...* (0.71).

The data we have does not contain the information on the reaction of the user to the recommendation. In particular, we do not know if the textual argumentation would have been sufficient for convincing Theo6 to see the film. But we know that after seeing it, he put a good rating (4.5/5) on this movie.

[5]Note that the key point here is the comparison of results obtained with the baseline and with the method we propose. Both of them have been evaluated with the same protocol: RMSE is computed with respect to rating predictions above some empirical threshold as done in (Gaillard et al., 2013).

6 Conclusion and perspectives

We have presented an innovative proposal for designing a domain-independent SRS relying on a word based similarity function (WBS), providing textually well-argued recommendations to users. Moreover, this system has been developed in a dynamic and adaptive framework. This might be the first step really made towards an anthromorphic and evolutive recommender. As future work, we plan to evaluate how the quality is impacted by the time dimension (adaptation delay, cache reset,etc.).

Acknowledgment

The authors would like to thank Vodkaster for providing the data.
This work has been partly supported by the European Commission within the framework of the CONGAS Project (FP7- ICT-2011-8-317672), see www.congas-project.eu.

References

R. Bell, Y. Koren and C. Volinsky. 2007. *The BellKor 2008 Solution to the Netflix Prize.* The Netflix Prize.

R. Burke. 2007. *Hybrid Web Recommender Systems.* The Adaptive Web, 377–408.

V. Codina and Luigi Ceccaroni. 2010. *Taking Advantage of Semantics in Recommendation Systems.* Proceedings of the 13th International Conference of the Catalan Association for A.I.,163–172

M. Deshpande and G. Karypis. 2004. *Item based top-N recommendation algorithms.* ACM Transactions on Information and System Security.

J. Gaillard, M. El-Beze, E. Altman and E. Ethis. 2013. *Flash reactivity: adaptive models in recommender systems.* International Conference on Data Mining (DMIN), WORLDCOMP.

J. Herlocker, J.A Konstan, L. Terveen and J. Riedl. 2004. *Evaluating collaborative filtering recommender systems.* ACM Transactions on Information Systems (TOIS).

V. Maidel, P. Shoval, B. Shapira, M. Taieb-Maimon. 2008. *Evaluation of an ontology-content based filtering method for a personalized newspaper.* RecSys'08: Proceedings, 91–98.

B. Mehta, T. Hofmann, and W. Nejdl. 2008. *Robust collaborative filtering.* In RecSys

F. Meyer. 2012. *Recommender systems in industrial contexts.* PhD thesis, University of Grenoble, France.

P. Resnick and R. Varian Hal. 1997. *Recommender systems (introduction to special section.)* Communications of the ACM

B.M Sarwar, J.A Konstan, A. Borchers,J. Herlocker, B. Miller, J. Riedl 1998. *Using filtering agents to improve prediction quality in the groupLens research collaborative filtering system.* Proceedings of the ACM Conference on Computer Supported Cooperative Work

A.I Schein, A. Popescul and L.H Ungar. 2002. *Methods and metrics for cold-start recommendations.* ACM SIGIR Conference on Research and Development in Information Retrieval.

P. Tan, M. Steinbach and V. Kumar. 2005 *Introduction to Data Mining.* Addison-Wesley, 500–524.

C. Ziegler, S.M McNee, J.A Konstan and G. Lausen. 2005. *Improving recommendation lists through topic diversification.* Fourteenth International World Wide Web Conference

Regularized Minimum Error Rate Training

Michel Galley
Microsoft Research
mgalley@microsoft.com

Chris Quirk
Microsoft Research
chrisq@microsoft.com

Colin Cherry
National Research Council
colin.cherry@nrc-cnrc.gc.ca

Kristina Toutanova
Microsoft Research
kristout@microsoft.com

Abstract

Minimum Error Rate Training (MERT) remains one of the preferred methods for tuning linear parameters in machine translation systems, yet it faces significant issues. First, MERT is an unregularized learner and is therefore prone to overfitting. Second, it is commonly used on a noisy, non-convex loss function that becomes more difficult to optimize as the number of parameters increases. To address these issues, we study the addition of a regularization term to the MERT objective function. Since standard regularizers such as ℓ_2 are inapplicable to MERT due to the scale invariance of its objective function, we turn to two regularizers—ℓ_0 and a modification of ℓ_2—and present methods for efficiently integrating them during search. To improve search in large parameter spaces, we also present a new direction finding algorithm that uses the gradient of expected BLEU to orient MERT's exact line searches. Experiments with up to 3600 features show that these extensions of MERT yield results comparable to PRO, a learner often used with large feature sets.

1 Introduction

Minimum Error Rate Training emerged a decade ago (Och, 2003) as a superior training method for small numbers of linear model parameters of machine translation systems, improving over prior work using maximum likelihood criteria (Och and Ney, 2002). This technique quickly rose to prominence, becoming standard in many research and commercial MT systems. Variants operating over lattices (Macherey et al., 2008) or hypergraphs (Kumar et al., 2009) were subsequently developed, with the benefit of reducing the approximation error from n-best lists.

The primary advantages of MERT are twofold. It directly optimizes the evaluation metric under consideration (e.g., BLEU) instead of some surrogate loss. Secondly, it offers a globally optimal line search. Unfortunately, there are several potential difficulties in scaling MERT to larger numbers of features, due to its non-convex loss function and its lack of regularization. These challenges have prompted some researchers to move away from MERT, in favor of linearly decomposable approximations of the evaluation metric (Chiang et al., 2009; Hopkins and May, 2011; Cherry and Foster, 2012), which correspond to easier optimization problems and which naturally incorporate regularization. In particular, recent work (Chiang et al., 2009) has shown that adding thousands or tens of thousands of features can improve MT quality when weights are optimized using a margin-based approximation. On simulated datasets, Hopkins and May (2011) found that conventional MERT struggles to find reasonable parameter vectors, where a smooth loss function based on Pairwise Ranking Optimization (PRO) performs much better; on real data, this PRO method appears at least as good as MERT on small feature sets, and also scales better as the number of features increases.

In this paper, we seek to preserve the advantages of MERT while addressing its shortcomings in terms of regularization and search. The idea of adding a regularization term to the MERT objective function can be perplexing at first, because the most common regularizers, such as ℓ_1 and ℓ_2, are not directly applicable to MERT. Indeed, these regularizers are *scale sensitive*, while the MERT objective function is not: scaling the weight vector neither changes the predictions of the linear model nor affects the error count. Hence, MERT can hedge any regularization penalty by maximally scaling down linear model weights.

The first contribution of this paper is to analyze various forms of regularization that are not susceptible to this scaling problem. We analyze and experiment with ℓ_0, a form of regularization that is *scale insensitive*. We also present new parameterizations of ℓ_2

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1948–1959,
Seattle, Washington, USA, 18-21 October 2013.

regularization, where we apply ℓ_2 regularization to scale-senstive linear transforms of the original linear model. In addition, we introduce efficient methods of incorporating regularization in Och (2003)'s exact line searches. For all of these regularizers, our methods let us find the true optimum of the regularized objective function along the line.

Finally, we address the issue of searching in a high-dimensional space by using the gradient of expected BLEU (Smith and Eisner, 2006) to find better search directions for our line searches. This direction finder addresses one of the serious concerns raised by Hopkins and May (2011): MERT widely failed to reach the optimum of a synthetic linear objective function. In replicating Hopkins and May's experiments, we confirm that existing search algorithms for MERT—including coordinate ascent, Powell's algorithm (Powell, 1964), and random direction sets (Cer et al., 2008)—perform poorly in this experimental condition. However, when using our gradient-based direction finder, MERT has no problem finding the true optimum even in a 1000-dimensional space.

Our results suggest that the combination of a regularized objective function and a gradient-informed line search algorithm enables MERT to scale well with a large number of features. Experiments with up to 3600 features show that these extensions of MERT yield results comparable to PRO (Hopkins and May, 2011), a parameter tuning method known to be effective with large feature sets.

2 Unregularized MERT

Prior to introducing regularized MERT, we briefly review standard unregularized MERT (Och, 2003). We use $\mathbf{f}_1^S = \{\mathbf{f}_1 \dots \mathbf{f}_S\}$ to denote the S input sentences of a given tuning set. For each sentence $\mathbf{f}_s$, let $\mathbf{C}_s = \{\mathbf{e}_{s,1} \dots \mathbf{e}_{s,M}\}$ denote the list of M-best candidate translations. Each input and output sentence pair $(\mathbf{f}_s, \mathbf{e}_{s,m})$ is weighted using a linear model that applies model parameters $\mathbf{w} = (w_1 \dots w_D) \in \mathbb{R}^D$ to D feature functions $h_1(\mathbf{f}, \mathbf{e}, \sim) \dots h_D(\mathbf{f}, \mathbf{e}, \sim)$, where $\sim$ is the hidden state associated with the derivation from $\mathbf{f}$ to $\mathbf{e}$, such as phrase segmentation and alignment. Furthermore, let $\mathbf{h}_{s,m} \in \mathbb{R}^D$ denote the feature vector representing the translation pair $(\mathbf{f}_s, \mathbf{e}_{s,m})$.

In MERT, the goal is to minimize a loss function $E(\mathbf{r}, \mathbf{e})$ that scores translation hypotheses against a set of reference translations $\mathbf{r}_1^S = \{\mathbf{r}_1 \dots \mathbf{r}_S\}$. This yields the following optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) \right\} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} \sum_{m=1}^{M} E(\mathbf{r}_s, \mathbf{e}_{s,m}) \delta(\mathbf{e}_{s,m}, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) \right\} \quad (1)$$

where

$$\hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w}) = \arg\max_{m \in \{1 \dots M\}} \left\{ \mathbf{w}^\intercal \mathbf{h}_{s,m} \right\} \quad (2)$$

While the error surface of Equation 1 is only an approximation of the true error surface of the MT decoder, the quality of this approximation depends on the size of the hypothesis space represented by the M-best list. Therefore, the hypothesis list is grown iteratively: decoding with an initial parameter vector seeds the M-best lists; next, parameter estimation and M-best list gathering alternate until the cumulative M-best list no longer grows, or until changes of $\mathbf{w}$ between two decoding runs are deemed too small. To increase the size of the hypothesis space, subsequent work (Macherey et al., 2008) instead operated on lattices, but this paper focuses on M-best lists.

A crucial observation is that the unsmoothed error count represented in Equation 1 is a piecewise constant function. This enabled Och (2003) to devise a line search algorithm guaranteed to find the optimum point along the line. To extend the search from one to multiple dimensions, MERT applies a sequence of line optimizations along some fixed or variable set of search directions $\{\mathbf{d}_t\}$ until some convergence criteria are met. Considering a given point $\mathbf{w}_t$ and a given direction $\mathbf{d}_t$ at iteration t, finding the most probable translation hypothesis in the set of candidates translations $\mathbf{C}_s = \{\mathbf{e}_{s,1} \dots \mathbf{e}_{s,M}\}$ corresponds to solving the following optimization problem:

$$\hat{\mathbf{e}}(\mathbf{f}_s; \gamma) = \arg\max_{m \in \{1 \dots M\}} \left\{ (\mathbf{w}_t + \gamma \cdot \mathbf{d}_t)^\intercal \mathbf{h}_{s,m} \right\} \quad (3)$$

The function in this equation is piecewise linear (Papineni, 1999), which enables an efficient exhaustive computation. Specifically, this function is optimized by enumerating the up to M hypotheses that form the *upper envelope* of the model score function. The error count, then, is a piecewise constant function

defined by the points $\gamma_1^{\mathbf{f}_s} < \cdots < \gamma_M^{\mathbf{f}_s}$ at which an increase in γ causes a change of optimum in Equation 3. Error counts for the whole corpus are simply the sums of sentence-level piecewise constant functions aggregated over all sentences of the corpus.[1] The optimal γ is finally computed by enumerating all piecewise constant intervals of the corpus-level error function, and by selecting the one that has the lowest error count (or, correspondingly, highest BLEU score). Assuming the optimum is found in the interval $[\gamma_{k-1}, \gamma_k]$, we define $\gamma_{\text{opt}} = (\gamma_{k-1} + \gamma_k)/2$ and change the parameters using the update $\mathbf{w}_{t+1} = \mathbf{w}_t + \gamma_{\text{opt}} \cdot \mathbf{d}_t$.

Finally, this method is turned into a global D-dimensional search using algorithms that repeatedly use the aforementioned exact line search algorithm. Och (2003) first advocated the use of Powell's method (Powell, 1964; Press et al., 2007). Pharaoh (Koehn, 2004) and subsequently Moses (Koehn et al., 2007) instead use coordinate ascent, and more recent work often uses random search directions (Cer et al., 2008; Macherey et al., 2008). In Section 4, we will present a novel direction finder for maximum-BLEU optimization, which uses the gradient of expected BLEU to find directions where the BLEU score is most likely to increase.

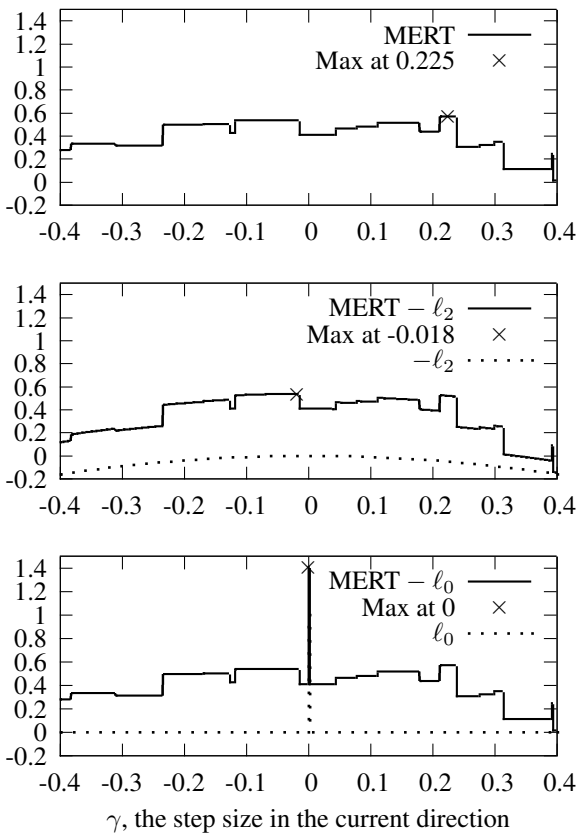


Figure 1: Example MERT values along one coordinate, first unregularized. When regularized with ℓ_2, the piecewise constant function becomes piecewise quadratic. When using ℓ_0, the function remains piecewise constant with a point discontinuity at 0.

3 Regularization for MERT

Because MERT is prone to overfitting when a large number of parameters must be optimized, we study the addition of a regularization term to the objective function. One conventional approach is to regularize the objective function with a penalty based on the Euclidean norm $||\mathbf{w}||_2 = \sqrt{\sum_i w_i^2}$, also known as ℓ_2 regularization. In the case of MERT, this yields the following objective function:[2]

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \mathbf{w})) + \frac{||\mathbf{w}||_2^2}{2\sigma^2} \right\} \quad (4)$$

where the regularization term $1/2\sigma^2$ is a free parameter that controls the strength of the regularization penalty. Similar regularizers have also been used in conjunction with other norms, such as ℓ_1 and ℓ_0 norms. The ℓ_1 norm, defined as $||\mathbf{w}||_1 = \sum_i |w_i|$, applies a constant force toward zero, preferring vectors with fewer non-zero components; ℓ_0, defined as $||\mathbf{w}||_0 = |\{i \mid w_i \neq 0\}|$, simply counts the number of non-zero components of the weight vector, encoding a preference for sparse vectors.

Geometrically, ℓ_2 is a parabola, ℓ_1 is the wedge-shaped absolute value function, and ℓ_0 is an impulse function with a spike at 0. The original formulation (Equation 1) of MERT consists of a piecewise constant representation of the loss, as a function of the step size in a given direction. But with these three reg-

[1] This assumes that the sufficient statistics of the metric under consideration are additively decomposable by sentence, which is the case with most popular evaluation metrics such as BLEU (Papineni et al., 2001).

[2] The ℓ_2 regularizer is often used in conjunction with log-likelihood objectives. The regularization term of Equation 4 could similarly be added to the log of an objective—e.g., log(BLEU) instead of BLEU—but we found that the distinction doesn't have much of an impact in practice.

ularization terms, the function respectively becomes piecewise quadratic, piecewise linear, or piecewise constant with a potential impulse jump for each distinct choice of regularizer. Figure 1 demonstrates this effect graphically.

As discussed in (McAllester and Keshet, 2011), the problem with optimizing Equation 4 directly is that the output of the underlying linear classifier, and therefore the error count, are not sensitive to the scale of $\mathbf{w}$. Moreover, ℓ_2 regularization (as well as ℓ_1 regularization) is scale sensitive, which means any optimizer of this function can drive the regularization term down to zero by scaling down $\mathbf{w}$. As special treatments for ℓ_2, we evaluate three linear transforms of the weight vector, where the vector $\mathbf{w}$ of the regularization term $||\mathbf{w}||_2^2/2\sigma^2$ is replaced with either:

1. an affine transform: $\mathbf{w} - \mathbf{w}_0$
2. a vector with only $(D-1)$ free parameters, e.g., $(1, w'_2, \cdots, w'_D)$
3. an ℓ_1 renormalization: $\mathbf{w}/||\mathbf{w}||_1$

In (1), regularization is biased towards $\mathbf{w}_0$, a weight vector previously optimized using a competitive yet much smaller feature set, such as core features of a phrase-based (Koehn et al., 2007) or hierarchical (Chiang, 2007) system. The requirement that this feature set be small is to prevent overfitting. Otherwise, any regularization toward an overfit parameter vector $\mathbf{w}_0$ would defeat the purpose of introducing a regularization term in the first place.[3] In (2), the transformation is motivated by the observation that the D-parameter linear model of Equation 2 only needs $(D-1)$ degrees of freedom. Fixing one of the components of $\mathbf{w}$ to any non-zero constant and allowing the others to vary, the new linear model retains the same modeling power, but the $(D-1)$ free parameters are no longer scale invariant, i.e., scaling the $(D-1)$-dimensional vector now has an effect on linear model predictions. In (3), the weight vector is normalized as to have an ℓ_1-norm equal to 1. In contrast, the ℓ_0 norm is scale insensitive, thus not affected by this problem.

3.1 Exact line search with regularization

Optimizing with a regularized error surface requires a change in the line search algorithm presented in Section 2, but the other aspects of MERT remain the same, and we can still use global search algorithms such as coordinate ascent, Powell, and random directions exactly the same way as with unregularized MERT. Line search with a regularization term is still as efficient as in (Och, 2003), and it is still guaranteed to find the optimum of the (now regularized) objective function along the line. Considering again a given point $\mathbf{w}_t$ and a given direction $\mathbf{d}_t$ at line search iteration t, finding the optimum γ_{opt} corresponds to finding γ that minimizes:

$$\sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \gamma)) + \frac{||\mathbf{w}_t + \gamma \cdot \mathbf{d}_t||_2^2}{2\sigma^2} \qquad (5)$$

Since regularization does not affect the points at which $\hat{\mathbf{e}}(\mathbf{f}_s; \gamma)$ changes its optimum, the points $\gamma_1^{\mathbf{f}_s} < \cdots < \gamma_M^{\mathbf{f}_s}$ of intersection in the upper envelope remain the same, so the points of discontinuity in the error surface remain the same. The difference now is that the error count on each segment $[\gamma_{i-1}, \gamma_i]$ is no longer constant. This means we need to adjust the final step of line search, which consists of enumerating all $[\gamma_{i-1}, \gamma_i]$, and keeping the optimum of Equation 5 for each segment. $\hat{\mathbf{e}}(\mathbf{f}_s; \gamma)$ remains constant within the segment, so we only need to consider the expression $||\mathbf{w}_t + \gamma \cdot \mathbf{d}_t||_2^2$ to select a segment point. The optimum is either at the left edge, the right edge, or in the middle if the vertex of the parabola happens to lie within that segment.[4] We compute this optimum by finding the value γ for which the derivative of the regularization term is zero. There is an easy closed-form solution:

$$\frac{\mathrm{d}}{\mathrm{d}\gamma}\left[\frac{||\mathbf{w}_t + \gamma \cdot \mathbf{d}_t||_2^2}{2\sigma^2}\right] = 0$$

$$\frac{\mathrm{d}}{\mathrm{d}\gamma}\left[\sum_i (w_{t,i}^2 + 2 \cdot \gamma \cdot w_{t,i} \cdot d_{t,i} + \gamma^2 \cdot d_{t,i}^2)\right] = 0$$

$$\sum_i (2 \cdot w_{t,i} \cdot d_{t,i} + 2 \cdot \gamma \cdot d_{t,i}^2) = 0$$

$$\gamma = -\big(\sum_i w_{t,i} \cdot d_{t,i}\big) / \big(\sum_i d_{t,i}^2\big) = -\frac{\mathbf{w}_t{}^\intercal \mathbf{d}_t}{\mathbf{d}_t{}^\intercal \mathbf{d}_t}$$

This closed-form solution is computed in time proportional to D, which doesn't slow down the com-

[3] (Gimpel and Smith, 2012, footnote 6) briefly mentions the use of such a regularizer with its ramp loss objective function.

[4] When the optimum is either at the left edge γ_{i-1} or right edge γ_i of a segment, we select a point at a small relative distance within the segment ($.999\gamma_{i-1} + .001\gamma_i$, in the former case) to avoid ties in objective values.

putation of Equation 5 for each segment (the construction of each segment of the upper envelope is proportional to D anyway).

We also use ℓ_0 regularization. While minimization of the ℓ_0-norm is known to be NP-hard in general (Hyder and Mahata, 2009), this optimization is relatively trivial in the case of a line search. Indeed, for a given segment, the value in Equation 5 is constant everywhere except where we intersect any of the coordinate hyperplanes, i.e., where one of the coordinates is zero. Thus, our method consists of evaluating Equation 5 at the intersection points between the line and coordinate hyperplanes, returning the optimal point within the given segment. For any segment that doesn't cross any of these hyperplanes, we evaluate the objective function at any point of the segment (since the value is constant across the entire segment).

4 Direction finding

4.1 A Gradient-based direction finder

Perhaps the greatest obstacle in scaling MERT to many dimensions is finding good search directions. In problems of lower dimensions, iterating through all the coordinates is computationally feasible, though not guaranteed to find a global maximum even in the case of a perfect line search. As the number of dimensions increases by orders of magnitude, this coordinate direction approach becomes less and less tractable, and the quality of the search also suffers (Hopkins and May, 2011).

Optimization has traditionally relied on finding the direction of steepest ascent: the gradient. Unfortunately, the objective function optimized by MERT is piecewise constant; while it may admit a subgradient, this direction is generally not very informative. Instead we may consider a smoothed variation of the original approximation. While some variants have been considered (Och, 2003; Flanigan et al., 2013), we use an expected BLEU approximation, assuming hypotheses are drawn from a log-linear distribution according to their parameter values (Smith and Eisner, 2006). That is, we assume the probability of a translation candidate $\mathbf{e}_{s,m}$ is proportional to $(\exp(\mathbf{w}^\intercal \mathbf{h}_{s,m}))^\mu$, where $\mathbf{w}$ are the parameters being optimized, $\mathbf{h}_{s,m}$ is the vector of the features for $\mathbf{e}_{s,m}$, and μ is a scaling parameter. As μ approaches infinity, the distribution places all its weight on the highest scoring candidate.

The log of the BLEU score may be written as:

$$\min\left(1-\frac{R}{C},0\right)+\frac{1}{N}\sum_{n=1}^{N}(\log m_n-\log c_n)$$

where R is the sum of reference lengths across the corpus, C is the sum of candidate lengths, m_n is the number of matched n-grams (potentially clipped), and c_n is the number of n-grams in all candidates.

Given a distribution over candidates, we can use the expected value of the log of the BLEU score. This is a smooth approximation to the BLEU score, which asymptotically approaches the true BLEU score as the scaling parameter μ approaches infinity. While this expectation is difficult to compute exactly, we can compute approximations thereof using Taylor series. Although prior work demonstrates that a second-order Taylor approximation is feasible to compute (Smith and Eisner, 2006), we find that a first-order approximation is faster and very close to the second-order approximation.[5] The first order Taylor approximation is as follows:

$$\min\left(1-\frac{R}{\mathbb{E}[C]},0\right)+\frac{1}{N}\sum_{n=1}^{N}(\log\mathbb{E}[m_n]-\log\mathbb{E}[c_n])$$

where $\mathbb{E}$ is the expectation operator using the probability distribution $P(\mathbf{h};\mathbf{w},\mu)$.

First we note that the gradient $\frac{\partial}{\partial w_i}P(\mathbf{h};\mathbf{w},\mu)$ is

$$P(\mathbf{h};\mathbf{w},\mu)\left(h_i-\sum_{\mathbf{h}'}h_i' P(\mathbf{h}';\mathbf{w},\mu)\right)$$

Using the chain rule, the gradient of the first order approximation to BLEU is as follows:

$$\frac{1}{N}\sum_{n=1}^{N}\Big(\frac{1}{\mathbb{E}[m_n]}\sum_{\mathbf{h}}m_n(\mathbf{h})\frac{\partial P(\mathbf{h};\mathbf{w},\mu)}{\partial w_i} - \frac{1}{\mathbb{E}[c_n]}\sum_{\mathbf{h}}c_n(\mathbf{h})\frac{\partial P(\mathbf{h};\mathbf{w},\mu)}{\partial w_i}\Big)$$

$$+\begin{cases}0 & \text{if } \mathbb{E}[C]>R\\ \frac{R}{\mathbb{E}[C]^2}\sum_{\mathbf{h}}c_1(\mathbf{h})\frac{\partial P(\mathbf{h};\mathbf{w},\mu)}{\partial w_i} & \text{otherwise}\end{cases}$$

[5] Experimentally, we compared our analytical gradient of the first-order Taylor approximation with the finite-difference gradients of the first- and second-order approximations, and we found these three gradients to be very close in terms of cosine similarity (> 0.99). We performed these measurements both at arbitrary points and at points of convergence of MERT.

In the case of ℓ_2-regularized MERT, the final gradient also includes the partial derivative of the regularization penalty of Equation 4, which is w_i/σ^2 for a given component i of the gradient. We do not update the gradient in the case of ℓ_0 regularization since the ℓ_0-norm is not differentiable.

4.2 Search

Our search strategy consists of looking at the directions of steepest increase of expected BLEU, which is similar to that of Smith and Eisner (2006), but with the difference that we do so in the context of MERT. We think this difference provides two benefits. First, while the smooth approximation of BLEU reduces the likelihood of remaining trapped in a local optimum, we avoid approximation error by retaining the original objective function. Second, the benefit of exact line searches in MERT is that there is no need to be concerned about step size, since step size in MERT line searches is guaranteed to be optimal with respect to the direction under consideration.

Finally, our gradient-based search algorithm operates as follows. Considering the current point $\mathbf{w}_t$, we compute the gradient $\mathbf{g}_t$ of the first order Taylor approximation at that point, using the current scaling parameter μ. (We initialize the search with $\mu = 0.01$.) We find the optimum along the line $\mathbf{w}_t + \gamma \cdot \mathbf{g}_t$. Whenever any given line search yields no improvement larger than a small tolerance threshold, we multiply μ by two and perform a new line search. The increase of this parameter μ corresponds to a cooling schedule (Smith and Eisner, 2006), which progressively sharpens the objective function to get a better estimate of BLEU as the search converges to an optimum. We repeatedly perform new line searches until μ exceeds 1000. The inability to improve the current optimum with a sharp approximation ($\mu > 1000$) doesn't mean line searches would fail with smaller values, so we find it helpful to repeat the above procedure until a full pass of updates of μ from 0.01 to 1000 yields no improvement.

4.3 Computational complexity

Computing the gradient increases the computational cost of MERT, though not its asymptotic complexity. The cost of a single exhaustive line search is

$$\mathcal{O}\left(SM(D + \log M + \log S)\right)$$

where S is the number of sentences, each with M possible translations, and D is the number of features. For each sentence, we first identify the model score as a linear function of the step size, requiring two dot products for an overall cost of $\mathcal{O}(SMD)$.[6] Next we construct the upper envelope for each sentence: first the equations are sorted in increasing order of slope, and then they are merged in linear time to form an envelope, with an overall cost of $\mathcal{O}(SM \log M)$. A linear pass through the envelope converts these into piecewise constant (or linear, or quadratic) representations of the (regularized) loss function. Finally the per-sentence envelopes are merged into a global representation of the loss along that direction. Our implementation successively merges adjacent pairs of piecewise smooth loss function representations until a single list remains. These $\log S$ passes lead to a merging runtime of $\mathcal{O}(SM \log S)$.

The time required to compute a gradient is proportional to $\mathcal{O}(SMD)$. For each sentence, we first gather the probability and its gradient, then use this to compute expected n-gram counts and matches as well as those gradients in time $\mathcal{O}(MD)$. A constant number of arithmetic operations suffice to compute the final expected loss value and its gradient. Therefore, computing the gradient does not increase the algorithmic complexity when compared to conventional approaches using coordinate ascent and random directions. Likewise the runtime of a single iteration is competitive with PRO, given that gradient finding is generally the most expensive part of convex optimization. Of course, it is difficult to compare overall runtime of convex optimization with that of MERT, as we know of no way to bound the number of gradient evaluations required for convergence with MERT. Therefore, we resort to empirical comparison later in the paper, and find that the two methods appear to have comparable runtime.

[6] In the special case where the difference between the prior direction and the current direction is sparse, we may update the individual linear functions in time proportional to the number of changed dimensions. Coordinate ascent in particular can update the linear functions in time $\mathcal{O}(SM)$: to the intercept of the equation for each translation, we may add the prior step size multiplied by the feature value in the prior coordinate, and the slope becomes the feature value in the new coordinate. However, this optimization does not appear to be widely adopted, likely because it does not lead to any speedup when random vectors, conjugate directions, or other non-sparse directions are used.

	Language pair	Train	Tune	Dev	Test
GBM	Chinese-English	0.99M	1,797 (mt02+03)	1,000	1,082 (mt05)
	Finnish-English	2.20M	11,935	2,001	4,855
SparseHRM	Chinese-English	3.51M	1,894 (mt05)	1,664 (mt06)	1,357 (mt08)
	Arabic-English	1.49M	1,663 (mt06)	1,360 (mt08)	1,313 (mt09)

Table 1: Datasets for the two experimental conditions.

5 Experimental Design

Following Hopkins and May (2011), our experimental setup utilizes both real and synthetic data. The motivation for using synthetic data is that it is a way of gauging the quality of optimization methods, since the data is constructed knowing the global optimum. Hopkins and May also note that the use of an objective function that is linear in some gold weight vector makes the search much simpler than in a real translation setting, and they suggest that a learner that performs poorly in such a simple scenario has little hope of succeeding in a more complex one.

The setup of our synthetic data experiment is almost the same as that performed by Hopkins and May (2011). We generate feature vectors of dimensionality ranging from 10 to 1000. These features are generated by drawing random numbers uniformly in the interval $[0, 500]$. This synthetic dataset consists of S=1000 source "sentences", and M=500 "translation" hypotheses for each sentence. A pseudo "BLEU" score is then computed for each hypothesis, by computing the dot product between a predefined gold weight vector $\mathbf{w}^*$ and each feature vector $\mathbf{h}_{s,m}$. By this linear construction, $\mathbf{w}^*$ is guaranteed to be a global optimum.[7] The pseudo-BLEU score is normalized for each M-best list, so that the translation with highest model score according to $\mathbf{w}^*$ has a BLEU score of 1, and so that the translation with lowest model score for the sentence gets a BLEU of zero. This normalization has no impact on search, but makes results more interpretable.

For our translation experiments, we use multi-stack phrase-based decoding (Koehn et al., 2007). We report results for two feature sets: non-linear features induced using Gradient Boosting Machines (Toutanova and Ahn, 2013) and sparse lexicalized reordering features (Cherry, 2013). We exploit these feature sets (GBM and SparseHRM, respectively) in two distinct experimental conditions, which we detail in the two next paragraphs. Both GBM and SparseHRM augment baseline features similar to Moses': relative frequency and lexicalized phrase translation scores for both translation directions; one or two language model features, depending on the language pair; distortion penalty; word and phrase count; six lexicalized reordering features. For both experimental conditions, phrase tables have maximum phrase length of 7 words on either side. In reference to Table 1, we used the training set (Train) for extracting phrase tables and language models; the Tune set for optimization with MERT or PRO; the Dev set for selecting hyperparameters of PRO and regularized MERT; and the Test set for reporting final results. In each experimental condition, we first trained weights for the base feature sets, and then decoded the Tune, Dev, and Test datasets, generating 500-best lists for each set. All results report reranking performance on these lists with different feature sets and optimization methods, based on lower-cased BLEU (Papineni et al., 2001).

The GBM feature set (Toutanova and Ahn, 2013) consists of about 230 features automatically induced using decision tree weak learners, which derive features using various word-level, phrase-level, and morphological attributes. For Chinese-English, the training corpus consists of approximately one million sentence pairs from the FBIS and Hong Kong portions of the LDC data for the NIST MT evaluation and the Tune and Test sets are from NIST competitions. A 4-gram language model was trained on the Xinhua portion of the English Gigaword corpus and on the target side of the bitext. For Finnish-English we used a dataset from a technical domain of software manuals. For this language pair we used two language models: one very large model trained on billions of words, and another language model trained from the target side of the parallel training set.

The SparseHRM set (Cherry, 2013) contains 3600 sparse reordering features. For each phrase, the features take the form of indicators describing its orientation in the derivation, and its lexical content in terms of word clusters or frequent words. For both Chinese-English and Arabic-English, systems are trained on data from the NIST 2012 MT evaluation. 4-gram

[7] The objective function remains piecewise constant, and the plateau containing $\mathbf{w}^*$ maps to the optimal *value* of the function.

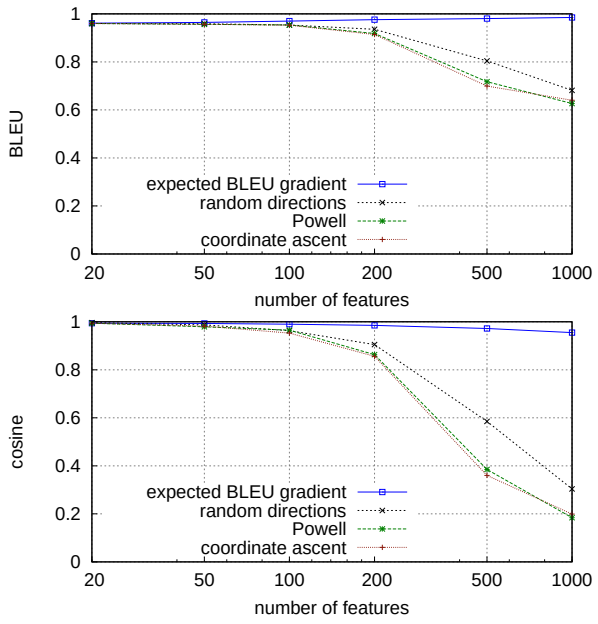


Figure 2: Change in BLEU score and cosine similarity to the gold weight vector $\mathbf{w}^*$ as the number of features increases, using the **noisy** synthetic experiments. The gradient-based direction finding method is barely affected by the noise. The increase of the number of dimensions enables our direction finder to find a slightly better optimum, which moved away from $\mathbf{w}^*$ due to noise.

language models were trained on the target side of the parallel training data for both Arabic and Chinese. The Chinese systems development set is taken from the NIST mt05 evaluation set, augmented with some material reserved from our NIST training corpora in order to better cover newsgroup and weblog domains.

6 Results

We conducted experiments with the synthetic data scenario described in the previous section, as well as with noise added to the data (Hopkins and May, 2011). The purpose of adding noise is to make the optimization task more realistic. Specifically, after computing all pseudo-BLEU scores, we added noise to each feature vector $\mathbf{h}_{s,m}$ by drawing from a zero-mean Gaussian with standard deviation 200. Our results with both noiseless and noisy data yield the same conclusion as Hopkins and May: standard MERT struggles with many dimensions, and fails to recover $\mathbf{w}^*$. However, our experiments with the gradient direction finder of Section 4 are much more positive. This direction finder not only recovers $\mathbf{w}^*$

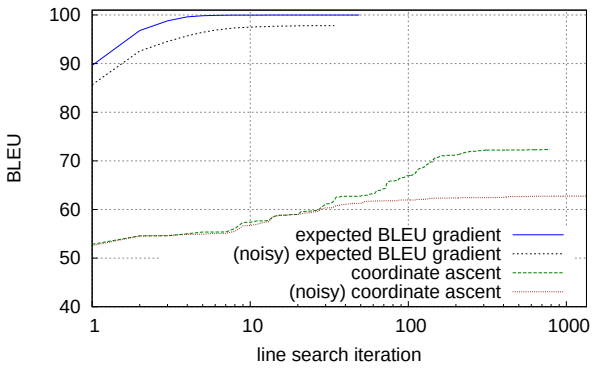


Figure 3: Comparison of rate of convergence between coordinate ascent and our expected BLEU direction finder ($D = 500$). **Noisy** refers to the noisy experimental setting.

(cosine > 0.999) even with 1000 dimensions, but its effectiveness is also visible with noisy data, as seen in Figure 2. The decrease of its cosine is relatively small compared to other search algorithms, and this decrease is not necessarily a sign of search errors since the addition of noise causes the true optimum to be different from $\mathbf{w}^*$. Finally, Figure 3 shows our rate of convergence compared to coordinate ascent.

Our experimental results with the GBM feature set data are shown in Table 2. Each table is divided into three sections corresponding respectively to MERT (Och, 2003) with Koehn-style coordinate ascent (Koehn, 2004), PRO, and our optimizer featuring both regularization and the gradient-based direction finder. All variants of MERT are initialized with a single starting point, which is either uniform weight or $\mathbf{w}_0$. Instead of providing MERT with additional random starting points as in Moses, we use random walks as in (Moore and Quirk, 2008) to attempt to move out of local optima.[8] Since PRO and our optimizer have hyperparameters, we use a held-out set (Dev) for adjusting them. For PRO, we adjust three parameters: a regularization penalty for ℓ_2, the parameter α in the add-α smoothed sentence-level version of BLEU (Lin and Och, 2004), and a parameter for scaling the corpus-level length of the references. The latter scaling parameter is discussed in (He and

[8] In the case of the gradient-based direction finder, we also use the following strategy whenever optimization converges to a (possibly local) optimum. We run one round of coordinate ascent, and continue with the gradient direction finder as soon as the optimum improves. If the none of the coordinate directions helped, we stop the search.

Method	Starting pt.	Chinese-English # feat.	Tune	Dev	Test	Finnish-English # feat.	Tune	Dev	Test
MERT	uniform	14	33.2	19.9	32.9	15	53.0	52.6	54.8
MERT	uniform	224	33.0	19.2	32.1	232	53.2	51.7	53.8
MERT	$\mathbf{w}_0$	224	34.1	20.1	33.0	232	53.9	52.5	54.7
PRO	$\mathbf{w}_0$	224	33.4	20.1	33.3	232	53.3	52.9	55.3
ℓ_2 MERT (v1: $\|\|\mathbf{w} - \mathbf{w}_0\|\|$)	$\mathbf{w}_0$	224	33.2	20.3	33.5	232	53.2	52.7	55.2
ℓ_2 MERT (v2: $D-1$ dimensions)	$\mathbf{w}_0$	224	33.0	20.4	33.2	232	52.9	52.6	55.0
ℓ_2 MERT (v3: ℓ_1-renormalized)	$\mathbf{w}_0$	224	33.1	20.0	33.3	232	53.1	52.5	55.1
ℓ_0 MERT	$\mathbf{w}_0$	224	33.4	20.3	33.2	232	53.2	52.6	55.1

Table 2: BLEU scores for GBM features. Model parameters were optimized on the Tune set. For PRO and regularized MERT, we optimized with different hyperparameters (regularization weight, etc.), and retained for each experimental condition the model that worked best on Dev. The table shows the performance of these retained models.

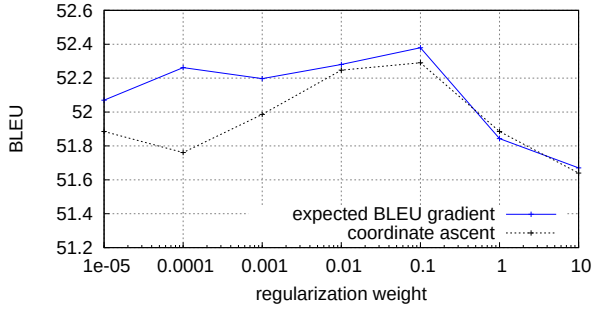


Figure 4: BLEU score on the Finnish Dev set (GBM) with different values for the $1/2\sigma^2$ regularization weight. To enable comparable results, the other hyperparameter (length) is kept fixed.

Deng, 2012; Nakov et al., 2012) and addresses the problem that systems tuned with PRO tend to produce sentences that are too short. On the other hand, regularized MERT only requires one hyperparameter to tune: a regularization penalty for ℓ_2 or ℓ_0. However, since PRO optimizes translation length on the Dev dataset and MERT does so using the Tune set, a comparison of the two systems would yield a discrepancy in length that would be undesirable. Therefore, we add another hyperparameter to regularized MERT to tune length in the same manner using the Dev set.

Table 2 offers several findings. First, unregularized MERT can achieve competitive results with a small set of highly engineered features, but adding a large set of more than 200 features causes MERT to perform poorly, particularly on the test set. However, unregularized MERT can recover much of this drop of performance if it is given a good sparse initializer $\mathbf{w}_0$. Regularized MERT (v1) provides an increase in the order of 0.5 BLEU on the test set compared to the best results with unregularized MERT. Regularized MERT is competitive with PRO, even though the number of features is relatively large. Using the same GBM experimental setting, Figure 4 compares regularized MERT using the gradient direction finder and coordinate ascent. At the best regularization setting, the two algorithms are comparable in terms of BLEU (though coordinate ascent is slower due to its lack of a good direction finder), but our method seems more robust with suboptimal regularization parameters.

Our results with the SparseHRM feature set data are shown in Table 3. As with the GBM feature set, we find again that the version of ℓ_2 MERT regularized towards $||\mathbf{w} - \mathbf{w}_0||$ is competitive with PRO, even though we train MERT with a large set of 3601 features.[9] One remaining question is whether MERT remains practical with large feature sets. As noted in the complexity analysis of Section 4.3, MERT has a dependence on the number of features that is comparable to PRO, i.e., it is linear in both cases. Practically, we find that optimization time is comparable between the two systems. In the case of Chinese-English for the GBM feature set, one run of the PRO optimizer took 26 minutes on average, while regularized MERT with the gradient direction finder took 37 minutes on average, taking into account the time to compute $\mathbf{w}_0$. In the case of Chinese-English for the SparseHRM feature set, average optimization times for PRO and our method were 3.10 hours and 3.84 hours on average, respectively.

[9] We note that the experimental setup of (Cherry, 2013) integrates the Sparse HRM features into the decoder, while we use them in an M-best reranking scenario. The reranking setup of this paper yields smaller improvements for both PRO and MERT than those of (Cherry, 2013).

Method	Starting pt.	Chinese-English # feat.	Tune	Dev	Test	Arabic-English # feat.	Tune	Dev	Test
MERT	uniform	14	25.7	34.0	27.8	14	43.2	42.8	45.5
MERT	uniform	3601	25.4	33.1	27.3	3601	45.7	42.3	44.9
MERT	$\mathbf{w}_0$	3601	27.7	33.5	27.5	3601	46.0	42.4	45.2
PRO	$\mathbf{w}_0$	3601	25.9	34.3	28.1	3601	44.6	43.4	46.1
ℓ_2 MERT (v1: $\|\|\mathbf{w} - \mathbf{w}_0\|\|$)	$\mathbf{w}_0$	3601	26.3	34.3	28.3	3601	45.2	43.2	46.0
ℓ_2 MERT (v2: $D-1$ dimensions)	$\mathbf{w}_0$	3601	26.4	34.1	28.2	3601	45.0	43.4	45.9
ℓ_2 MERT (v3: ℓ_1-renormalized)	$\mathbf{w}_0$	3601	26.1	34.0	27.9	3601	44.9	43.3	45.7
ℓ_0 MERT	$\mathbf{w}_0$	3601	26.5	34.2	28.1	3601	45.4	43.1	46.0

Table 3: BLEU scores for SparseHRM features. Notes in Table 2 also apply here.

Finally, as shown in Table 2, we see that MERT experiments that rely on a good initial starting point $\mathbf{w}_0$ generally perform better than when starting from a uniform vector. While having to compute $\mathbf{w}_0$ in the first place is a bit of a disadvantage compared to standard MERT, the need for good initializer is hardly surprising in the context of non-convex optimization. Other non-convex problems in machine learning, such as deep neural networks (DNN) and word alignment models, commonly require such initializers in order to obtain decent performance. In the case of DNN, extensive research is devoted to the problem of finding good initializers.[10] In the case of word alignment, it is common practice to initialize search in non-convex optimization problems—such as IBM Model 3 and 4 (Brown et al., 1993)—with solutions of simpler models—such as IBM Model 1.

7 Related work

MERT and its extensions have been the target of extensive research (Och, 2003; Macherey et al., 2008; Cer et al., 2008; Moore and Quirk, 2008; Kumar et al., 2009; Galley and Quirk, 2011). More recent work has focused on replacing MERT with a linearly decomposable approximations of the evaluation metric (Smith and Eisner, 2006; Liang et al., 2006; Watanabe et al., 2007; Chiang et al., 2008; Hopkins and May, 2011; Rosti et al., 2011; Gimpel and Smith, 2012; Cherry and Foster, 2012), which generally involve a surrogate loss function incorporating a regularization term such as the ℓ_2-norm. While we are not aware of any previous work adding a penalty on the weights in the context of MERT, (Cer et al., 2008) achieves a related effect. Cer et al.'s goal is to achieve a more regular or smooth objective function, while ours is to obtain a more regular set of parameters. The two approaches may be complementary.

More recently, new research has explored direction finding using a smooth surrogate loss function (Flanigan et al., 2013). Although this method is successful in helping MERT find better directions, it also exacerbates the tendency of MERT to overfit.[11] As an indirect way of controlling overfitting on the tuning set, their line searches are performed over directions estimated over a separate dataset.

8 Conclusion

In this paper, we have shown that MERT can scale to a much larger number of features than previously thought, thanks to regularization and a direction finder that directs the search towards the greatest increase of expected BLEU score. While our best results are comparable to PRO and not significantly better, we think that this paper provides a deeper understanding of why standard MERT can fail when handling an increasingly larger number of features. Furthermore, this paper complements the analysis by Hopkins and May (2011) of the differences between MERT and optimization with a surrogate loss function.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions.

[10] For example, (Larochelle et al., 2009) presents a pre-trained DNN that outperforms a shallow network, but the performance of the DNN becomes much worse relative to the shallow network once pre-training is turned off.

[11] Indeed, in their Table 3, a comparison between HILS and HOLS suggests tuning set performance improves substantially, while held out performance degrades.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Daniel Cer, Dan Jurafsky, and Christopher D. Manning. 2008. Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226.

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

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 248–258.

Michel Galley and Chris Quirk. 2011. Optimal search for minimum error rate training. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 38–49.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Xiaodong He and Li Deng. 2012. Maximum expected BLEU training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 292–301.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362.

M. Hyder and K. Mahata. 2009. An approximate L0 norm minimization algorithm for compressed sensing. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3365–3368.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Demonstration Session*.

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

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.*, 10:1–40.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, Stroudsburg, PA, USA.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734.

David McAllester and Joseph Keshet. 2011. Generalization bounds and consistency for latent structural probit and ramp loss. In *Advances in Neural Information Processing Systems 24*, pages 2205–2212.

Robert C. Moore and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, pages 585–592.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. In *Proceedings of COLING 2012*, pages 1979–1994.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Kishore Papineni. 1999. Discriminative training via linear programming. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 561–564, Vol. 2.

M.J.D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7(2):155–162.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition.

Antti-Veikko Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2011. Expected BLEU training for graphs: BBN system description for WMT11 system combination task. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 159–165.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794.

Kristina Toutanova and Byung-Gyu Ahn. 2013. Learning non-linear features for machine translation using gradient boosting machines. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 406–411.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773.

Of words, eyes and brains: Correlating image-based distributional semantic models with neural representations of concepts

Andrew J. Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio and Marco Baroni
Center for Mind/Brain Sciences (University of Trento, C.so Bettini 31, Rovereto, Italy)
first.last@unitn.it

Abstract

Traditional distributional semantic models extract word meaning representations from co-occurrence patterns of words in text corpora. Recently, the distributional approach has been extended to models that record the co-occurrence of words with visual features in image collections. These image-based models should be complementary to text-based ones, providing a more cognitively plausible view of meaning grounded in visual perception. In this study, we test whether image-based models capture the semantic patterns that emerge from fMRI recordings of the neural signal. Our results indicate that, indeed, there is a significant correlation between image-based and brain-based semantic similarities, and that image-based models complement text-based ones, so that the best correlations are achieved when the two modalities are combined. Despite some unsatisfactory, but explained outcomes (in particular, failure to detect differential association of models with brain areas), the results show, on the one hand, that image-based distributional semantic models can be a precious new tool to explore semantic representation in the brain, and, on the other, that neural data can be used as the ultimate test set to validate artificial semantic models in terms of their cognitive plausibility.

1 Introduction

Many recent neuroscientific studies have brought support to the view that concepts are represented in terms of patterns of neural activation over broad areas, naturally encoded as vectors in a neural semantic space (Haxby et al., 2001; Huth et al., 2012). Similar representations are also widely used in computational linguistics, and in particular in distributional semantics (Clark, 2012; Erk, 2012; Turney and Pantel, 2010), that captures meaning in terms of vectors recording the patterns of co-occurrence of words in large corpora, under the hypothesis that words that occur in similar contexts are similar in meaning.

Since the seminal work of Mitchell et al. (2008), there has thus being interest in investigating whether corpus-harvested semantic representations can contribute to the study of concepts in the brain. The relation is mutually beneficial: From the point of view of brain activity decoding, a strong correlation between corpus-based and brain-derived conceptual representations would mean that we could use the former (much easier to construct on a very large scale) to make inferences about the second: e.g., using corpus-based representations to reconstruct the likely neural signal associated to words we have no direct brain data for. From the point of view of computational linguistics, neural data provide the ultimate testing ground for models that strive to capture important aspects of human semantic memory (much more so than the commonly used explicit semantic rating benchmarks). If we found that a corpus-based model of meaning can make non-trivial predictions about the structure of the semantic space in the brain, that would make a pretty strong case for the intriguing idea that the model is approximating, in interesting ways, the way in which humans acquire and represent semantic knowledge.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1960–1970,
Seattle, Washington, USA, 18-21 October 2013.

We take as our starting point the extensive experiments reported in Murphy et al. (2012), who showed that purely corpus-based distributional models are at least as good at brain signal prediction tasks as earlier models that made use of manually-generated or controlled knowledge sources (Chang et al., 2011; Palatucci et al., 2009; Pereira et al., 2011), and we evaluate a very recent type of distributional model, namely one that is not extracted from textual data but from *image collections* through automated visual feature extraction techniques. It has been argued that this new generation of image-based distributional models (Bruni et al., 2011; Bruni et al., 2012b; Feng and Lapata, 2010; Leong and Mihalcea, 2011) provides a more realistic view of meaning, since humans obviously acquire a large proportion of their semantic knowledge from perceptual data. The first question that we ask, thus, is whether the more "grounded" image-based models can help us in interpreting conceptual representations in the brain. More specifically, we will compare the performance of different image-based representations, and we will test whether text- and image-based representations are complementary, so that when used together they can better account for patterns in neural data. Finally, we will check for differences between anatomical regions in the degree to which text and/or image models are effective, as one might expect given the well-known functional specializations of different anatomical regions.

2 Brain data

We use the data that were recorded and preprocessed by Mitchell et al. (2008), available for download in their supporting online material.[1] Full details of the experimental protocol, data acquisition and preprocessing can be found in Mitchell et al. (2008) and the supporting material. Key points are that there were nine right-handed adult participants (5 female, age between 18 and 32). The experimental task was to actively think about the properties of sixty objects that were presented visually, each as a line drawing in combination with a text label. The entire set of objects was presented in a random order in six sessions, each object remained on screen for 3 seconds with a seven second fixation gap between presentations.

Mitchell and colleagues examined 12 categories, five objects per category, for a total of 60 concepts (words). Due to coverage limitations, we use 51/60 words representing 11/12 categories. Table 1 contains the full list of 51 words organized by category.

fMRI acquisition and preprocessing Mitchell et al. (2008) acquired functional images on a Siemens Allegra 3.0T scanner using a gradient echo EPI pulse sequence with TR=1000 ms, TE=30 ms and a 60° angle. Seventeen 5-mm thick oblique-axial slices were imaged with a gap of 1-mm between slices. The acquisition matrix was 64×64 with 3.125×3.125×5-mm voxels. They subsequently corrected data for slice timing, motion, linear trend, and performed temporal smoothing with a high-pass filter at 190s cutoff. The data were normalized to the MNI template brain image, spatially normalized into MNI space and resampled to $3\times3\times6$ mm^3 voxels. The voxel-wise percent signal change relative to the fixation condition was computed for each object presentation. The mean of the four images acquired 4s post stimulus presentation was used for analysis.

To create a single representation per object per participant, we took the voxel-wise mean of the six presentations of each word. Likewise to create a single representation per category per participant, we took the voxel-wise mean of all word models per category, per participant.

Anatomical parcellation Analysis was conducted on the whole brain, and to address the question of whether there are differences in models' effectiveness between anatomical regions, brains were further partitioned into frontal, parietal, temporal and occipital lobes. This partitioning is coarse (each lobe is large and serves many diverse functions), but, for an initial test, appropriate, given that each lobe has specialisms that on face value are amenable to interpretation by our different distributional models and the exact nature of specialist processing in localised areas is often subject to debate (so being overly restrictive may be risky). Formulation of the distributional models is described in detail in the Section 3, but for now it is sufficient to know that the Object model is derived from image statistics of the object depicted in images, Context from image statistics of the background scene, Object&Context is a com-

[1] http://www.cs.cmu.edu/~tom/science2008/

Animals	Bear, Cat, Cow, Dog Horse
Building	Apartment, Barn, Church, House
Building parts	Arch, Chimney, Closet, Door, Window
Clothing	Coat, Dress, Pants, Shirt, Skirt
Furniture	Bed, Chair, Desk, Dresser, Table
Insect	Ant, Bee, Beetle, Butterfly, Fly
Kitchen utensils	Bottle, Cup, Glass, Knife, Spoon
Man made objects	Bell, Key, Refrigerator, Telephone, Watch
Tool	Chisel, Hammer, Screwdriver
Vegetable	Celery, Corn, Lettuce, Tomato
Vehicle	Airplane, Bicycle, Car, Train, Truck

Table 1: The 51 words represented by the brain and the distributional models, organized by category.

bination of the two, and Window2 is a text-based model.

The *occipital* lobe houses the primary visual processing system and consequently it is reasonable to expect some bias toward image-based semantic models. Furthermore, given that experimental stimuli incorporated line drawings of the object,and the visual cortex has a well-established role in processing low-level visual statistics including edge detection (Bruce et al., 2003), we naturally expected a good performance from Object (formulated from edge orientation histograms of similar objects).

Following Goodale and Milner (1992)'s influential perception-action model (see McIntosh and Schenk (2009) for recent discussion), visual information is channeled from the occipital lobe in two streams: a perceptual stream, serving object identification and recognition; and an action stream, specialist in processing egocentric spatial relationships and ultimately supporting interaction with the world.

The perceptual stream leads to the *temporal lobe*. Here the fusiform gyrus (shared with the occipital lobe) plays a general role in object categorisation (e.g., animals and tools (Chao et al., 1999), faces (Kanwisher and Yovel, 2006), body parts (Peelen and Downing, 2005) and even word form perception (McCandliss et al., 2003)). As the parahippocampus is strongly associated with scene representation (Epstein, 2008), we expect both the Object and Context models to capture variability in the temporal lobe. Of wider relevance to semantic processing, the medial temporal gyrus, inferior temporal gyrus and ventral temporal lobe have generally been implicated to have roles in supramodal integration and concept retrieval (Binder et al., 2009). Given this, we expected that incorporating text would also be valuable and that the Window2&Object&Context combination would be a good model.

The visual action stream leads from the occipital lobe to the *parietal* lobe to support spatial cognition tasks and action control (Sack, 2009). In that there seems to be an egocentric frame of reference, placing actor in environment, it is tempting to speculate that the Context model is more appropriate than the Object model here. As the parietal lobe also contains the angular gyrus, thought to be involved in complex, supra-modal information integration and knowledge retrieval (Binder et al., 2009), we might again forecast that integrating text and image information would boost performance, so Window2&Context was earmarked as a strong candidate.

The *frontal lobe*, is traditionally associated with high-level processing and manipulation of abstract knowledge and rules and controlled behaviour (Miller et al., 2002). Regarding semantics, the dorsomedial prefrontal cortex has been implicated in self-guided retrieval of semantic information (e.g., uncued speech production), the ventromedial prefrontal cortex in motivation and emotional processing, the inferior frontal gyrus in phonological and syntactic processing, (Binder et al., 2009) and integration of lexical information (Hagoort, 2005). Given the association with linguistic processing we anticipated a bias in favour of Window2.

The four lobes were identified and partitioned using Tzourio-Mazoyer et al. (2002)'s automatic anatomical labelling scheme.

Voxel selection The set of 500 most stable voxels, both within the whole brain and from within each region of interest were identified for analysis. The most stable voxels were those showing consistent variation across the different stimuli between scanning sessions. Specifically, and following a similar strategy to Mitchell et al. (2008), for each voxel, the set of 51 words from each unique pair of scanning sessions were correlated using Pearson's correlation (6 sessions and therefore 15 unique pairs), and the mean of the 15 resulting correlation coefficients was taken as the measure of stability. The 500 voxels with highest mean correlations were selected.

3 Distributional models

Distributional semantic models approximate word meaning by keeping track of word co-occurrence statistics from large textual input, relying on the distributional hypothesis: The meaning of a word can be induced by the context in which it occurs (Turney and Pantel, 2010). Despite their great success, these models still rely on verbal input only, while humans base their meaning representation also on perceptual information (Louwerse, 2011).

Thanks to recent developments in computer vision, it is nowadays possible to take the visual perceptual channel into account, and build new computational models of semantics enhanced with visual information (Feng and Lapata, 2010; Bruni et al., 2011; Leong and Mihalcea, 2011; Bergsma and Goebel, 2011; Bruni et al., 2012a). Given a set of target concepts and a collection of images depicting those concepts, it is indeed possible to first encode the image content into low-level features, and subsequently convert it into a higher-level representation based on the bag-of-visual-words method (Grauman and Leibe, 2011). Recently, Bruni et al. (2012b) have shown that better semantic representations can be extracted if we first localize the concept in the image, and then extract distinct higher-level features (visual words) from the box containing the concept and from the surrounding context. We also follow this strategy here.

In our experiments we utilize both traditional text-based models and experimental image-based models, as well as their combination.

3.1 Textual models

Verb We experiment with the original text-based semantic model used to predict fMRI patterns by Mitchell et al. (2008). Each object stimulus word is represented as a 25-dimensional vector, with each value corresponding to the normalized sentence-wide co-occurrence of that word with one of 25 manually-picked sensorimotor verbs (such as *see*, *hear*, *eat*, ...) in a trillion word text corpus.

Window2 To create this model, we collect text co-occurrence statistics from the freely available ukWaC and Wackypedia corpora combined (about 3 billion words in total).[2] As collocates of our distributional model we select a set of 30K words, namely the top 20K most frequent nouns, 5K most frequent adjectives and 5K most frequent verbs.

In the tradition of HAL (Lund and Burgess, 1996), the model is based on co-occurrence statistics with collocates within a fixed-size window of 2 to the left and right of each target word. Despite their simplicity, narrow-window-based models have shown to achieve state-of-the-art results in various standard semantic tasks (Bullinaria and Levy, 2007) and to outperform both document-based and syntax-based models trained on the same corpus (Bruni et al., 2012a). Moreover, in Murphy et al. (2012) a window-based model very similar to ours was not significantly worse than their best model for brain decoding. We tried also a few variations, e.g., using a larger window or different transformations on the raw co-occurrences from those presented below, but with little, insignificant changes in performance. Given that our focus here is on visual information, we only report results for Window2 and its combination with visual models.

3.2 Visual models

Our visual models are inspired by Bruni et al. (2012b), that have explored to what extent extracting features from images where objects are localized results in better semantic representations. They found that extracting visual features separately from the object and its surrounding context leads to better performance than not using localization, and using only object- and, more surprisingly, context-extracted features also results in performant models

[2] http://wacky.sslmit.unibo.it/

(especially when evaluating inter-object similarity, the context in which an object is located can significantly contribute to semantic representation, in certain cases carrying even more information than the depicted object itself).

More in detail, with localization the visual features (visual words) can be extracted from the object bounding box (in our experiments, the **Object** model) or from only outside the object box (the **Context** model). A combined model is obtained by concatenating the two feature vectors (the **Object&Context** model).

Visual model construction pipeline To extract visual co-occurrence statistics, we use images from ImageNet (Deng et al., 2009),[3] a very large image database organized on top of the WordNet hierarchy (Fellbaum, 1998). ImageNet has more than 14 million images, covering 21K WordNet nominal synsets. ImageNet stands out for the high quality of its images, both in terms of resolution and concept annotations. Moreover, for around 3K concepts, annotations of object bounding boxes is provided. This last feature allows us to exploit object localization within our experiments.

To build visual distributional models, we utilize the bag-of-visual-words (BoVW) representation of images (Sivic and Zisserman, 2003; Csurka et al., 2004). Inspired by NLP, BoVW discretizes the image content in terms of a histogram of visual word counts. Differently from NLP, in vision there is not a natural notion of visual words, hence a visual vocabulary has to be built from scratch. The process works as follows. First, a large set of low-level features is extracted from a corpus of images. The low-level feature vectors are subsequently clustered into different regions (visual words). Given then a new image, each of the low-level feature vectors extracted from the patches that compose it is mapped to the nearest visual word (e.g., in terms of Euclidean distance from the cluster centroid) such that the image can be represented with a histogram counting the instances of each visual word in the image.

As low-level features we use SIFT, the Scale Invariant Feature Transform (Lowe, 2004). SIFT features are good at capturing parts of objects and are designed to be invariant to image transformations such as change in scale, rotation and illumination. To construct the visual vocabulary, we cluster the SIFT features into 25K different clusters.[4] We add also spatial information by dividing the image into several subregions, representing each of them in terms of BoVW and then stacking the resulting histograms (Lazebnik et al., 2006). We use in total 8 different regions, obtaining a final vector of 200K dimensions (25K visual words $\times$ 8 regions). Since each concept in our dataset is represented by multiple images, we pool the visual word occurrences across images by summing them up into a single vector.

To perform the entire visual pipeline we use VSEM, an open library for visual semantics (Bruni et al., 2013).[5]

3.3 Model transformations and combination

Once both the textual and the visual models are built, we perform two different transformations on the raw co-occurrence counts. First, we transform them into nonnegative Pointwise Mutual Information (PMI) association scores (Church and Hanks, 1990). As a second transformation, we apply dimensionality reduction to the two matrices. In particular, we adopt the Singular Value Decomposition (SVD), one of the most effective methods to approximate the original data in lower dimensionality space (Schütze, 1997), and reduce the vectors to 50 dimensions.

To combine text- and image-based semantic models in a joint representation, we separately normalize their vectors to unit length, and concatenate them, along the lines of Bruni et al. (2011). More sophisticated combination models have been proposed in the recent literature on multimodal semantics. For example, Bruni et al. (2012a) use SVD as a mixing strategy, given its ability to smooth the matrices and uncover latent dimensions. Another example is Silberer and Lapata (2013), where Canonical Correlation Analysis is used. We reserve the exploration of more advanced combination methods for further studies.

Finally, to represent the 11 categories we experiment with (see Table 1), we average the vectors of the concepts they include.

[3] http://www.image-net.org/

[4] We use *k*-means, the most commonly employed clustering algorithm for this task.

[5] http://clic.cimec.unitn.it/vsem/

4 Experiments

A question is posed over how to evaluate the relationship between the different distributional models and brain data. Comparing each model's predictive performance using the same strategy as Mitchell et al. (2008) (also followed by Murphy et al. (2012)) is one possibility: they used multiple regression to relate distributional codes to individual voxel activations, thus allowing brain states to be estimated from previously unseen distributional codes. Regression models were trained on 58/60 words and in testing the regression models estimated the brain state associated with the 2 unseen distributional codes. The predicted brain states were compared with the actual fMRI data, and the process repeated for each permutation of left-out words, to build a metric of prediction accuracy. For our purposes, a fair comparison of models using this strategy is complicated by differences in dimensionality between both semantic models and lobes (which we compare to other lobes) in association with the comparatively small number of words in the fMRI data set. Large dimensionality models risk overfitting the data, and it is a nuisance to try to reliably correct for the effects of overfitting in performance comparisons. Not least, to thoroughly evaluate all possible cross-validation permutations is demanding in processing time, and we have many models to compare.

An alternative approach, and that which we have adopted, is representational similarity analysis (Kriegeskorte et al., 2008). Representational similarity analysis circumvents the previous problems by abstracting each fMRI/distributional data source to a common structure capturing the interrelationships between each pair of data items (e.g., words). Specifically, for each model/participant's fMRI data/anatomical region, the similarity structure was evaluated by taking the pairwise correlation (Pearson's correlation coefficient) between all unique category or word combinations. This produced a list of 55 category pair correlations and 121 word pair correlations for each data source. For all brain data, correlation lists were averaged across the nine participants to produce a single list of mean word pair correlations and a single list of mean category pair correlations for each anatomical region and the whole brain. Then to provide a measure of similarity between models and brain data, the correlation lists for respective data sources were themselves correlated using Spearman's rank correlation. Statistical significance was tested using a permutation test: The word-pair (or category-pair) labels were randomly shuffled 10,000 times to estimate a null distribution when the two similarity lists are not correlated. The p-value is calculated as the proportion of random correlation coefficients that are greater than or equal to the observed coefficient.

5 Results

5.1 Category-level analyses

Do image models correlate with brain data? Table 2 displays results of Spearman's correlations between the per-category similarity structure of distributional models and brain data. There is a significant correlation between every purely image-based model and the occipital, parietal and temporal lobes, and also the whole brain ($.38 \leq \rho \geq .51$, all $p \leq .01$). The frontal lobe is less well described. Still, whilst not significant, correlations are only marginally above the conventional $p = .05$ cutoff (all are less than $p = .064$). This strongly suggests that the answer to our first question is yes: *distributional models derived from images can be used to explain concept fMRI data*. Otherwise Window2 significantly correlates with the whole brain and all anatomical regions except for the frontal lobe where ρ=.34, $p = .07$. In contrast Verb (the original, partially hand-crafted model used by Mitchell and colleagues) captures inter-relationships poorly and neither correlates with the whole brain or any lobe.

Do different models correlate with different anatomical regions? 2-way ANOVA without replication was used to test for differences in correlation coefficients between the five pure-modality models (Verb, Window2, Object, Context and Object&Context), and the four brain lobes. This revealed a highly significant difference between models $F(4,12)=45.2$, $p<.001$. Post-hoc 2-tailed t-tests comparing model pairs found that Verb differed significantly from all other models (correlations were lower). There was a clear difference even when Verb (mean$\pm$sd over lobes = .1$\pm$.1) was compared to the second weakest model, Object (mean$\pm$sd=.4$\pm$.09), where t =-7.7, p <.01, df=4. There were no

	Frontal	Parietal	Occipital	Temporal	Whole-Brain
Verb	0.00 (0.51)	0.06 (0.37)	0.24 (0.10)	0.07 (0.35)	0.17 (0.17)
Window2	0.34 (0.06)	0.49 (0.00)	0.47 (0.01)	0.47 (0.00)	0.44 (0.00)
Object	0.27 (0.07)	0.38 (0.02)	0.45 (0.00)	0.47 (0.00)	0.43 (0.01)
Context	0.33 (0.06)	0.50 (0.00)	0.44 (0.00)	0.44 (0.01)	0.44 (0.01)
Object&Context	0.32 (0.05)	0.48 (0.00)	0.51 (0.00)	0.49 (0.00)	0.49 (0.00)
Window2&Object	0.32 (0.06)	0.45 (0.00)	0.52 (0.00)	0.53 (0.00)	0.49 (0.00)
Window2&Context	**0.39 (0.04)**	**0.57 (0.00)**	0.53 (0.00)	**0.55 (0.00)**	0.51 (0.00)
Window2&Object&Context	0.37 (0.04)	0.52 (0.00)	**0.55 (0.00)**	**0.55 (0.00)**	**0.53 (0.00)**

Table 2: Matrix of correlations between each pairwise combination of distributional semantic models and brain data. Correlations correspond to the pairwise similarity between the 11 categories. In each column the first value corresponds to Spearman's rank correlation coefficient and the value in parenthesis is the p-value.

	Frontal	Parietal	Occipital	Temporal	Whole-Brain
Verb	-0.04 (0.72)	0.09 (0.06)	0.07 (0.20)	0.03 (0.31)	0.07 (0.18)
Window2	0.07 (0.13)	0.19 (0.00)	0.12 (0.06)	0.21 (0.00)	0.13 (0.04)
Object	0.01 (0.40)	0.08 (0.07)	**0.17 (0.01)**	0.18 (0.00)	0.17 (0.01)
Context	0.04 (0.24)	0.14 (0.01)	0.01 (0.44)	0.12 (0.02)	0.02 (0.38)
Object&Context	0.03 (0.31)	0.13 (0.01)	0.10 (0.07)	0.17 (0.00)	0.11 (0.06)
Window2&Object	0.04 (0.24)	0.16 (0.00)	0.16 (0.01)	**0.23 (0.00)**	**0.17 (0.00)**
Window2&Context	**0.07 (0.12)**	**0.20 (0.00)**	0.09 (0.11)	0.22 (0.00)	0.11 (0.07)
Window2&Object&Context	0.05 (0.18)	0.18 (0.00)	0.12 (0.05)	**0.23 (0.00)**	0.13 (0.02)

Table 3: Matrix of correlations between each pairwise combination of distributional semantic models and brain data. Correlations correspond to the pairwise similarity between the 51 words. In each column the first value corresponds to Spearman's rank correlation coefficient and the value in parenthesis is the p-value.

other significant differences between models. However there was a highly significant difference between lobes F(3,12)=13.77, p <.001. Post-hoc 2-tailed t-tests comparing lobe pairs found that the frontal lobe yielded significantly different correlations (lower) than each other lobe. When the frontal lobe (mean±sd over models = .25±.14) was compared to the second weakest anatomical region, the parietal lobe (mean±sd=.38±.19), the difference was highly significant, t =-8, df=3, p <.01. This introduces the question of whether this difference in correlations is the result of differences in neural category organisation and representation, or differences in the quality of the signal, which we address next.

Category-level inter-correlations between lobes were all relatively strong and highly significant. The occipital lobe was found to be the most distinct, being similar to the temporal lobe (ρ=.71, p <.001), but less so to the parietal and frontal lobes (ρ=.53, p <.001 and ρ=.57, p <.001 respectively). The temporal lobe shows roughly similar levels of correlation to each other lobe (all .71≤ ρ ≥.73, all p <.001). The frontal and parietal lobes are related most strongly to each other (ρ=.77, p <.001), to a slightly lesser extent to the temporal lobe (in both cases ρ=.73, p <.001) and least so to the occipital lobe. These strong relationships are consistent with there being a broadly similar category organisation across lobes.

To appraise this assertion in the context of the previously detected difference between the frontal lobe and all other lobes, we examine the raw category pair similarity matrices derived from the occipital lobe and the frontal lobe (Figure 1). All the below observations are qualitative. Although it is difficult to have intuitions about the relative differences between all category pairs (e.g., whether tools or furniture should be more similar to animals), we might reasonably expect some obvious similarities. For instance, for animals to be visually similar to in-

sects and clothing, because all have legs and arms and curves (of course we would not expect a strong relationship between insects and clothes in function or other modalities such as sound), buildings to be similar to building parts and vehicles (hard edges and windows), building parts to be similar to furniture (e.g., from Table 1 we see there is some overlap in category membership between these categories, such as closet and door) and tools to be similar to kitchen utensils. All of these relationships are maintained in the occipital lobe, and many are visible in the frontal lobe (including the similarity between insects and clothes), however there are exceptions that are difficult to explain e.g., within the frontal lobe, building parts are not similar to furniture, kitchen utensils are closer to clothing than to tools and vehicles are more similar to clothing than anything else. As such we conclude that *category-level representations were similar across lobes* with differences likely due to variation in signal quality between lobes.

Are text- and image-based semantic models complementary? Turning to the question of whether text- and image-derived semantic information can be complementary, we observe from Table 2 that there is not a single instance of a joint model with a weaker correlation than its pure-image counterpart. The Window2 model showed a stronger correlation than the Window2&Object model for the frontal and parietal lobes, but was weaker than Window2&Object&Context and Window2&Context in all tests and was also weaker than any joint model in whole-brain comparisons. The mean$\pm$sd correlations for all purely image-based results pooled over lobes (3 models * 4 lobes) was .42$\pm$.08 in comparison to .49$\pm$.08 for the joint models. The relative performance of Object vs. Context vs. Object&Context on the four different lobes is preserved between image-based and joint models: correlating the 12 combinations using Spearman's correlation gives ρ=.85, p <.001. Differences can be statistically quantified by pooling all image related correlation coefficients for each anatomical region (3 models * 4 regions), as for the respective joint models, and comparing with a 2-tailed Wilcoxon signed rank test. Differences were highly significant (W=0, p <.001,n=12). This evidence accumulates to suggest that *text and image-derived semantic information can be complementary* in interpreting concept fMRI data.

Figure 1: Similarity (Pearson correlation) between each category pair in (top) occipital and (bottom) frontal lobes.

5.2 Word-level analyses

Do image models capture word pair similarities? Per-word results generally corroborate the relationships observed in the previous section in the sense that Spearman's correlation between per-word and per-category results for the 40 combinations of models and lobes was ρ=.78, p <.001. There were differences, most obviously a dramatic drop in the strength of correlation coefficients for the per-word results, visible in Table 3. *Subsets*

of per-word image-based models correlated with three lobes and the whole brain. Correlations corresponding to significance values of $p < .05$ were observed in the temporal and parietal lobes, for Context, Object&Context and Window2 whereas Object was correlated with the occipital and temporal lobes ($p < .05$). 2-way ANOVA without replication was used to test for differences between models and lobes. This revealed a significant difference between models (F(4,12)=4.05, $p = .027$). Post-hoc t-tests showed that the Window2 model significantly differed from (was stronger than) the Context (t=3.8, $p = .03$, df=3) and Object&Context models ($t = 4.5$, $p = .02$, df=3). There were no other significant differences between models. There was again a significant difference between lobes (F(3,12)=7.89, $p < .01$), with the frontal lobe showing the weakest correlations. Post-hoc 2-tailed t-tests comparing lobe-pairs found that the frontal lobe differed significantly (correlations were weaker) from the parietal ($t = -9$, $p < .001$, df=4) and temporal lobes ($t = -6.4$, $p < .01$, df=4) but not from the occipital lobe ($t = -2.18$, $p = .09$, df=4). No other significant differences between lobes were observed.

Are there differences between models/lobes? Word-level inter-correlations between lobes were all significant and the pattern of differences in correlation strength largely resembled that of the category-level analyses. The occipital lobe was again most similar to the temporal lobe (ρ=.57, $p < .001$), but less so to the parietal and frontal lobes (ρ=.47, $p < .001$ and ρ=.34, $p < .001$ respectively). The temporal lobe this time showed stronger correlation to the parietal (ρ=.68, $p < .001$) and frontal lobes (ρ=.61, $p < .001$) than the occipital lobe. The frontal and parietal lobes were again strongly related to one another (ρ=.67, $p < .001$). These results echo the category-level findings, that *word-level brain activity is also organised in a similar way across lobes*. Consequently this diminishes our chances of uncovering neat interactions between models and brain areas (where for instance the Window2 model correlates with the frontal lobe and Object model matches the occipital lobe). It is however noteworthy that we can observe some interpretable selectivity in lobe*model combinations. In particular the Context model better matches the parietal lobe than the Object model, which in turn better captures the occipital and temporal lobes (Observations are qualitative). Also as we see next, adding text information boosts performance in both parietal and temporal lobes (see Section 2 on our expectations about information encoded in the lobes).

Does joining text and image models help word-level interpretation? As concerns the benefits of joining Text and Image information, *per-word joint models were generally stronger than the respective image-based models*. There was one exception: adding text to the Object model weakened correlation with the occipital lobe. Joint models were exclusively stronger than Window2 for the temporal and occipital lobes, and were stronger in 1/3 of cases for the frontal and parietal lobes. In an analogous comparison to the per-category analysis, a Wilcoxon signed rank test was used to examine the difference made by adding text information to image models (pooling 3 models over 4 anatomical areas for both image and joint models). The mean$\pm$sd of image models was .1$\pm$.06 whereas for Joint models it was .15$\pm$.07. The difference was highly significant (W=1, $p < .001$, n=12).

6 Conclusion

This study brought together, for the first time, two recent research lines: The exploration of "semantic spaces" in the brain using distributional semantic models extracted from corpora, and the extension of the latter to image-based features. We showed that image-based distributional semantic measures significantly correlate with fMRI-based neural similarity patterns pertaining to categories of concrete concepts as well as concrete basic-level concepts expressed by specific words (although correlations, especially at the basic-concept level, are rather low, which might signify the need to develop still more performant distributional models and/or noise inherent to neural data). Moreover, image-based models complement a state-of-the-art text-based model, with the best performance achieved when the two modalities are combined. This not only presents an optimistic outlook for the future use of image-based models as an interpretative tool to explore issues of cognitive grounding, but also demonstrates that they are capturing useful additional aspects of meaning to

the text models, which are likely relevant for computational semantic tasks.

The weak comparative performance of the original Mitchell et al.'s Verb model is perhaps surprising given its previous success in prediction (Mitchell et al., 2008), but a useful reminder that a good predictor does not necessarily have to capture the internal structure of the data it predicts.

The lack of finding organisational differences between anatomical regions differentially described by the various models is perhaps disappointing, but not uncontroversial, given that the dataset was not originally designed to tease apart visual information from linguistic context. It is however interesting that in the more challenging word-level analysis some meaningful trend was visible. In future experiments it may prove valuable to configure a fMRI stimulus set where text-based and image-based interrelationships are maximally different. Collecting our own fMRI data will also allow us to move beyond exploratory analysis, to test sharper predictions about distributional models and their brain area correlates. There are also many opportunities for focusing analyses on different subsets of brain regions, with the semantic system identified by Binder et al. (2009) in particular presenting one interesting avenue for investigation.

Acknowledgments

This research was partially funded by a Google Research Award to the fifth author.

References

Shane Bergsma and Randy Goebel. 2011. Using visual information to predict lexical preference. In *Proceedings of RANLP*, pages 399–405, Hissar, Bulgaria.

Jeffrey R. Binder, Rutvik H. Desai, William W. Graves, and Lisa L. Conant. 2009. Where is the semantic system? a critical review and meta-analysis of 120 functional neuroimaging studies. *Cerebral Cortexl*, 12:2767–2796.

Vicki Bruce, Patrick R Green, and Georgeson Mark A. 2003. *Visual perception: Physiology, psychology, and ecology*. Psychology Pr.

Elia Bruni, Giang Binh Tran, and Marco Baroni. 2011. Distributional semantics from text and images. In *Proceedings of the EMNLP GEMS Workshop*, pages 22–32, Edinburgh, UK.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012a. Distributional semantics in Technicolor. In *Proceedings of ACL*, pages 136–145, Jeju Island, Korea.

Elia Bruni, Jasper Uijlings, Marco Baroni, and Nicu Sebe. 2012b. Distributional semantics with eyes: Using image analysis to improve computational representations of word meaning. In *Proceedings of ACM Multimedia*, pages 1219–1228, Nara, Japan.

Elia Bruni, Ulisse Bordignon, Adam Liska, Jasper Uijlings, and Irina Sergienya. 2013. Vsem: An open library for visual semantics representation. In *Proceedings of ACL*, Sofia, Bulgaria.

John Bullinaria and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Kai-min Chang, Tom Mitchell, and Marcel Just. 2011. Quantitative modeling of the neural representation of objects: How semantic feature norms can account for fMRI activation. *NeuroImage*, 56:716–727.

Linda L Chao, James V Haxby, and Alex Martin. 1999. Attribute-based neural substrates in temporal cortex for perceiving and knowing about objects. *Nature neuroscience*, 2(10):913–919.

Kenneth Church and Peter Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Stephen Clark. 2012. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd edition*. Blackwell, Malden, MA. In press.

Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. 2004. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, Prague, Czech Republic.

Jia Deng, Wei Dong, Richard Socher, Lia-Ji Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of CVPR*, pages 248–255, Miami Beach, FL.

Russell A Epstein. 2008. Parahippocampal and retrosplenial contributions to human spatial navigation. *Trends in cognitive sciences*, 12(10):388–396.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

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

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Proceedings of HLT-NAACL*, pages 91–99, Los Angeles, CA.

Melvyn A. Goodale and David Milner. 1992. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15:20–25.

Kristen Grauman and Bastian Leibe. 2011. *Visual Object Recognition*. Morgan & Claypool, San Francisco.

Peter Hagoort. 2005. On Broca, brain, and binding: a new framework. *Trends in cognitive sciences*, 9(9):416–423.

James Haxby, Ida Gobbini, Maura Furey, Alumit Ishai, Jennifer Schouten, and Pietro Pietrini. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293:2425–2430.

Alexander Huth, Shinji Nishimoto, An Vu, and Jack Gallant. 2012. A continuous semantic space describes the representation of thousands of object and action categories across the human brain. *Neuron*, 76(6):1210–1224.

Nancy Kanwisher and Galit Yovel. 2006. The fusiform face area: a cortical region specialized for the perception of faces. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1476):2109–2128.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. 2008. Representational similarity analysis–connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2.

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

Chee Wee Leong and Rada Mihalcea. 2011. Going beyond text: A hybrid image-text approach for measuring word relatedness. In *Proceedings of IJCNLP*, pages 1403–1407.

Max Louwerse. 2011. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science*, 3:273–302.

David G Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods*, 28:203–208.

Bruce D McCandliss, Laurent Cohen, and Stanislas Dehaene. 2003. The visual word form area: expertise for reading in the fusiform gyrus. *Trends in cognitive sciences*, 7(7):293–299.

Robert D McIntosh and Thomas Schenk. 2009. Two visual streams for perception and action: Current trends. *Neuropsychologia*, 47(6):1391–1396.

Earl K Miller, David J Freedman, and Jonathan D Wallis. 2002. The prefrontal cortex: categories, concepts and cognition. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 357(1424):1123–1136.

Tom Mitchell, Svetlana Shinkareva, Andrew Carlson, Kai-Min Chang, Vincente Malave, Robert Mason, and Marcel Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320:1191–1195.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Selecting corpus-semantic models for neurolinguistic decoding. In *Proceedings of *SEM*, pages 114–123, Montreal, Canada.

Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom Mitchell. 2009. Zero-shot learning with semantic output codes. In *Proceedings of NIPS*, pages 1410–1418, Vancouver, Canada.

Marius V Peelen and Paul E Downing. 2005. Selectivity for the human body in the fusiform gyrus. *Journal of Neurophysiology*, 93(1):603–608.

Francisco Pereira, Greg Detre, and Matthew Botvinick. 2011. Generating text from functional brain images. *Frontiers in Human Neuroscience*, 5(72). Published online: `http://www.frontiersin.org/human_neuroscience/10.3389/fnhum.2011.00072/abstract`.

Alexander T Sack. 2009. Parietal cortex and spatial cognition. *Behavioural brain research*, 202(2):153–161.

Hinrich Schütze. 1997. *Ambiguity Resolution in Natural Language Learning*. CSLI, Stanford, CA.

Carina Silberer and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *Proceedings of ACL*, Sofia, Bulgaria.

Josef Sivic and Andrew Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of ICCV*, pages 1470–1477, Nice, France.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

N Tzourio-Mazoyer, B Landeau, D Papathanassiou, F Crivello, O Etard, N Delcroix, B Mazoyer, and M Joliot. 2002. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *Neuroimage*, 15(1):273–289.

Easy Victories and Uphill Battles in Coreference Resolution

Greg Durrett and **Dan Klein**
Computer Science Division
University of California, Berkeley
{gdurrett,klein}@cs.berkeley.edu

Abstract

Classical coreference systems encode various syntactic, discourse, and semantic phenomena explicitly, using heterogenous features computed from hand-crafted heuristics. In contrast, we present a state-of-the-art coreference system that captures such phenomena implicitly, with a small number of homogeneous feature templates examining shallow properties of mentions. Surprisingly, our features are actually more effective than the corresponding hand-engineered ones at modeling these key linguistic phenomena, allowing us to win "easy victories" without crafted heuristics. These features are successful on syntax and discourse; however, they do not model semantic compatibility well, nor do we see gains from experiments with shallow semantic features from the literature, suggesting that this approach to semantics is an "uphill battle." Nonetheless, our final system[1] outperforms the Stanford system (Lee et al. (2011), the winner of the CoNLL 2011 shared task) by 3.5% absolute on the CoNLL metric and outperforms the IMS system (Björkelund and Farkas (2012), the best publicly available English coreference system) by 1.9% absolute.

1 Introduction

Coreference resolution is a multi-faceted task: humans resolve references by exploiting contextual and grammatical clues, as well as semantic information and world knowledge, so capturing each of these will be necessary for an automatic system to fully solve the problem. Acknowledging this complexity, coreference systems, either learning-based (Bengtson and Roth, 2008; Stoyanov et al., 2010; Haghighi and Klein, 2010; Rahman and Ng, 2011b) or rule-based (Haghighi and Klein, 2009; Lee et al., 2011), draw on diverse information sources and complex heuristics to resolve pronouns, model discourse, determine anaphoricity, and identify semantically compatible mentions. However, this leads to systems with many heterogenous parts that can be difficult to interpret or modify.

We build a learning-based, mention-synchronous coreference system that aims to use the simplest possible set of features to tackle the various aspects of coreference resolution. Though they arise from a small number of simple templates, our features are numerous, which works to our advantage: we can both implicitly model important linguistic effects and capture other patterns in the data that are not easily teased out by hand. As a result, our data-driven, homogeneous feature set is able to achieve high performance despite only using surface-level document characteristics and shallow syntactic information. We win "easy victories" without designing features and heuristics explicitly targeting particular phenomena.

Though our approach is successful at modeling syntax, we find semantics to be a much more challenging aspect of coreference. Our base system uses only two recall-oriented features on nominal and proper mentions: head match and exact string match. Building on these features, we critically evaluate several classes of semantic features which intu-

[1] The Berkeley Coreference Resolution System is available at http://nlp.cs.berkeley.edu.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1971–1982,
Seattle, Washington, USA, 18-21 October 2013.

itively should prove useful but have had mixed results in the literature, and we observe that they are ineffective for our system. However, these features are beneficial when gold mentions are provided to our system, leading us to conclude that the large number of system mentions extracted by most coreference systems (Lee et al., 2011; Fernandes et al., 2012) means that weak indicators cannot overcome the bias against making coreference links. Capturing semantic information in this shallow way is an "uphill battle" due to this structural property of coreference resolution.

Nevertheless, using a simple architecture and feature set, our final system outperforms the two best publicly available English coreference systems, the Stanford system (Lee et al., 2011) and the IMS system (Björkelund and Farkas, 2012), by wide margins: 3.5% absolute and 1.9% absolute, respectively, on the CoNLL metric.

2 Experimental Setup

Throughout this work, we use the datasets from the CoNLL 2011 shared task[2] (Pradhan et al., 2011), which is derived from the OntoNotes corpus (Hovy et al., 2006). When applicable, we use the standard automatic parses and NER tags for each document. All experiments use system mentions except where otherwise indicated. For each experiment, we report MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAF_e (Luo, 2005), as well as their average, the CoNLL metric. All metrics are computed using version 5 of the official CoNLL scorer.[3]

3 A Mention-Synchronous Framework

We first present the basic architecture of our coreference system, independent of a feature set. Unlike binary classification-based coreference systems where independent binary decisions are made about each pair (Soon et al., 2001; Bengtson and Roth, 2008; Versley et al., 2008; Stoyanov et al., 2010), we use a log-linear model to select at most one antecedent for each mention or determine that it begins a new cluster (Denis and Baldridge, 2008). In this mention-ranking or mention-synchronous framework, features examine single mentions to evaluate whether or not they are anaphoric and pairs of mentions to evaluate whether or not they corefer. While other work has used this framework as a starting point for entity-level systems (Luo et al., 2004; Rahman and Ng, 2009; Haghighi and Klein, 2010; Durrett et al., 2013), we will show that a mention-synchronous approach is sufficient to get state-of-the-art performance on its own.

3.1 Mention Detection

Our system first identifies a set of predicted mentions from text annotated with parses and named entity tags. We extract three distinct types of mentions: proper mentions from all named entity chunks except for those labeled as QUANTITY, CARDINAL, or PERCENT, pronominal mentions from single words tagged with PRP or PRP$, and nominal mentions from all other maximal NP projections. These basic rules are similar to those of Lee et al. (2011), except that their system uses an additional set of filtering rules designed to discard instances of pleonastic *it*, partitives, certain quantified noun phrases, and other spurious mentions. In contrast to this highly engineered approach and to systems which use a trained classifier to compute anaphoricity separately (Rahman and Ng, 2009; Björkelund and Farkas, 2012), we aim for the highest possible recall of gold mentions with a low-complexity method, leaving us with a large number of spurious system mentions that we will have to reject later.

3.2 Coreference Model

Figure 1 shows the mention-ranking architecture that serves as the backbone of our coreference system. Assume we have extracted n mentions from a document x, where x denotes the surface properties of a document and any precomputed information. The ith mention in a document has an associated random variable a_i taking values in the set $\{1, \ldots, i-1, \text{NEW}\}$; this variable specifies mention i's selected antecedent or indicates that it begins a new coreference chain. A setting of the a_i, denoted by $a = (a_1, ..., a_n)$, implies a unique set of coreference chains C that serve as our system output.

[2]This dataset is identical to the English portion of the CoNLL 2012 data, except for the absence of a small pivot text.

[3]Note that this version of the scorer implements a modified version of B^3, described in Cai and Strube (2010), that was used for the CoNLL shared tasks. The implementation of CEAF_e is also not exactly as described in Luo et al. (2004), but for completeness we include this metric as well.

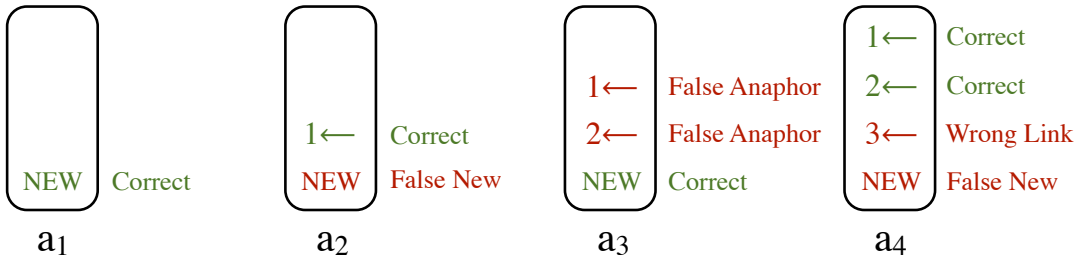


Figure 1: The basic structure of our coreference model. The ith mention in a document has i possible antecedence choices: link to one of the $i-1$ preceding mentions or begin a new cluster. We place a distribution over these choices with a log-linear model. Structurally different kinds of errors are weighted differently to optimize for final coreference loss functions; error types are shown corresponding to the decisions for each mention.

We use a log linear model of the conditional distribution $P(a|x)$ as follows:

$$P(a|x) \propto \exp\left(\sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(i, a_i, x)\right)$$

where $\mathbf{f}(i, a_i, x)$ is a feature function that examines the coreference decision a_i for mention i with document context x. When $a_i = \text{NEW}$, the features fired indicate the suitability of the given mention to be anaphoric or not; when $a_i = j$ for some j, the features express aspects of the pairwise linkage, and can examine any relevant attributes of the anaphor i or the antecedent j, since information about each mention is contained in x.

Inference in this model is efficient: because $\log P(a|x)$ decomposes linearly over mentions, we can compute $a_i = \arg\max_{a_i} P(a_i|x)$ separately for each mention and return the set of coreference chains implied by these decisions.

3.3 Learning

During learning, we optimize for conditional log-likelihood augmented with a parameterized loss function (Durrett et al., 2013). The main complicating factor in this process is that the supervision in coreference consists of a gold clustering C^* defined over gold mentions. This is problematic for two reasons: first, because the clustering is defined over gold mentions rather than our system mentions, and second, because a clustering does not specify a full antecedent structure of the sort our model produces. We can address the first of these problems by imputing singleton clusters for mentions that do not appear in the gold standard; our system will then simply learn to put spurious mentions in their own clusters. Singletons are always removed before evaluation because the OntoNotes corpus does not annotate them, so in this way we can neatly dispose of spurious mentions. To address the lack of explicit antecedents in C^*, we simply sum over all possible antecedent structures licensed by the gold clusters.

Formally, we will maximize the conditional log-likelihood of the set $\mathcal{A}(C^*)$ of antecedent vectors a for a document that are *consistent* with the gold annotation.[4] Consistency for an antecedent choice a_i under gold clusters C^* is defined as follows:

1. If $a_i = j$, a_i is consistent iff mentions i and j are present in C^* and are in the same cluster.

2. If $a_i = \text{NEW}$, a_i is consistent off mention i is not present in C^*, or it is present in C^* and has no gold antecedents, or it is present in C^* and none of its gold antecedents are among the set of system predicted mentions.

Given t training examples of the form (x_k, C_k^*), we write the following likelihood function:

$$\ell(\mathbf{w}) = \sum_{k=1}^{t} \log\left(\sum_{a \in \mathcal{A}(C_k^*)} P'(a|x_k)\right) + \lambda\|\mathbf{w}\|_1$$

where $P'(a|x_k) \propto P(a|x_k)\exp(l(a, C_k^*))$ with $l(a, C^*)$ being a real-valued loss function. The loss

[4] Because of this marginalization over latent antecedent choices, our objective is non-convex.

here plays an analogous role to the loss in structured max-margin objectives; incorporating it into a conditional likelihood objective is a technique called softmax-margin (Gimpel and Smith, 2010).

Our loss function $l(a, C^*)$ is a weighted linear combination of three error types, examples of which are shown in Figure 1. A false anaphor (FA) error occurs when a_i is chosen to be anaphoric when it should start a new cluster. A false new (FN) error occurs in the opposite case, when a_i wrongly indicates a new cluster when it should be anaphoric. Finally, a wrong link (WL) error occurs when the antecedent chosen for a_i is the wrong antecedent (but a_i is indeed anaphoric). Our final parameterized loss function is a weighted sum of the counts of these three error types:

$$l(a, C^*) = \alpha_{\text{FA}}\text{FA}(a, C^*) + \alpha_{\text{FN}}\text{FN}(a, C^*) + \alpha_{\text{WL}}\text{WL}(a, C^*)$$

where $\text{FA}(a, C^*)$ gives the number of false anaphor errors in prediction a with gold chains C^* (FN and WL are analogous). By setting α_{FA} low and α_{FN} high relative to α_{WL}, we can counterbalance the high number of singleton mentions and bias the system towards making more coreference linkages. We set $(\alpha_{\text{FA}}, \alpha_{\text{FN}}, \alpha_{\text{WL}}) = (0.1, 3.0, 1.0)$ and $\lambda = 0.001$ and optimize the objective using AdaGrad (Duchi et al., 2011).

4 Easy Victories from Surface Features

Our primary goal with this work is to show that a high-performance coreference system is attainable with a small number of feature templates that use only surface-level information sources. These features will be general-purpose and capture linguistic effects to the point where standard heuristic-driven features are no longer needed in our system.

4.1 SURFACE Features and Conjunctions

Our SURFACE feature set only considers the following properties of mentions and mention pairs:

- Mention type (nominal, proper, or pronominal)
- The complete string of a mention
- The semantic head of a mention
- The first word and last word of each mention
- The word immediately preceding and the word immediately following a mention
- Mention length, in words
- Two distance measures between mentions (number of sentences and number of mentions)

Feature name	Count
Features on the current mention	
[ANAPHORIC] + [HEAD WORD]	41371
[ANAPHORIC] + [FIRST WORD]	18991
[ANAPHORIC] + [LAST WORD]	19184
[ANAPHORIC] + [PRECEDING WORD]	54605
[ANAPHORIC] + [FOLLOWING WORD]	57239
[ANAPHORIC] + [LENGTH]	4304
Features on the antecedent	
[ANTECEDENT HEAD WORD]	57383
[ANTECEDENT FIRST WORD]	24239
[ANTECEDENT LAST WORD]	23819
[ANTECEDENT PRECEDING WORD]	53421
[ANTECEDENT FOLLOWING WORD]	55718
[ANTECEDENT LENGTH]	4620
Features on the pair	
[EXACT STRING MATCH (T/F)]	47
[HEAD MATCH (T/F)]	46
[SENTENCE DISTANCE, CAPPED AT 10]	2037
[MENTION DISTANCE, CAPPED AT 10]	1680

Table 1: Our SURFACE feature set, which exploits a small number of surface-level mention properties. Feature counts for each template are computed over the training set, and include features generated by our conjunction scheme (not explicitly shown in the table; see Figure 2), which yields large numbers of features at varying levels of expressivity.

Table 1 shows the SURFACE feature set. Features that look only at the current mention fire on all decisions ($a_i = j$ or $a_i = \text{NEW}$), whereas features that look at the antecedent in any way (the latter two groups of features) only fire on pairwise linkages ($a_i \neq \text{NEW}$).

Two conjunctions of each feature are also included: first with the "type" of the mention being resolved (either NOMINAL, PROPER, or, if it is pronominal, the citation form of the pronoun), and then additionally with the antecedent type (only if the feature is over a pairwise link). This conjunction process is shown in Figure 2. Note that features that just examine the antecedent will end up with

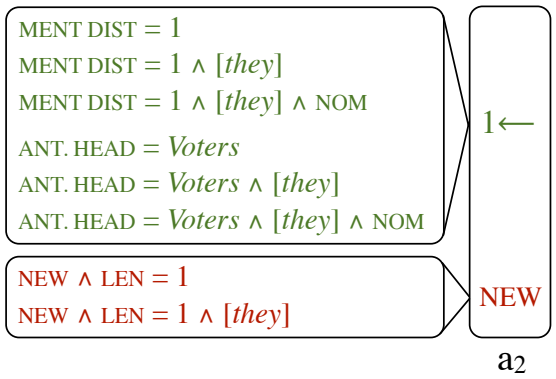


Figure 2: Demonstration of the conjunction scheme we use. Each feature on anaphoricity is conjoined with the type (NOMINAL, PROPER, or the citation form if it is a pronoun) of the mention being resolved. Each feature on a mention pair is additionally conjoined with the types of the current and antecedent mentions.

conjunctions that examine properties of the current mention as well, as shown with the ANT. HEAD feature in the figure.

Finally, we found it beneficial for our lexical indicator features to only fire on words occurring at least 20 times in the training set; for rare words, we use the part of speech of the word instead.

The performance of our system is shown in Table 2. We contrast our performance with that of the Stanford system (Lee et al. (2011), the winner of the CoNLL 2011 shared task) and the IMS system (Björkelund and Farkas (2012), the best publicly available English coreference system). Despite its simplicity, our SURFACE system is sufficient to outperform these sophisticated systems: the Stanford system uses a cascade of ten rule-based sieves each of which has customized heuristics, and the IMS system uses a similarly long pipeline consisting of a learned referentiality classifier followed by multiple resolvers, which are run in sequence and rely on the outputs of the previous resolvers as features.

4.2 Data-Driven versus Heuristic-Driven Features

Why are the SURFACE features sufficient to give high coreference performance, when they do not make apparent reference to important linguistic phenomena? The main reason is that they actually do capture the same phenomena as standard coreference features, just implicitly. For example, rather than having rules targeting person, number, gender, or animacy of mentions, we use conjunctions with pronoun identity, which contains this information. Rather than explicitly writing a feature targeting definiteness, our indicators on the first word of a mention will capture this and other effects. And finally, rather than targeting centering theory (Grosz et al., 1995) with rule-based features identifying syntactic positions (Stoyanov et al., 2010; Haghighi and Klein, 2010), our features on word context can identify configurational clues like whether a mention is preceded or followed by a verb, and therefore whether it is likely in subject or object position.[5]

	MUC	B^3	CEAF_e	Avg.
STANFORD	60.46	65.48	47.07	57.67
IMS	62.15	65.57	46.66	58.13
SURFACE	**64.39**	**66.78**	**49.00**	**60.06**

Table 2: Results for our SURFACE system, the STANFORD system, and the IMS system on the CoNLL 2011 development set. Complete results are shown in Table 7. Despite using limited information sources, our system is able to substantially outperform the other two, the two best publicly-available English coreference systems. Bolded values are significant with $p < 0.05$ according to a bootstrap resampling test.

Not only are data-driven features able to capture the same phenomena as heuristic-driven features, but they do so at a finer level of granularity, and can therefore model more patterns in the data. To contrast these two types of features, we experiment with three ablated versions of our system, where we replace data-driven features with their heuristic-driven counterparts:

1. Instead of using an indicator on the first word of a mention (1STWORD), we instead fire a feature based on that mention's manually-computed definiteness (DEF).

2. Instead of conjoining features on pronominal-pronominal linkages with the citation form of

[5]Heuristic-driven approaches were historically more appropriate, since past coreference corpora such as MUC and ACE were smaller and therefore more prone to overfitting feature-rich models. However, the OntoNotes dataset contains thousands of documents, so having support for features is less of a concern.

	MUC	B^3	CEAF_e	Avg.
SURFACE	64.39	66.78	49.00	60.06
−1STWORD	63.32	66.22	47.89	59.14
+DEF−1STWORD	63.79	66.46	48.35	59.53
−PRONCONJ	59.97	63.46	47.94	57.12
+AGR−PRONCONJ	63.54	66.10	48.72	59.45
−CONTEXT	60.88	64.66	47.60	57.71
+POSN−CONTEXT	62.45	65.44	48.08	58.65
+DEF+AGR+POSN	64.55	66.93	48.94	60.14

Table 3: CoNLL metric scores on the development set, for the three different ablations and replacement features described in Section 4.2. Feature types are described in the text; + indicates inclusion of that feature class, − indicates exclusion. Each individual shallow indicator appears to do as well at capturing its target phenomenon as the hand-engineered features, while capturing other information as well. Moreover, the hand-engineered features give no benefit over the SURFACE system.

each pronoun (PRONCONJ), we only conjoin with a PRONOUN indicator and add features targeting the person, number, gender, and animacy of the two pronouns (AGR).

3. Instead of using our context features on the preceding and following word (CONTEXT), we use manual determinations of when mentions are in subject, direct object, indirect objection, or oblique position (POSN).

All rules for computing person, number, gender, animacy, definiteness, and syntactic position are taken from the system of Lee et al. (2011).

Table 3 shows each of the target ablations, as well as the SURFACE system with the DEF, AGR, and POSN features added. While the heuristic-driven feature always help over the corresponding ablated system, they cannot do the work of the fine-grained data-driven features. Most tellingly, though, none of the heuristic-driven features give statistically significant improvements on top of the data-driven features we have already included, indicating that we are at the point of diminishing returns on modeling those specific phenomena. While this does not preclude further engineering to take better advantage of other syntactic constraints, our simple features represent an "easy victory" on this subtask.

5 Uphill Battles on Semantics

In Section 4, we gave a simple set of features that yielded a high-performance coreference system; this high performance is possible because features targeting only superficial properties in a fine-grained way can actually model complex linguistic constraints. However, while our existing features capture syntactic and discourse-level phenomena surprisingly well, they are not effective at capturing semantic phenomena like type compatibility. We will show that due to structural aspects of the coreference resolution problem, even a combination of several shallow semantic features from the literature fails to adequately model semantics.

5.1 Analysis of the SURFACE System

What can the SURFACE system resolve correctly, and what errors does it still make? To answer this question, we will split mentions into several categories based on their observable properties and the gold standard coreference information, and examine our system's accuracy on each mention subclass in order to more thoroughly characterize its performance.[6] These categories represent important distinctions in terms of the difficulty of mention resolution for our system.

We first split mentions into three categories by their *status* in the gold standard: singleton (unannotated in the OntoNotes corpus), starting a new entity with at least two mentions, or anaphoric. It is important to note that while singletons and mentions starting new entities are outwardly similar in that they have no antecedents, and the prediction should be the same in either case (NEW), we treat them as distinct because the factors that impact the coreference decision differ in the two cases. Mentions that start new clusters are semantically similar to anaphoric mentions, but may be marked by heaviness or by a tendency to be named entities, whereas singletons may be generic or temporal NPs which might be thought of as coreferent in a loose sense, but are not

[6]This method of analysis is similar to that undertaken in Stoyanov et al. (2009) and Rahman and Ng (2011b), though we split our mentions along different axes, and can simply evaluate on accuracy because our decisions do not directly imply multiple links, as they do in binary classification-based systems (Stoyanov et al., 2009) or in entity-mention models (Rahman and Ng, 2011b).

	Nominal/Proper				Pronominal	
	1st w/head		2nd+ w/head			
Singleton	99.7%	18.1K	85.5%	7.3K	66.5%	1.7K
Starts Entity	98.7%	2.1K	78.9%	0.7K	48.5%	0.3K
Anaphoric	7.9%	0.9K	75.5%	3.9K	72.0%	4.4K

Table 4: Analysis of our SURFACE system on the development set. We characterize each predicted mention by its status in the gold standard (singleton, starting a new entity, or anaphoric), its type (pronominal or nominal/proper), and by whether its head has appeared as the head of a previous mention. Each cell shows our system's accuracy on that mention class as well as the size of the class. The biggest weakness of our system appears to be its inability to resolve anaphoric mentions with new heads (bottom-left cell).

included in the OntoNotes dataset due to choices in the annotation standard.

Second, we divide mentions by their type, pronominal versus nominal/proper; we then further subdivide nominals and propers based on whether or not the head word of the mention has appeared as the head of a previous mention in the document.

Table 4 shows the results of our analysis. In each cell, we show the fraction of mentions that we correctly resolve (i.e., for which we make an antecedence decision consistent with the gold standard), as well as the total number of mentions falling into that cell. First, we observe that there are a surprisingly large number of singleton mentions with misleading head matches to previous mentions (often recurring temporal nouns phrases, like *July*). The features in our system targeting anaphoricity are useful for exactly this reason: the more bad head matches we can rule out based on other criteria, the more strongly we can rely on head match to make correct linkages.

Our system is most noticeably poor at resolving anaphoric mentions whose heads have not appeared before. The fact that exact match and head match are our only recall-oriented features on nominals and propers is starkly apparent here: when we cannot rely on head match, as is true for this mention class, we only resolve 7.9% of anaphoric mentions correctly.[7] Many of the mentions in this category can only be correctly resolved by exploiting world knowledge, so we will need to include features that capture this knowledge in some fashion.

5.2 Incorporating Shallow Semantics

As we were able to incorporate syntax with shallow features, so too might we hope to incorporate semantics. However, the semantic information contained even in a coreference corpus of thousands of documents is insufficient to generalize to unseen data,[8] so system designers have turned to external resources such as semantic classes derived from WordNet (Soon et al., 2001), WordNet hypernymy or synonymy (Stoyanov et al., 2010), semantic similarity computed from online resources (Ponzetto and Strube, 2006), named entity type features, gender and number match using the dataset of Bergsma and Lin (2006), and features from unsupervised clusters (Hendrickx and Daelemans, 2007; Durrett et al., 2013). In this section, we consider the following subset of these information sources:

- WordNet hypernymy and synonymy
- Number and gender data for nominals and propers from Bergsma and Lin (2006)
- Named entity types
- Latent clusters computed from English Gigaword (Graff et al., 2007), where a latent cluster label generates each nominal head (excluding pronouns) and a conjunction of its verbal governor and semantic role, if any (Durrett et al., 2013). We use twenty clusters, which include clusters like *president* and *leader* (things which *announce*).

Together, we call these the SEM features. We show results from this expansion of the feature set in Table 5. When using system mentions, the improvements are not statistically significant on every metric, and are quite marginal given that these features add information that is intuitively central to coreference and otherwise unavailable to the system. We explore the reasons behind this in the next section.

[7] There are an additional 346 anaphoric nominal/proper mentions in the 2nd+ category whose heads only appeared previously as part of a different cluster; we only resolve 1.7% of these extremely tricky cases correctly.

[8] We experimented with bilexical features on head pairs, but they did not give statistically significant improvements over the SURFACE features.

	MUC	B^3	CEAF_e	Avg.
SURFACE	64.39	66.78	49.00	60.06
SURFACE+SEM	64.70	**67.27**	**49.28**	**60.42**
SURFACE (G)	82.80	74.10	68.33	75.08
SURFACE+SEM (G)	**84.49**	**75.65**	**69.89**	**76.68**

Table 5: CoNLL metric scores on the development set for our SEM features when added on top of our SURFACE features. We experiment on both system mentions and gold mentions. Surprisingly, despite the fact that absolute performance numbers are much higher on gold mentions and there is less room for improvement, the semantic features help much more than they do on system mentions.

5.3 Analysis of Semantic Features

The main reason that weak semantic cues are not more effective is the small fraction of positive coreference links present in the training data. From Table 4, the number of annotated coreferent spans in the OntoNotes data is about a factor of five smaller than the number of system mentions.[9] This both means that most NPs are not coreferent, and for those that are, choosing the correct links is much more difficult because of the large number of possible antecedents. Even head match, which is generally considered a high-precision indicator (Lee et al., 2011), would introduce many spurious coreference arcs if applied too liberally (see Table 4).

In light of this fact, a system needs very strong evidence to overcome the default hypothesis that a mention is not coreferent, and a weak indicator will have such a high "false positive" rate that it cannot be relied on (given high weight, this feature would do more harm than good, by introducing many false linkages).

To confirm this intuition, we show in the bottom part of Table 5 results when we apply these semantic features on top of our SURFACE system on *gold mentions*, where there are no singletons. In the gold mention setting, we see that the semantic features give a consistent improvement on every metric. Moreover, if we look at a breakdown of errors, the main improvement the semantic features give us is on resolution of anaphoric nominals with no head match: accuracy on the 1601 mentions that fall into this category improves from 28.0% to 37.9%. On predicted mentions, by contrast, this category only improves from 7.9% to 12.2%, a much smaller absolute improvement and one that comes at the expense of performance on most other resolution class. The one class that does not get worse, singleton pronouns, actually improves by a similar 4% margin, indicating that roughly half of the gains we observe are not even necessarily a result of our features doing what they were designed to do.

Our weak cues do yield some small gains, so there is hope that better weak indicators of semantic compatibility could prove more useful. However, while extremely high-precision approaches with carefully engineered features have been shown to be successful (Rahman and Ng, 2011a; Bansal and Klein, 2012; Recasens et al., 2013a), we conclude that capturing semantics in a data-driven, shallow manner remains an uphill battle.

6 FINAL System and Results

While semantic features ended up giving only marginal benefit, we have demonstrated that nevertheless our SURFACE system is a state-of-the-art English coreference system. However, there remain a few natural features that we omitted in order to keep the system as simple as possible, since they were orthogonal to the discussion of data-driven versus heuristic-driven features and do not target world knowledge. Before giving final results, we will present a small set of additional features that consider four additional mention properties beyond those in Section 4.1:

- Whether two mentions are nested
- Ancestry of each mention head: the dependency parent and grandparent POS tags and arc directions (shown in Figure 3)
- The speaker of each mention
- Number and gender of each mention as determined by Bergsma and Lin (2006)

The specific additional features we use are shown in Table 6. Note that unlike in Section 5, we use the number and gender information only on the antecedent. Due to our conjunction scheme, both this

[9] This observation is more general than just our system: the majority of coreference systems, including the winners of the CoNLL shared tasks (Lee et al., 2011; Fernandes et al., 2012), opt for high mention recall and resolve a relatively large number of system mentions.

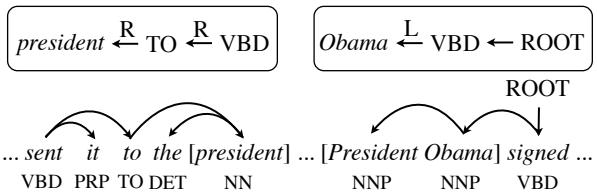


Figure 3: Demonstration of the ancestry extraction process. These features capture more sophisticated configurational information than our context word features do: in this example, *president* is in a characteristic indirect object position based on its dependency parents, and *Obama* is the subject of the main verb of the sentence.

Feature name	Count
Features of the SURFACE system	418704
Features on the current mention	
[ANAPHORIC] + [CURRENT ANCESTRY]	46047
Features on the antecedent	
[ANTECEDENT ANCESTRY]	53874
[ANTECEDENT GENDER]	338
[ANTECEDENT NUMBER]	290
Features on the pair	
[HEAD CONTAINED (T/F)]	136
[EXACT STRING CONTAINED (T/F)]	133
[NESTED (T/F)]	355
[DOC TYPE] + [SAME SPEAKER (T/F)]	437
[CURRENT ANCESTRY] + [ANT. ANCESTRY]	2555359

Table 6: FINAL feature set; note that this includes the SURFACE feature set. As with the features of the SURFACE system, two conjoined variants of each feature are included: first with the type of the current mention (NOMINAL, PROPER, or the citation form of the pronoun), then with the types of both mentions in the pair. These conjunctions allow antecedent features on gender and number to impact pronoun resolution, and they allow speaker match to capture effects like *I* and *you* being coreferent when the speakers differ.

semantic information and the speaker information can apply in a fine-grained way to different pronouns, and can therefore improve pronoun resolution substantially; however, these features generally only improve pronoun resolution.

Full results for our SURFACE and FINAL feature sets are shown in Table 7. Again, we compare to Lee et al. (2011) and Björkelund and Farkas (2012).[10] Despite our system's emphasis on one-pass resolution with as simple a feature set as possible, we are able to outperform even these sophisticated systems by a wide margin.

7 Related Work

Many of the individual features we employ in the FINAL feature set have appeared in other coreference systems (Björkelund and Nugues, 2011; Rahman and Ng, 2011b; Fernandes et al., 2012). However, other authors have often emphasized bilexical features on head pairs, whereas our features are heavily monolexical. For feature conjunctions, other authors have exploited three classes (Lee et al., 2011) or automatically learned conjunction schemes (Fernandes et al., 2012; Lassalle and Denis, 2013), but to our knowledge we are the first to do fine-grained modeling of every pronoun. Inclusion of a hierarchy of features with regularization also means that we organically get distinctions among different mention types without having to choose a level of granularity a priori, unlike the distinct classifiers employed by Denis and Baldridge (2008).

In terms of architecture, many coreference systems operate in a pipelined fashion, making partial decisions about coreference or pruning arcs before full resolution. Some systems use separate rule-based and learning-based passes (Chen and Ng, 2012; Fernandes et al., 2012), a series of learning-based passes (Björkelund and Farkas, 2012), or referentiality classifiers that prune the set of mentions before resolution (Rahman and Ng, 2009; Björkelund and Farkas, 2012; Recasens et al., 2013b). By contrast, our system resolves all mentions in one pass and does not need pruning: the SURFACE system can train in less than two hours without any subsampling of coreference arcs, and rule-based pruning of coreference arcs actually causes our system to perform less well, since our features can learn valuable information from these negative examples.

[10] Discrepancies between scores here and those printed in Pradhan et al. (2012) arise from two sources: improvements to the system of Lee et al. (2011) since the first CoNLL shared task, and a fix to the scoring of B^3 in the official scorer since results of the two CoNLL shared tasks were released. Unfortunately, because of this bug in the scoring program, direct comparison to the printed results of the other highest-scoring English systems, Fernandes et al. (2012) and Martschat et al. (2012), is impossible.

	MUC			B^3			CEAF_e			Avg.
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	F_1
	CoNLL 2011 Development Set									
STANFORD	61.62	59.34	60.46	74.05	58.70	65.48	45.98	48.22	47.07	57.67
IMS	66.67	58.20	62.15	77.60	56.77	65.57	42.92	51.11	46.66	58.13
SURFACE*	68.42	60.80	64.39	76.57	59.21	66.78	45.30	53.36	49.00	60.06
FINAL*	68.97	63.47	**66.10**	76.58	62.06	**68.56**	47.32	53.19	**50.09**	**61.58**
	CoNLL 2011 Test Set									
STANFORD	60.91	62.13	61.51	70.61	57.31	63.27	45.79	44.56	45.17	56.65
IMS	68.15	61.60	64.71	75.97	56.39	64.73	42.30	48.88	45.35	58.26
FINAL*	66.81	66.04	**66.43**	71.07	61.89	**66.16**	47.37	48.22	**47.79**	**60.13**

Table 7: CoNLL metric scores for our systems on the CoNLL development and blind test sets, compared to the results of Lee et al. (2011) (STANFORD) and Björkelund and Farkas (2012) (IMS). Starred systems are contributions of this work. Bolded F_1 values represent statistically significant improvements over other systems with $p < 0.05$ using a bootstrap resampling test. Metric values reflect version 5 of the CoNLL scorer.

8 Conclusion

We have presented a coreference system that uses a simple, homogeneous set of features in a discriminative learning framework to achieve high performance. Large numbers of lexicalized, data-driven features implicitly model linguistic phenomena such as definiteness and centering, obviating the need for heuristic-driven rules explicitly targeting these same phenomena. Additional semantic features give only slight benefit beyond head match because they do not provide strong enough signals of coreference to improve performance in the system mention setting; modeling semantic similarity still requires complex outside information and deep heuristics.

Our system, the Berkeley Coreference Resolution System, is publicly available at `http://nlp.cs.berkeley.edu`.

Acknowledgments

This work was partially supported by BBN under DARPA contract HR0011-12-C-0014 and by an NSF fellowship for the first author. Thanks to Sameer Pradhan for helpful discussions regarding the CoNLL scoring program, and thanks to the anonymous reviewers for their insightful comments.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for Scoring Coreference Chains. In *Proceedings of the Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*.

Mohit Bansal and Dan Klein. 2012. Coreference Semantics from Web Features. In *Proceedings of the Association for Computational Linguistics*.

Eric Bengtson and Dan Roth. 2008. Understanding the Value of Features for Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Shane Bergsma and Dekang Lin. 2006. Bootstrapping Path-Based Pronoun Resolution. In *Proceedings of the Conference on Computational Linguistics and the Association for Computational Linguistics*.

Anders Björkelund and Richárd Farkas. 2012. Data-driven Multilingual Coreference Resolution using Resolver Stacking. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Proceedings and Conference on Computational Natural Language Learning - Shared Task*.

Anders Björkelund and Pierre Nugues. 2011. Exploring Lexicalized Features for Coreference Resolution. In *Proceedings of the Conference on Computational Natural Language Learning: Shared Task*.

Jie Cai and Michael Strube. 2010. Evaluation Metrics for End-to-End Coreference Resolution Systems. In *Proceedings of the Special Interest Group on Discourse and Dialogue*.

Chen Chen and Vincent Ng. 2012. Combining the Best of Two Worlds: A Hybrid Approach to Multilingual Coreference Resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Proceedings and Conference on Computational Natural Language Learning - Shared Task*.

Pascal Denis and Jason Baldridge. 2008. Specialized Models and Ranking for Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Greg Durrett, David Hall, and Dan Klein. 2013. Decentralized Entity-Level Modeling for Coreference Resolution. In *Proceedings of the Association for Computational Linguistics*.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent Structure Perceptron with Feature Induction for Unrestricted Coreference Resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Proceedings and Conference on Computational Natural Language Learning - Shared Task*.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-Margin CRFs: Training Log-Linear Models with Cost Functions. In *Proceedings of the North American Chapter for the Association for Computational Linguistics*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2007. English Gigaword Third Edition. Linguistic Data Consortium, Catalog Number LDC2007T07.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics*, 21(2):203–225, June.

Aria Haghighi and Dan Klein. 2009. Simple Coreference Resolution with Rich Syntactic and Semantic Features. In *Proceedings of Empirical Methods in Natural Language Processing*.

Aria Haghighi and Dan Klein. 2010. Coreference Resolution in a Modular, Entity-Centered Model. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Iris Hendrickx and Walter Daelemans, 2007. *Adding Semantic Information: Unsupervised Clusters for Coreference Resolution*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Short Papers*.

Emmanuel Lassalle and Pascal Denis. 2013. Improving Pairwise Coreference Models Through Feature Space Hierarchy Learning. In *Proceedings of the Association for Computational Linguistics*.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. In *Proceedings of the Conference on Computational Natural Language Learning: Shared Task*.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree. In *Proceedings of the Association for Computational Linguistics*.

Xiaoqiang Luo. 2005. On Coreference Resolution Performance Metrics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sebastian Martschat, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. 2012. A Multigraph Model for Coreference Resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Proceedings and Conference on Computational Natural Language Learning - Shared Task*.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution. In *Proceedings of the North American Chapter of the Association of Computational Linguistics*.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the Conference on Computational Natural Language Learning: Shared Task*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*.

Altaf Rahman and Vincent Ng. 2009. Supervised Models for Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Altaf Rahman and Vincent Ng. 2011a. Coreference Resolution with World Knowledge. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies*.

Altaf Rahman and Vincent Ng. 2011b. Narrowing the Modeling Gap: A Cluster-Ranking Approach to Coreference Resolution. *Journal of Artificial Intelligence Research*, 40(1):469–521, January.

Marta Recasens, Matthew Can, and Daniel Jurafsky. 2013a. Same Referent, Different Words: Unsupervised Mining of Opaque Coreferent Mentions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013b. The Life and Death of Discourse Entities: Identifying Singleton Mentions. In

Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*, 27(4):521–544, December.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in Noun Phrase Coreference Resolution: Making Sense of the State-of-the-Art. In *Proceedings of the Association for Computational Linguistics*.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference Resolution with Reconcile. In *Proceedings of the Association for Computational Linguistics: Short Papers*.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A Modular Toolkit for Coreference Resolution. In *Proceedings of the Association for Computational Linguistics: Demo Session*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A Model-Theoretic Coreference Scoring Scheme. In *Proceedings of the Conference on Message Understanding*.

Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction

Valentin I. Spitkovsky
valentin@cs.stanford.edu

Hiyan Alshawi
hiyan@google.com

Daniel Jurafsky
jurafsky@stanford.edu

Abstract

Many statistical learning problems in NLP call for local model search methods. But accuracy tends to suffer with current techniques, which often explore either too narrowly or too broadly: hill-climbers can get stuck in local optima, whereas samplers may be inefficient. We propose to arrange individual local optimizers into organized networks. Our building blocks are operators of two types: (i) *transform*, which suggests new places to search, via non-random restarts from already-found local optima; and (ii) *join*, which merges candidate solutions to find better optima. Experiments on grammar induction show that pursuing different transforms (e.g., discarding parts of a learned model or ignoring portions of training data) results in improvements. Groups of locally-optimal solutions can be further perturbed jointly, by constructing mixtures. Using these tools, we designed several modular dependency grammar induction networks of increasing complexity. Our complete system achieves 48.6% accuracy (directed dependency macro-average over all 19 languages in the 2006/7 CoNLL data) — more than 5% higher than the previous state-of-the-art.

1 Introduction

Statistical methods for grammar induction often boil down to solving non-convex optimization problems. Early work attempted to locally maximize the likelihood of a corpus, using EM to estimate probabilities of dependency arcs between word bigrams (Paskin 2001a; 2001b). That parsing model has since been extended to make unsupervised learning more feasible (Klein and Manning, 2004; Headden et al., 2009; Spitkovsky et al., 2012b). But even the latest techniques can be quite error-prone and sensitive to initialization, because of approximate, local search.

In theory, global optima can be found by enumerating all parse forests that derive a corpus, though this is usually prohibitively expensive in practice. A preferable brute force approach is sampling, as in Markov-chain Monte Carlo (MCMC) and random restarts (Hu et al., 1994), which hit exact solutions eventually. Restarts can be giant steps in a parameter space that undo all previous work. At the other extreme, MCMC may cling to a neighborhood, rejecting most proposed moves that would escape a local attractor. Sampling methods thus take unbounded time to solve a problem (and can't certify optimality) but are useful for finding approximate solutions to grammar induction (Cohn et al., 2011; Mareček and Žabokrtský, 2011; Naseem and Barzilay, 2011).

We propose an alternative (deterministic) search heuristic that combines local optimization via EM with *non*-random restarts. Its new starting places are informed by previously found solutions, unlike conventional restarts, but may not resemble their predecessors, unlike typical MCMC moves. We show that one good way to construct such steps in a parameter space is by forgetting some aspects of a learned model. Another is by merging promising solutions, since even simple interpolation (Jelinek and Mercer, 1980) of local optima may be superior to all of the originals. Informed restarts can make it possible to explore a combinatorial search space more rapidly and thoroughly than with traditional methods alone.

2 Abstract Operators

Let C be a collection of counts — the sufficient statistics from which a candidate solution to an optimization problem could be computed, e.g., by smoothing and normalizing to yield probabilities. The counts may be fractional and solutions could take the form of multinomial distributions. A local optimizer L will convert C into $C^* = L_{\mathcal{D}}(C)$ — an updated collection of counts, resulting in a probabilistic model that is no less (and hopefully more) consistent with a data set $\mathcal{D}$ than the original C:

$$C \longrightarrow \boxed{L_{\mathcal{D}}} \longrightarrow C^* \tag{1}$$

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1983–1995,
Seattle, Washington, USA, 18-21 October 2013.

Unless C^* is a global optimum, we should be able to make further improvements. But if L is idempotent (and ran to convergence) then $L(L(C)) = L(C)$. Given only C and $L_{\mathcal{D}}$, the single-node optimization network above would be the minimal search pattern worth considering. However, if we had another optimizer L' — or a fresh starting point C' — then more complicated networks could become useful.

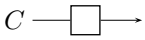
2.1 Transforms (Unary)

New starts could be chosen by perturbing an existing solution, as in MCMC, or independently of previous results, as in random restarts. We focus on intermediate changes to C, without injecting randomness.

All of our transforms involve selective forgetting or filtering. For example, if the probabilistic model that is being estimated decomposes into independent constituents (e.g., several multinomials) then a subset of them can be reset to uniform distributions, by discarding associated counts from C. In text classification, this could correspond to eliminating frequent or rare tokens from bags-of-words. We use circular shapes to represent such model ablation operators:

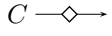
(2)

An orthogonal approach might separate out various counts in C by their provenance. For instance, if $\mathcal{D}$ consisted of several heterogeneous data sources, then the counts from some of them could be ignored: a classifier might be estimated from just news text. We will use squares to represent data-set filtering:



(3)

Finally, if C represents a mixture of possible interpretations over $\mathcal{D}$ — e.g., because it captures the output of a "soft" EM algorithm — contributions from less likely, noisier completions could also be suppressed (and their weights redistributed to the more likely ones), as in "hard" EM. Diamonds will represent plain (single) steps of Viterbi training:

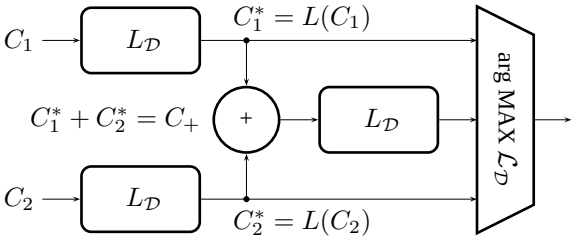


(4)

2.2 Joins (Binary)

Starting from different initializers, say C_1 and C_2, it may be possible for L to arrive at distinct local optima, $C_1^* \neq C_2^*$. The better of the two solutions, according to likelihood $\mathcal{L}_{\mathcal{D}}$ of $\mathcal{D}$, could then be selected — as is standard practice when sampling.

Our joining technique could do better than either C_1^* or C_2^*, by entertaining also a third possibility, which combines the two candidates. We construct a mixture model by adding together all counts from C_1^* and C_2^* into $C_+ = C_1^* + C_2^*$. Original initializers C_1, C_2 will, this way, have equal pull on the merged model,[1] regardless of nominal size (because C_1^*, C_2^* will have converged using a shared training set, $\mathcal{D}$). We return the best of C_1^*, C_2^* and $C_+^* = L(C_+)$. This approach may uncover more (and never returns less) likely solutions than choosing among C_1^*, C_2^* alone:



(5)

We will use a short-hand notation to represent the combiner network diagrammed above, less clutter:

(6)

3 The Task and Methodology

We apply transform and join paradigms to grammar induction, an important problem of computational linguistics that involves notoriously difficult objectives (Pereira and Schabes, 1992; de Marcken, 1995; Gimpel and Smith, 2012, *inter alia*). The goal is to induce grammars capable of parsing unseen text. Input, in both training and testing, is a sequence of tokens labeled as: (i) a lexical item and its category, (w, c_w); (ii) a punctuation mark; or (iii) a sentence boundary. Output is unlabeled dependency trees.

3.1 Models and Data

We constrain all parse structures to be projective, via dependency-and-boundary grammars (Spitkovsky et al., 2012a; 2012b): DBMs 0–3 are head-outward generative parsing models (Alshawi, 1996) that distinguish complete sentences from incomplete fragments in a corpus $\mathcal{D}$: $\mathcal{D}_{\text{comp}}$ comprises inputs ending with punctuation; $\mathcal{D}_{\text{frag}} = \mathcal{D} - \mathcal{D}_{\text{comp}}$ is everything

[1] If desired, a scaling factor could be used to bias C_+ towards either C_1^* or C_2^*, for example based on their likelihood ratio.

else. The "complete" subset is further partitioned into simple sentences, $\mathcal{D}_{\text{simp}} \subseteq \mathcal{D}_{\text{comp}}$, with no internal punctuation, and others, which may be complex.

As an example, consider the beginning of an article from (simple) Wikipedia: (i) ***Linguistics*** (ii) *Linguistics (sometimes called philology) is the science that studies language.* (iii) *Scientists who study language are called linguists.* Since the title does not end with punctuation, it would be relegated to $\mathcal{D}_{\text{frag}}$. But two complete sentences would be in $\mathcal{D}_{\text{comp}}$, with the last also filed under $\mathcal{D}_{\text{simp}}$, as it has only a trailing punctuation mark. Spitkovsky et al. suggested two curriculum learning strategies: (i) one in which induction begins with clean, simple data, $\mathcal{D}_{\text{simp}}$, and a basic model, DBM-1 (2012b); and (ii) an alternative bootstrapping approach: starting with still more, simpler data — namely, short inter-punctuation fragments up to length $l = 15$, $\mathcal{D}^l_{\text{split}} \supseteq \mathcal{D}^l_{\text{simp}}$ — and a bare-bones model, DBM-0 (2012a). In our example, $\mathcal{D}_{\text{split}}$ would hold five text snippets: (i) *Linguistics*; (ii) *Linguistics*; (iii) *sometimes called philology*; (iv) *is the science that studies language*; and (v) *Scientists who study language are called linguists.* Only the last piece of text would still be considered complete, isolating its contribution to sentence root and boundary word distributions from those of incomplete fragments. The sparse model, DBM-0, assumes a uniform distribution for roots of incomplete inputs and reduces conditioning contexts of stopping probabilities, which works well with split data. We will exploit both DBM-0 and the full DBM,[2] drawing also on split, simple and raw views of input text.

All experiments prior to final multi-lingual evaluation will use the Penn English Treebank's Wall Street Journal (WSJ) portion (Marcus et al., 1993) as the underlying tokenized and sentence-broken corpus $\mathcal{D}$. Instead of gold parts-of-speech, we plugged in 200 context-sensitive unsupervised tags, from Spitkovsky et al. (2011c),[3] for the word categories.

3.2 Smoothing and Lexicalization

All unlexicalized instances of DBMs will be estimated with "add one" (a.k.a. Laplace) smoothing, using only the word category c_w to represent a token. Fully-lexicalized grammars (L-DBM) are left unsmoothed, and represent each token as both a word and its category, i.e., the whole pair (w, c_w). To evaluate a lexicalized parsing model, we will always obtain a delexicalized-and-smoothed instance first.

3.3 Optimization and Viterbi Decoding

We use "early-switching lateen" EM (Spitkovsky et al., 2011a, §2.4) to train unlexicalized models, alternating between the objectives of ordinary (soft) and hard EM algorithms, until neither can improve its own objective without harming the other's. This approach does not require tuning termination thresholds, allowing optimizers to run to numerical convergence if necessary, and handles only our shorter inputs ($l \leq 15$), starting with soft EM ($L = \text{SL}$, for "soft lateen"). Lexicalized models will cover full data ($l \leq 45$) and employ "early-stopping lateen" EM (2011a, §2.3), re-estimating via hard EM until soft EM's objective suffers. Alternating EMs would be expensive here, since updates take (at least) $O(l^3)$ time, and hard EM's objective ($L = \text{H}$) is the one better suited to long inputs (Spitkovsky et al., 2010).

Our decoders always force an inter-punctuation fragment to derive itself (Spitkovsky et al., 2011b, §2.2).[4] In evaluation, such (*loose*) constraints may help attach *sometimes* and *philology* to *called* (and *the science...* to *is*). In training, stronger (*strict*) constraints also disallow attachment of fragments' heads by non-heads, to connect *Linguistics*, *called* and *is* (assuming each piece got parsed correctly).

3.4 Final Evaluation and Metrics

Evaluation is against held-out CoNLL shared task data (Buchholz and Marsi, 2006; Nivre et al., 2007), spanning 19 languages. We compute performance as directed dependency accuracies (DDA), fractions of correct unlabeled arcs in parsed output (an extrinsic metric).[5] For most WSJ experiments we include also sentence and parse tree cross-entropies (soft and hard EMs' intrinsic metrics), in bits per token (bpt).

[2] We use the short-hand DBM to refer to DBM-3, which is equivalent to DBM-2 if $\mathcal{D}$ has no internally-punctuated sentences ($\mathcal{D}$=$\mathcal{D}_{\text{split}}$), and DBM-1 if all inputs also have trailing punctuation ($\mathcal{D}$=$\mathcal{D}_{\text{simp}}$); DBM_0 is our short-hand for DBM-0.

[3] `http://nlp.stanford.edu/pubs/goldtags-data.tar.bz2`

[4] But these constraints do not impact training with shorter inputs, since there is no internal punctuation in $\mathcal{D}_{\text{split}}$ or $\mathcal{D}_{\text{simp}}$.

[5] We converted gold labeled constituents in WSJ to unlabeled reference dependencies using deterministic "head-percolation" rules (Collins, 1999); sentence root symbols, though not punctuation arcs, contribute to scores, as is standard (Paskin, 2001b).

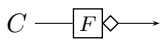
4 Concrete Operators

We will now instantiate the operators sketched out in §2 specifically for the grammar induction task.

Throughout, we repeatedly employ single steps of Viterbi training to transfer information between subnetworks in a model-independent way: when a module's output is a set of (Viterbi) parse trees, it necessarily contains sufficient information required to estimate an arbitrarily-factored model down-stream.[6]

4.1 Transform #1: A Simple Filter

Given a model that was estimated from (and therefore parses) a data set $\mathcal{D}$, the simple filter (F) attempts to extract a cleaner model, based on the simpler complete sentences of $\mathcal{D}_{\text{simp}}$. It is implemented as a single (unlexicalized) step of Viterbi training:

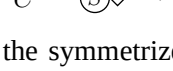


(7)

The idea here is to focus on sentences that are not too complicated yet grammatical. This punctuation-sensitive heuristic may steer a learner towards easy but representative training text and, we showed, aids grammar induction (Spitkovsky et al., 2012b, §7.1).

4.2 Transform #2: A Symmetrizer

The symmetrizer (S) reduces input models to sets of word association scores. It blurs all details of induced parses in a data set $\mathcal{D}$, except the number of times each (ordered) word pair participates in a dependency relation. We implemented symmetrization also as a single unlexicalized Viterbi training step, but now with proposed parse trees' scores, for a sentence in $\mathcal{D}$, proportional to a product over non-root dependency arcs of one plus how often the left and right tokens (are expected to) appear connected:



(8)

The idea behind the symmetrizer is to glean information from skeleton parses. Grammar inducers can sometimes make good progress in resolving undirected parse structures despite being wrong about the polarities of most arcs (Spitkovsky et al., 2009, Figure 3: Uninformed). Symmetrization offers an extra chance to make heads or tails of syntactic relations, after learning which words tend to go together.

At each instance where a word ⓐ attaches ⓩ on (say) the right, our implementation attributes half its weight to the intended construction, ⓐ⁀ⓩ, reserving the other half for the symmetric structure, ⓩ attaching ⓐ to its left: ⓐ⁀ⓩ. For the desired effect, these aggregated counts are left unnormalized, while all other counts (of word fertilities and sentence roots) get discarded. To see why we don't turn word attachment scores into probabilities, consider sentences ⓐ ⓩ and ⓒ ⓩ. The fact that ⓩ co-occurs with ⓐ introduces an asymmetry into ⓩ's relation with ⓒ: $\mathbb{P}(\text{ⓩ} \mid \text{ⓒ}) = 1$ differs from $\mathbb{P}(\text{ⓒ} \mid \text{ⓩ}) = 1/2$. Normalizing might force the interpretation ⓒ⁀ⓩ (and also ⓐ⁀ⓩ), not because there is evidence in the data, but as a side-effect of a model's head-driven nature (i.e., factored with dependents conditioned on heads). Always branching right would be a mistake, however, for example if ⓩ is a noun, since either of ⓐ or ⓒ could be a determiner, with the other a verb.

4.3 Join: A Combiner

The combiner must admit arbitrary inputs, including models not estimated from $\mathcal{D}$, unlike the transforms. Consequently, as a preliminary step, we convert each input C_i into parse trees of $\mathcal{D}$, with counts C_i', via Viterbi-decoding with a smoothed, unlexicalized version of the corresponding incoming model. Actual combination is then performed in a more precise (unsmoothed) fashion: C_i^* are the (lexicalized) solutions starting from C_i'; and C_+^* is initialized with their sum, $\sum_i C_i^*$. Counts of the lexicalized model with lowest cross-entropy on $\mathcal{D}$ become the output:[7]

(9)

5 Basic Networks

We are ready to propose a non-trivial subnetwork for grammar induction, based on the transform and join operators, which we will reuse in larger networks.

5.1 Fork/Join (FJ)

Given a model that parses a base data set $\mathcal{D}_0$, the fork/join subnetwork will output an adaptation of that model for $\mathcal{D}$. It could facilitate a grammar induction process, e.g., by advancing it from smaller

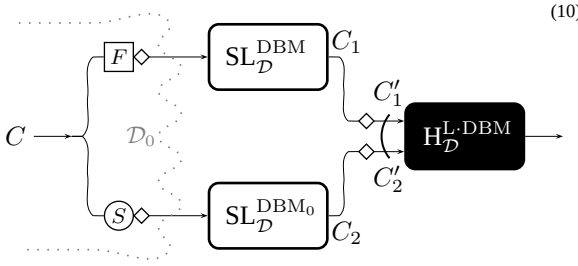
[6] A related approach — initializing EM training with an M-step — was advocated by Klein and Manning (2004, §3).

[7] In our diagrams, lexicalized modules are shaded black.

to larger — or possibly more complex — data sets.

We first fork off two variations of the incoming model based on $\mathcal{D}_0$: (i) a filtered view, which focuses on cleaner, simpler data (transform #1); and (ii) a symmetrized view that backs off to word associations (transform #2). Next is grammar induction over $\mathcal{D}$. We optimize a full DBM instance starting from the first fork, and bootstrap a reduced DBM_0 from the second. Finally, the two new induced sets of parse trees, for $\mathcal{D}$, are merged (lexicalized join):

(10)



The idea here is to prepare for two scenarios: an incoming grammar that is either good or bad for $\mathcal{D}$. If the model is good, DBM should be able to hang on to it and make improvements. But if it is bad, DBM could get stuck fitting noise, whereas DBM_0 might be more likely to ramp up to a good alternative. Since we can't know ahead of time which is the true case, we pursue both optimization paths simultaneously and let a combiner later decide for us.

Note that the forks start (and end) optimizing with soft EM. This is because soft EM integrates previously unseen tokens into new grammars better than hard EM, as evidenced by our failed attempt to reproduce the "baby steps" strategy with Viterbi training (Spitkovsky et al., 2010, Figure 4). A combiner then executes hard EM, and since outputs of transforms are trees, the end-to-end process is a chain of lateen alternations that starts and ends with hard EM.

We will use a "grammar inductor" to represent subnetworks that transition from $\mathcal{D}^l_{\text{split}}$ to $\mathcal{D}^{l+1}_{\text{split}}$, by taking transformed parse trees of inter-punctuation fragments up to length l (base data set, $\mathcal{D}_0$) to initialize training over fragments up to length $l+1$:

$$C \longrightarrow \overset{l+1}{\sim\!\sim\!\sim} \tag{11}$$

The FJ network instantiates a grammar inductor with $l = 14$, thus training on inter-punctuation fragments up to length 15, as in previous work, starting from an empty set of counts, $C = \emptyset$. Smoothing causes initial parse trees to be chosen uniformly at random, as suggested by Cohen and Smith (2010):

$$\emptyset \longrightarrow \overset{15}{\sim\!\sim\!\sim} \tag{12}$$

5.2 Iterated Fork/Join (IFJ)

Our second network daisy-chains grammar inductors, starting from the single-word inter-punctuation fragments in $\mathcal{D}^1_{\text{split}}$, then retraining on $\mathcal{D}^2_{\text{split}}$, and so forth, until finally stopping at $\mathcal{D}^{15}_{\text{split}}$, as before:

$$\overset{1}{\sim\!\sim\!\sim} \bullet \overset{2}{\sim\!\sim\!\sim} \bullet \cdots \bullet \overset{14}{\sim\!\sim\!\sim} \bullet \overset{15}{\sim\!\sim\!\sim} \longrightarrow \tag{13}$$

We diagrammed this system as not taking an input, since the first inductor's output is fully determined by unique parse trees of single-token strings. This iterative approach to optimization is akin to deterministic annealing (Rose, 1998), and is patterned after "baby steps" (Spitkovsky et al., 2009, §4.2).

Unlike the basic FJ, where symmetrization was a no-op (since there were no counts in $C = \emptyset$), IFJ makes use of symmetrizers — e.g., in the third inductor, whose input is based on strings with up to two tokens. Although it should be easy to learn words that go together from very short fragments, extracting correct polarities of their relations could be a challenge: to a large extent, outputs of early inductors may be artifacts of how our generative models factor (see §4.2) or how ties are broken in optimization (Spitkovsky et al., 2012a, Appendix B). We therefore expect symmetrization to be crucial in earlier stages but to weaken any high quality grammars, nearer the end; it will be up to combiners to handle such phase transitions correctly (or gracefully).

5.3 Grounded Iterated Fork/Join (GIFJ)

So far, our networks have been either purely iterative (IFJ) or static (FJ). These two approaches can also be combined, by injecting FJ's solutions into IFJ's more dynamic stream. Our new transition subnetwork will join outputs of grammar inductors that either (i) continue a previous solution (as in IFJ); or (ii) start over from scratch ("grounding" to an FJ):

$$\begin{matrix} C_l \longrightarrow \overset{l+1}{\sim\!\sim\!\sim} \\ \emptyset \longrightarrow \overset{l+1}{\sim\!\sim\!\sim} \end{matrix} \quad \text{H}^{\text{L·DBM}}_{\mathcal{D}^{l+1}_{\text{split}}} \longrightarrow C_{l+1} \tag{14}$$

The full GIFJ network can then be obtained by unrolling the above template from $l = 14$ back to one.

Instance Label	Model	WSJ^{15}_{split} H_{sents}	WSJ^{15}_{split} H_{trees}	WSJ^{15}_{split} DDA	WSJ^{15}_{simp} H_{sents}	WSJ^{15}_{simp} H_{trees}	WSJ^{15}_{simp} DDA	WSJ^{15}_{simp} TA	Description
	DBM	6.54	6.75	83.7	6.05	6.21	85.1	42.7	Supervised (MLE of WSJ^{45})
$\emptyset = C$	—	8.76	10.46	21.4	8.58	10.52	20.7	3.9	Random Projective Parses
$\mathrm{SL}(S(C)) = C_2$	DBM_0	6.18	6.39	57.0	5.90	6.11	57.5	10.4	B } *Unlexicalized*
$\mathrm{SL}(F(C)) = C_1$	DBM	5.89	5.99	62.2	5.79	5.90	60.9	12.0	A } *Baselines*
$\mathrm{H}(C'_2) = C^*_2$	L-DBM	7.28	7.30	59.2	6.87	6.88	58.6	10.4	
$\mathrm{H}(C'_1) = C^*_1$	L-DBM	7.07	7.08	62.3	6.72	6.73	60.8	12.0	} *Baseline*
$C^*_1 + C^*_2 = C_+$	L-DBM	7.20	7.27	64.0	6.82	6.88	62.5	12.3	} *Combination*
$\mathrm{H}(C_+) = C^*_+$	L-DBM	7.02	7.04	64.2	6.64	6.65	62.7	12.8	Fork/Join }
	L-DBM	6.95	6.96	70.5	6.55	6.56	68.2	14.9	Iterated Fork/Join (IFJ)
	L-DBM	6.91	6.92	71.4	6.52	6.52	69.2	15.6	Grounded Iterated Fork/Join
	L-DBM	6.83	6.83	72.3	6.41	6.41	70.2	17.9	Grammar Transformer (GT)
	L-DBM	6.92	6.93	71.9	6.53	6.53	69.8	16.7	IFJ } *w/Iterated*
	L-DBM	6.83	6.83	72.9	6.41	6.41	70.6	18.0	GT } *Combiners*

Table 1: Sentence string and parse tree cross-entropies (in bpt), and accuracies (DDA), on inter-punctuation fragments up to length 15 (WSJ^{15}_{split}) and its subset of simple, complete sentences (WSJ^{15}_{simp}, with exact tree accuracies — TA).

6 Performance of Basic Networks

We compared our three networks' performance on their final training sets, WSJ^{15}_{split} (see Table 1, which also tabulates results for a cleaner subset, WSJ^{15}_{simp}). The first network starts from $C = \emptyset$, helping us establish several straw-man baselines. Its empty initializer corresponds to guessing (projective) parse trees uniformly at random, which has 21.4% accuracy and sentence string cross-entropy of 8.76bpt.

6.1 Fork/Join (FJ)

FJ's symmetrizer yields random parses of WSJ^{14}_{split}, which initialize training of DBM_0. This baseline (B) lowers cross-entropy to 6.18bpt and scores 57.0%. FJ's filter starts from parse trees of WSJ^{14}_{simp} only, and trains up a full DBM. This choice makes a stronger baseline (A), with 5.89bpt cross-entropy, at 62.2%.

The join operator uses counts from A and B, C_1 and C_2, to obtain parse trees whose own counts C'_1 and C'_2 initialize lexicalized training. From each C'_i, an optimizer arrives at C^*_i. Grammars corresponding to these counts have higher cross-entropies, because of vastly larger vocabularies, but also better accuracies: 59.2 and 62.3%. Their mixture C_+ is a simple sum of counts in C^*_1 and C^*_2: it is not expected to be an improvement but happens to be a good move, resulting in a grammar with higher accuracy (64.0%), though not better Viterbi cross-entropy (7.27 falls between 7.08 and 7.30bpt) than both sources. The combiner's third alternative, a locally optimal C^*_+, is then obtained by re-optimizing from C_+. This solution performs slightly better (64.2%) and will be the local optimum returned by FJ's join operator, because it attains the lowest cross-entropy (7.04bpt).

6.2 Iterated Fork/Join (IFJ)

IFJ's iterative approach results in an improvement: 70.5% accuracy and 6.96bpt cross-entropy. To test how much of this performance could be obtained by a simpler iterated network, we experimented with ablated systems that don't fork or join, i.e., our classic "baby steps" schema (chaining together 15 optimizers), using both DBM and DBM_0, with and without a transform in-between. However, all such "linear" networks scored well below 50%. We conclude from these results that an ability to branch out into different promising regions of a solution space, and to merge solutions of varying quality into better models, are important properties of FJ subnetworks.

6.3 Grounded Iterated Fork/Join (GIFJ)

Grounding improves GIFJ's performance further, to 71.4% accuracy and 6.92bpt cross-entropy. This result shows that fresh perspectives from optimizers that start over can make search efforts more fruitful.

7 Enhanced Subnetworks

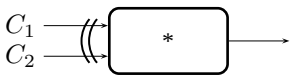
Modularity and abstraction allow for compact representations of complex systems. Another key benefit is that individual components can be understood and improved in isolation, as we will demonstrate next.

7.1 An Iterative Combiner (IC)

Our basic combiner introduced a third option, C^*_+, into a pool of candidate solutions, $\{C^*_1, C^*_2\}$. This new entry may not be a simple mixture of the originals, because of non-linear effects from applying L to $C^*_1 + C^*_2$, but could most likely still be improved.

Rather than stop at C^*_+, when it is better than both originals, we could recombine it with a next best solution, continuing until no further improvement is made. Iterating can't harm a given combiner's cross-entropy (e.g., it lowers FJ's from 7.04 to 7.00bpt), and its advantages can be realized more fully in the larger networks (albeit without any end-to-end guarantees): upgrading all 15 combiners in IFJ would improve performance (slightly) more than grounding (71.5 *vs.* 71.4%), and lower cross-entropy (from 6.96 to 6.93bpt). But this approach is still a bit timid.

A more greedy way is to proceed so long as C^*_+ is not worse than *both* predecessors. We shall now state our most general iterative combiner (IC) algorithm: Start with a solution pool $p = \{C^*_i\}_{i=1}^n$. Next, construct p' by adding $C^*_+ = L(\sum_{i=1}^n C^*_i)$ to p and removing the worst of $n+1$ candidates in the new set. Finally, if $p = p'$, return the best of the solutions in p; otherwise, repeat from $p := p'$. At $n = 2$, one could think of taking $L(C^*_1 + C^*_2)$ as performing a kind of bisection search in some (strange) space. With these new and improved combiners, the IFJ network performs better: 71.9% (up from 70.5 — see Table 1), lowering cross-entropy (down from 6.96 to 6.93bpt). We propose a distinguished notation for the ICs:

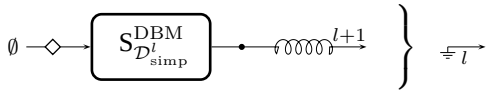


(15)

7.2 A Grammar Transformer (GT)

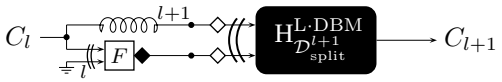
The levels of our systems' performance at grammar induction thus far suggest that the space of possible networks (say, with up to k components) may itself be worth exploring more thoroughly. We leave this exercise to future work, ending with two relatively straight-forward extensions for grounded systems.

Our static bootstrapping mechanism ("ground" of GIFJ) can be improved by pretraining with simple sentences first — as in the curriculum for learning DBM-1 (Spitkovsky et al., 2012b, §7.1), but now with a variable length cut-off l (much lower than the original 45) — instead of starting from $\emptyset$ directly:



(16)

The output of this subnetwork can then be refined, by reconciling it with a previous dynamic solution. We perform a mini-join of a new ground's counts with C_l, using the filter transform (single steps of *lexicalized* Viterbi training on clean, simple data), ahead of the main join (over more training data):

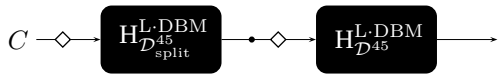


(17)

This template can be unrolled, as before, to obtain our last network (GT), which achieves 72.9% accuracy and 6.83bpt cross-entropy (slightly less accurate with basic combiners, at 72.3% — see Table 1).

8 Full Training and System Combination

All systems that we described so far stop training at $\mathcal{D}^{15}_{\mathrm{split}}$. We will use a two-stage adaptor network to transition their grammars to a full data set, $\mathcal{D}^{45}$:



(18)

The first stage exposes grammar inducers to longer inputs (inter-punctuation fragments with up to 45 tokens); the second stage, at last, reassembles text snippets into actual sentences (also up to $l = 45$).[8]

After full training, our IFJ and GT systems parse Section 23 of WSJ at 62.7 and 63.4% accuracy, better than the previous state-of-the-art (61.2% — see Table 2). To test the generalized IC algorithm, we merged our implementations of these three strong grammar induction pipelines into a combined system (CS). It scored highest: 64.4%.

(19)

The quality of bracketings corresponding to (non-trivial) spans derived by heads of our dependency structures is competitive with the state-of-the-art in unsupervised *constituent* parsing. On the WSJ sentences up to length 40 in Section 23, CS attains similar F_1-measure (54.2 *vs.* 54.6, with higher recall) to

[8] Note that smoothing in the final (unlexicalized) Viterbi step masks the fact that model parts that could not be properly estimated in the first stage (e.g., probabilities of punctuation-crossing arcs) are being initialized to uniform multinomials.

	System		DDA (@10)
	(Gimpel and Smith, 2012)		53.1 (64.3)
	(Gillenwater et al., 2010)		53.3 (64.3)
	(Bisk and Hockenmaier, 2012)		53.3 (71.5)
	(Blunsom and Cohn, 2010)		55.7 (67.7)
	(Tu and Honavar, 2012)		57.0 (71.4)
	(Spitkovsky et al., 2011b)		58.4 (71.4)
	(Spitkovsky et al., 2011c)		59.1 (71.4)
#3	(Spitkovsky et al., 2012a)		61.2 (71.4)
#2	*w/Full Training*	IFJ	62.7 (70.3)
#1	*w/Full Training*	GT	63.4 (70.3)
#1 + #2 + #3	**System Combination**	CS	**64.4** (**72.0**)
	Supervised DBM (also with *loose* decoding)		76.3 (85.4)

Table 2: Directed dependency accuracies (DDA) on Section 23 of WSJ (all sentences and up to length ten) for recent systems, our full networks (IFJ and GT), and three-way combination (CS) with the previous state-of-the-art.

System	F_1	P	R
Binary-Branching Upper Bound	85.7		
Left-Branching Baseline	12.0		
CCM (Klein and Manning, 2002)	33.7		
Right-Branching Baseline	40.7		
F-CCM (Huang et al., 2012)	45.1		
HMM (Ponvert et al., 2011)	46.3		
LLCCM (Golland et al., 2012)	47.6	P	R
CCL (Seginer, 2007)	52.8	54.6	51.1
PRLG (Ponvert et al., 2011)	**54.6**	**60.4**	49.8
CS **System Combination**	**54.2**	55.6	**52.8**
Supervised DBM Skyline	59.3	65.7	54.1
Dependency-Based Upper Bound	87.2	100	77.3

Table 3: Harmonic mean (F_1) of precision (P) and recall (R) for unlabeled constituent bracketings on Section 23 of WSJ (sentences up to length 40) for our combined system (CS), recent state-of-the-art and the baselines.

PRLG (Ponvert et al., 2011), which is the strongest system of which we are aware (see Table 3).[9]

9 Multi-Lingual Evaluation

Last, we checked how our algorithms generalize outside English WSJ, by testing in 23 more set-ups: all 2006/7 CoNLL test sets (Buchholz and Marsi, 2006; Nivre et al., 2007), spanning 19 languages. Most recent work evaluates against this multi-lingual data, with the unrealistic assumption of part-of-speech tags. But since inducing high quality word clusters for many languages would be beyond the scope of our paper, here we too plugged in gold tags for word categories (instead of unsupervised tags, as in §3–8).

We compared to the two strongest systems we knew:[10] MZ (Mareček and Žabokrtský, 2012) and SAJ (Spitkovsky et al., 2012b), which report average accuracies of 40.0 and 42.9% for CoNLL data (see Table 4). Our fully-trained IFJ and GT systems score 40.0 and 47.6%. As before, combining these networks with our own implementation of the best previous state-of-the-art system (SAJ) yields a further improvement, increasing final accuracy to 48.6%.

[9] These numbers differ from Ponvert et al.'s (2011, Table 6) for the full Section 23 because we restricted their `eval-ps.py` script to a maximum length of 40 words, in our evaluation, to match other previous work: Golland et al.'s (2012, Figure 1) for CCM and LLCCM; Huang et al.'s (2012, Table 2) for the rest.

[10] During review, another strong system (Mareček and Straka, 2013, scoring 48.7%) of possible interest to the reader came out, exploiting prior knowledge of stopping probabilities (estimated from large POS-tagged corpora, via reducibility principles).

10 Discussion

CoNLL training sets were intended for comparing supervised systems, and aren't all suitable for unsupervised learning: 12 languages have under 10,000 sentences (with Arabic, Basque, Danish, Greek, Italian, Slovenian, Spanish and Turkish particularly small), compared to WSJ's nearly 50,000. In some treebanks sentences are very short (e.g., Chinese and Japanese, which appear to have been split on punctuation), and in others extremely long (e.g., Arabic). Even gold tags aren't always helpful, as their number is rarely ideal for grammar induction (e.g., 42 *vs.* 200 for English). These factors contribute to high variances of our (and previous) results (see Table 4).

Nevertheless, if we look at the more stable average accuracies, we see a positive trend as we move from a simpler fully-trained system (IFJ, 40.0%), to a more complex system (GT, 47.6%), to system combination (CS, 48.6%). Grounding seems to be more important for the CoNLL sets, possibly because of data sparsity or availability of gold tags.

11 Related Work

The surest way to avoid local optima is to craft an objective that doesn't have them. For example, Wang et al. (2008) demonstrated a convex training method for semi-supervised dependency parsing; Lashkari and Golland (2008) introduced a convex reformulation of likelihood functions for clustering tasks; and Corlett and Penn (2010) designed

Directed Dependency Accuracies (DDA)							
CoNLL Data		MZ	SAJ	IFJ	GT	CS	*(@10)*
Arabic	2006	26.5	10.9	**33.3**	8.3	9.3	(30.2)
	’7	27.9	**44.9**	26.1	25.6	26.8	(45.6)
Basque	’7	26.8	**33.3**	23.5	24.2	24.4	(32.8)
Bulgarian	’7	46.0	**65.2**	35.8	64.2	63.4	(69.1)
Catalan	’7	47.0	62.1	65.0	**68.4**	68.0	(79.2)
Chinese	’6	—	**63.2**	56.0	55.8	58.4	(60.8)
	’7	—	**57.0**	49.0	48.6	52.5	(56.0)
Czech	’6	49.5	**55.1**	44.5	43.9	44.0	(52.3)
	’7	48.0	**54.2**	42.9	24.5	34.3	(51.1)
Danish	’6	**38.6**	22.2	37.8	17.1	21.4	(29.8)
Dutch	’6	44.2	46.6	40.8	**51.3**	48.0	(48.7)
English	’7	49.2	29.6	39.3	57.6	**58.2**	(75.0)
German	’6	44.8	39.1	34.1	54.5	**56.2**	(71.2)
Greek	’6	20.2	26.9	23.7	45.0	**45.4**	(52.2)
Hungarian	’7	51.8	58.2	24.8	52.9	**58.3**	(67.6)
Italian	’7	43.3	40.7	**56.8**	31.1	34.9	(44.9)
Japanese	’6	50.8	22.7	32.6	**63.7**	63.0	(68.9)
Portuguese	’6	50.6	72.4	38.0	72.7	**74.5**	(81.1)
Slovenian	’6	18.1	35.2	42.1	50.8	**50.9**	(57.3)
Spanish	’6	51.9	28.2	57.0	**61.7**	61.4	(73.2)
Swedish	’6	48.2	**50.7**	46.6	48.6	49.7	(62.1)
Turkish	’6	—	**34.4**	28.0	32.9	29.2	(33.2)
	’7	15.7	**44.8**	42.1	41.7	37.9	(42.4)
Average:		40.0	42.9	40.0	47.6	**48.6**	(57.8)

Table 4: Blind evaluation on 2006/7 CoNLL test sets (all sentences) for our full networks (IFJ and GT), previous state-of-the-art systems of Spitkovsky et al. (2012b) and Mareček and Žabokrtský (2012), and three-way combination with SAJ (CS, including results up to length ten).

a search algorithm for encoding decipherment problems that guarantees to quickly converge on optimal solutions. Convexity can be ideal for comparative analyses, by eliminating dependence on initial conditions. But for many NLP tasks, including grammar induction, the most relevant known objective functions are still riddled with local optima. Renewed efforts to find exact solutions (Eisner, 2012; Gormley and Eisner, 2013) may be a good fit for the smaller and simpler, earlier stages of our iterative networks.

Multi-start methods (Solis and Wets, 1981) can recover certain global extrema almost surely (i.e., with probability approaching one). Moreover, random restarts via uniform probability measures can be optimal, in a worst-case-analysis sense, with parallel processing sometimes leading to exponential speed-ups (Hu et al., 1994). This approach is rarely emphasized in NLP literature. For instance, Moore and Quirk (2008) demonstrated consistent, substantial gains from random restarts in statistical machine translation (but also suggested better and faster replacements — see below); Ravi and Knight (2009, §5, Figure 8) found random restarts for EM to be crucial in parts-of-speech disambiguation. However, other reviews are few and generally negative (Kim and Mooney, 2010; Martin-Brualla et al., 2010).

Iterated local search methods (Hoos and Stützle, 2004; Johnson et al., 1988, *inter alia*) escape local basins of attraction by perturbing candidate solutions, without undoing all previous work. “Large-step” moves can come from jittering (Hinton and Roweis, 2003), dithering (Price et al., 2005, Ch. 2) or smoothing (Bhargava and Kondrak, 2009). Non-improving “sideways” moves offer substantial help with hard satisfiability problems (Selman et al., 1992); and injecting non-random noise (Selman et al., 1994), by introducing “uphill” moves via mixtures of random walks and greedy search strategies, does better than random noise alone or simulated annealing (Kirkpatrick et al., 1983). In NLP, Moore and Quirk’s (2008) random walks from previous local optima were faster than uniform sampling and also increased BLEU scores; Elsner and Schudy (2009) showed that local search can outperform greedy solutions for document clustering and chat disentanglement tasks; and Mei et al. (2001) incorporated tabu search (Glover, 1989; Glover and Laguna, 1993, Ch. 3) into HMM training for ASR.

Genetic algorithms are a fusion of what’s best in local search and multi-start methods (Houck et al., 1996), exploiting a problem’s structure to combine valid parts of any partial solutions (Holland, 1975; Goldberg, 1989). Evolutionary heuristics proved useful in the induction of phonotactics (Belz, 1998), text planning (Mellish et al., 1998), factored modeling of morphologically-rich languages (Duh and Kirchhoff, 2004) and plot induction for story generation (McIntyre and Lapata, 2010). Multi-objective genetic algorithms (Fonseca and Fleming, 1993) can handle problems with equally important but conflicting criteria (Stadler, 1988), using Pareto-optimal ensembles. They are especially well-suited to language, which evolves under pressures from competing (e.g., speaker, listener and learner) constraints, and have been used to model configurations of vowels and tone systems (Ke et al., 2003). Our transform and join mechanisms also exhibit some features of genetic search, and make use of competing objec-

tives: good sets of parse trees must make sense both lexicalized and with word categories, to rich and impoverished models of grammar, and for both long, complex sentences and short, simple text fragments.

This selection of text filters is a specialized case of more general "data perturbation" techniques — even cycling over randomly chosen mini-batches that partition a data set helps avoid some local optima (Liang and Klein, 2009). Elidan et al. (2002) suggested how example-reweighing could cause "informed" changes, rather than arbitrary damage, to a hypothesis. Their (adversarial) training scheme guided learning toward improved generalizations, robust against input fluctuations. Language learning has a rich history of reweighing data via (cooperative) "starting small" strategies (Elman, 1993), beginning from simpler or more certain cases. This family of techniques has met with success in semi-supervised named entity classification (Collins and Singer, 1999; Yarowsky, 1995),[11] parts-of-speech induction (Clark, 2000; 2003), and language modeling (Krueger and Dayan, 2009; Bengio et al., 2009), in addition to unsupervised parsing (Spitkovsky et al., 2009; Tu and Honavar, 2011; Cohn et al., 2011).

12 Conclusion

We proposed several simple algorithms for combining grammars and showed their usefulness in merging the outputs of iterative and static grammar induction systems. Unlike conventional system combination methods, e.g., in machine translation (Xiao et al., 2010), ours do not require incoming models to be of similar quality to make improvements. We exploited these properties of the combiners to reconcile grammars induced by different views of data (Blum and Mitchell, 1998). One such view retains just the simple sentences, making it easier to recognize root words. Another splits text into many inter-punctuation fragments, helping learn word associations. The induced dependency trees can themselves also be viewed not only as directed structures but also as skeleton parses, facilitating the recovery of correct polarities for unlabeled dependency arcs.

By reusing templates, as in dynamic Bayesian network (DBN) frameworks (Koller and Friedman, 2009, §6.2.2), we managed to specify relatively "deep" learning architectures without sacrificing (too much) clarity or simplicity. On a still more speculative note, we see two (admittedly, tenuous) connections to human cognition. First, the benefits of not normalizing probabilities, when symmetrizing, might be related to human language processing through the base-rate fallacy (Bar-Hillel, 1980; Kahneman and Tversky, 1982) and the availability heuristic (Chapman, 1967; Tversky and Kahneman, 1973), since people are notoriously bad at probability (Attneave, 1953; Kahneman and Tversky, 1972; Kahneman and Tversky, 1973). And second, intermittent "unlearning" — though perhaps not of the kind that takes place inside of our transforms — is an adaptation that can be essential to cognitive development in general, as evidenced by neuronal pruning in mammals (Craik and Bialystok, 2006; Low and Cheng, 2006). "Forgetful EM" strategies that reset subsets of parameters may thus, possibly, be no less relevant to unsupervised learning than is "partial EM," which only suppresses updates, other EM variants (Neal and Hinton, 1999), or "dropout training" (Hinton et al., 2012; Wang and Manning, 2013), which is important in supervised settings.

Future parsing models, in grammar induction, may benefit by modeling head-dependent relations separately from direction. As frequently employed in tasks like semantic role labeling (Carreras and Màrquez, 2005) and relation extraction (Sun et al., 2011), it may be easier to first establish existence, before trying to understand its nature. Other key next steps may include exploring more intelligent ways of combining systems (Surdeanu and Manning, 2010; Petrov, 2010) and automating the operator discovery process. Furthermore, we are optimistic that both count transforms and model recombination could be usefully incorporated into sampling methods: although symmetrized models may have higher cross-entropies, hence prone to rejection in vanilla MCMC, they could work well as seeds in multi-chain designs; existing algorithms, such as MCMCMC (Geyer, 1991), which switch contents of adjacent chains running at different temperatures, may also benefit from introducing the option to combine solutions, in addition to just swapping them.

[11] The so-called Yarowsky-*cautious* modification of the original algorithm for unsupervised word-sense disambiguation.

Acknowledgments

We thank Yun-Hsuan Sung, for early-stage discussions on ways of extending "baby steps," Elias Ponvert, for sharing all of the relevant experimental results and evaluation scripts from his work with Jason Baldridge and Katrin Erk, and the anonymous reviewers, for their helpful comments on the draft version of this paper. Funded, in part, by Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program, under Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Once again, the first author thanks Moofus.

References

H. Alshawi. 1996. Head automata for speech translation. In *ICSLP*.

F. Attneave. 1953. Psychological probability as a function of experienced frequency. *Experimental Psychology*, 46.

M. Bar-Hillel. 1980. The base-rate fallacy in probability judgments. *Acta Psychologica*, 44.

A. Belz. 1998. Discovering phonotactic finite-state automata by genetic search. In *COLING-ACL*.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. In *ICML*.

A. Bhargava and G. Kondrak. 2009. Multiple word alignment with profile hidden Markov models. In *NAACL-HLT: Student Research and Doctoral Consortium*.

Y. Bisk and J. Hockenmaier. 2012. Simple robust grammar induction with combinatory categorial grammars. In *AAAI*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *CoNLL*.

L. J. Chapman. 1967. Illusory correlation in observational report. *Verbal Learning and Verbal Behavior*, 6.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL-LLL*.

A. Clark. 2003. Combining distributional and morphological information for part of speech induction. In *EACL*.

S. B. Cohen and N. A. Smith. 2010. Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *ACL*.

T. Cohn, P. Blunsom, and S. Goldwater. 2011. Inducing tree-substitution grammars. *JMLR*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *EMNLP*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

E. Corlett and G. Penn. 2010. An exact A* method for deciphering letter-substitution ciphers. In *ACL*.

F. I. M. Craik and E. Bialystok. 2006. Cognition through the lifespan: mechanisms of change. *TRENDS in Cognitive Sciences*, 10.

C. de Marcken. 1995. Lexical heads, phrase structure and the induction of grammar. In *WVLC*.

K. Duh and K. Kirchhoff. 2004. Automatic learning of language model structure. In *COLING*.

J. Eisner. 2012. Grammar induction: Beyond local search. In *ICGI*.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. 2002. Data perturbation for escaping local maxima in learning. In *AAAI*.

J. L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48.

M. Elsner and W. Schudy. 2009. Bounding and comparing methods for correlation clustering beyond ILP. In *NAACL-HLT: Integer Linear Programming for NLP*.

C. M. Fonseca and P. J. Fleming. 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *ICGA*.

C. J. Geyer. 1991. Markov chain Monte Carlo maximum likelihood. In *Interface Symposium*.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. 2010. Posterior sparsity in unsupervised dependency parsing. Technical report, University of Pennsylvania.

K. Gimpel and N. A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *NAACL-HLT*.

F. Glover and M. Laguna. 1993. Tabu search. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications.

F. Glover. 1989. Tabu search — Part I. *ORSA Journal on Computing*, 1.

D. E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.

D. Golland, J. DeNero, and J. Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *EMNLP-CoNLL*.

M. R. Gormley and J. Eisner. 2013. Nonconvex global optimization for latent-variable models. In *ACL*.

W. P. Headden, III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

G. Hinton and S. Roweis. 2003. Stochastic neighbor embedding. In *NIPS*.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. In *ArXiv*.

J. H. Holland. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press.

H. H. Hoos and T. Stützle. 2004. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

C. R. Houck, J. A. Joines, and M. G. Kay. 1996. Comparison of genetic algorithms, random restart, and two-opt switching for solving large location-allocation problems. *Computers & Operations Research*, 23.

X. Hu, R. Shonkwiler, and M. C. Spruill. 1994. Random restarts in global optimization. Technical report, GT.

Y. Huang, M. Zhang, and C. L. Tan. 2012. Improved constituent context model with features. In *PACLIC*.

F. Jelinek and R. L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*.

D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. 1988. How easy is local search? *Journal of Computer and System Sciences*, 37.

D. Kahneman and A. Tversky. 1972. Subjective probability: A judgment of representativeness. *Cognitive Psychology*, 3.

D. Kahneman and A. Tversky. 1973. On the psychology of prediction. *Psychological Review*, 80.

D. Kahneman and A. Tversky. 1982. Evidential impact of base rates. In D. Kahneman, P. Slovic, and A. Tversky, editors, *Judgment under uncertainty: Heuristics and biases*. Cambridge University Press.

J. Ke, M. Ogura, and W. S.-Y. Wang. 2003. Optimization models of sound systems using genetic algorithms. *Computational Linguistics*, 29.

J. Kim and R. J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *COLING*.

S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

K. A. Krueger and P. Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110.

D. Lashkari and P. Golland. 2008. Convex clustering with exemplar-based models. In *NIPS*.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *NAACL-HLT*.

L. K. Low and H.-J. Cheng. 2006. Axon pruning: an essential step underlying the developmental plasticity of neuronal connections. *Royal Society of London Philosophical Transactions Series B*, 361.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

D. Mareček and M. Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *ACL*.

D. Mareček and Z. Žabokrtský. 2011. Gibbs sampling with treeness constraint in unsupervised dependency parsing. In *ROBUS*.

D. Mareček and Z. Žabokrtský. 2012. Exploiting reducibility in unsupervised dependency parsing. In *EMNLP-CoNLL*.

R. Martin-Brualla, E. Alfonseca, M. Pasca, K. Hall, E. Robledo-Arnuncio, and M. Ciaramita. 2010. Instance sense induction from attribute sets. In *COLING*.

N. McIntyre and M. Lapata. 2010. Plot induction and evolutionary search for story generation. In *ACL*.

X.-d. Mei, S.-h. Sun, J.-s. Pan, and T.-Y. Chen. 2001. Optimization of HMM by the tabu search algorithm. In *ROCLING*.

C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell. 1998. Experiments using stochastic search for text planning. In *INLG*.

R. C. Moore and C. Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *COLING*.

T. Naseem and R. Barzilay. 2011. Using semantic cues to learn syntax. In *AAAI*.

R. M. Neal and G. E. Hinton. 1999. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*.

M. A. Paskin. 2001a. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical report, UCB.

M. A. Paskin. 2001b. Grammatical bigrams. In *NIPS*.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.

S. Petrov. 2010. Products of random latent variable grammars. In *NAACL-HLT*.

E. Ponvert, J. Baldridge, and K. Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *ACL-HLT*.

K. V. Price, R. M. Storn, and J. A. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*. Springer.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *ACL-IJCNLP*.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression and related optmization problems. *Proceedings of the IEEE*, 86.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*.

B. Selman, H. Levesque, and D. Mitchell. 1992. A new method for solving hard satisfiability problems. In *AAAI*.

B. Selman, H. A. Kautz, and B. Cohen. 1994. Noise strategies for improving local search. In *AAAI*.

F. J. Solis and R. J.-B. Wets. 1981. Minimization by random search techniques. *Mathematics of Operations Research*, 6.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2009. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *GRLL*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *CoNLL*.

V. I. Spitkovsky, A. X. Chang, H. Alshawi, and D. Jurafsky. 2011c. Unsupervised dependency parsing without gold part-of-speech tags. In *EMNLP*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2012a. Bootstrapping dependency grammar inducers from incomplete sentence fragments via austere models. In *ICGI*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2012b. Three dependency-and-boundary models for grammar induction. In *EMNLP-CoNLL*.

W. Stadler, editor. 1988. *Multicriteria Optimization in Engineering and in the Sciences*. Plenum Press.

A. Sun, R. Grishman, and S. Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *ACL*.

M. Surdeanu and C. D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *NAACL-HLT*.

K. Tu and V. Honavar. 2011. On the utility of curricula in unsupervised learning of probabilistic grammars. In *IJCAI*.

K. Tu and V. Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *EMNLP-CoNLL*.

A. Tversky and D. Kahneman. 1973. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5.

S. I. Wang and C. D. Manning. 2013. Fast dropout training. In *ICML*.

Q. I. Wang, D. Schuurmans, and D. Lin. 2008. Semi-supervised convex training for dependency parsing. In *HLT-ACL*.

T. Xiao, J. Zhu, M. Zhu, and H. Wang. 2010. Boosting-based system combination for machine translation. In *ACL*.

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

Cross-Lingual Discriminative Learning of Sequence Models with Posterior Regularization

Kuzman Ganchev
Google Research
76 9th Avenue
New York, NY 10011
kuzman@google.com

Dipanjan Das
Google Research
76 9th Avenue
New York, NY 10011
dipanjand@google.com

Abstract

We present a framework for cross-lingual transfer of sequence information from a resource-rich source language to a resource-impoverished target language that incorporates soft constraints via posterior regularization. To this end, we use automatically word aligned bitext between the source and target language pair, and learn a discriminative conditional random field model on the target side. Our posterior regularization constraints are derived from simple intuitions about the task at hand and from cross-lingual alignment information. We show improvements over strong baselines for two tasks: part-of-speech tagging and named-entity segmentation.

1 Introduction

Supervised systems for NLP tasks are available for a handful of languages. These systems achieve high accuracy for many applications; a variety of robust algorithms to train them from labeled data have been developed. Here, we focus on learning sequence models for the languages that lack annotated resources. For a given resource-poor target language of interest, we assume that parallel data with a resource-rich source language exists. With the help of this bitext and a supervised system in the source language, we infer constraints over the label distribution in the target language, and train a discriminative model using posterior regularization (Ganchev et al., 2010).

Cross-lingual learning of structured prediction models via parallel data has been applied for several natural language processing problems, including part-of-speech (POS) tagging (Yarowsky and Ngai, 2001), syntactic parsing (Hwa et al., 2005) and named-entity recognition (Kim et al., 2012). These methods are useful in several ways. First, they help in fast prototyping of natural language systems for new languages that do not boast human annotations. Second, the output of such systems could be used to bootstrap more extensive human annotation projects (Vlachos, 2006). Finally, they are significantly more accurate than purely unsupervised systems (McDonald et al., 2011; Das and Petrov, 2011).

Recently, Täckström et al. (2013) presented a technique for coupling token constraints derived from projected cross-lingual information and type constraints derived from noisy tag dictionaries to learn POS taggers. Although this technique resulted in state-of-the-art weakly supervised taggers, the authors used a heuristic to combine the aforementioned two sources of constraints: the dictionary constraints pruned the tagger's search space, and the intersected token-level projections were treated as hard observations. On the other hand, Ganchev et al. (2009) presented a framework for learning weakly-supervised systems (in their case, dependency parsers) that incorporated alignment-based information too, but used the cross-lingual information only as soft constraints, via posterior regularization. The advantage of this framework lay in the fact that the projections were only trusted to a certain degree, determined by a strength hyperparameter, which unfortunately the authors did not have an elegant way to tune. In this paper, we exploit the better aspects of these two lines of work: first, we extend the framework of Täckström et al. by treating the alignment-based projections only as soft constraints (see §3.4); second, we choose the constraint strength by utilizing the tag ambiguity of tokens for a given resource-poor language (see §6.1).

Other than validating our framework on part-of-speech tagging, we experiment on named-entity segmentation in a cross-lingual framework. For this

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1996–2006,
Seattle, Washington, USA, 18-21 October 2013.

task, we present a novel method to perform high-precision phrase-level entity transfer (§5.2.2); we also provide ways to balance precision and recall with posterior regularization (§6.2) by incorporating intuitive soft constraints during learning. We measure performance on standard benchmark datasets for both of these tasks, and report improvements over state-of-the-art baselines.

2 Prior Work

Cross-lingual projection methods can be classified by their use of two very broad ideas. The first idea utilizes parallel data to create full or partial annotations in the low-resource language and trains from this data. This was popularized by Yarowsky and Ngai (2001) who applied this to POS tagging and shallow parsing. It was later applied to parsing (Hwa et al., 2005) and named entity recognition (Kim et al., 2012). The second idea, first proposed by Zeman and Resnik (2008) and applied more broadly by McDonald et al. (2011), is to train a model on a resource-rich language and apply it to a resource-poor language directly. The disparity between the languages is mitigated by the choice of features. In addition to cross-lingual projection, purely unsupervised methods have been explored but with limited success (Christodoulopoulos et al., 2010). Here, we resort to cross-lingual projection and incorporate the first idea; we also follow Li et al. (2012) and use Wiktionary to further constrain the POS tagging task.

Our learning setup is similar to that of Ganchev et al. (2009), who also use posterior regularization but focus on dependency parsing alone. Our work differs with respect to the tasks, the learning algorithm and also in that we use corpus-wide constraints, while Ganchev et al. use one constraint per sentence. For the part-of-speech tagging task, our approach is similar to that of Täckström et al. (2013), who use an almost identical learning setup but only make use of hard constraints. By relaxing these constraints, we allow the model to identify and ignore inconsistently labeled parts of sentences, and achieve better results using identical training and test data.

3 Approach

We give an overview of our approach, and present the details of our model used for cross-lingual learning.

Algorithm 1 Cross-Lingual Learning with Posterior Regularization

Require: Parallel source and target language data $\mathcal{D}^e$ and $\mathcal{D}^f$, source language model $(M)^e$, task-specific target language constraints $\mathcal{C}$.
Ensure: Θ^f, a set of target language parameters.
1: $\mathcal{D}^{e \leftrightarrow f} \leftarrow$ word-align-bitext$(\mathcal{D}^e, \mathcal{D}^f)$
2: $\widehat{\mathcal{D}^e} \leftarrow$ label-supervised$(\mathcal{D}^e)$
3: $\widehat{\mathcal{D}^f} \leftarrow$ project-and-filter-labels$(\mathcal{D}^{e \leftrightarrow f}, \widehat{\mathcal{D}^e})$
4: $\Theta^f \leftarrow$ learn-posterior-constrained$(\widehat{\mathcal{D}^f}, \mathcal{C})$
5: Return Θ^f

3.1 General Overview

The general overview of our framework is provided in Algorithm 1. The process of learning parameters for a target language for a given task involves four subtasks. First, we run word alignment over a large corpus of parallel data between the resource-rich source language and the resource-impoverished target language (see §4.3). In the second step, we use a supervised model to label the source side of the parallel data (see §5.1.1 and §5.2.1). The third step involves a task-specific word-alignment filtering step; this step involves heuristics for which we use cues from prior state-of-the-art (Das and Petrov, 2011; Täckström et al., 2013, see §5.1.2) and also introduce some novel ones for the NE segmentation problem (see §5.2.2). In the fourth step, we train a linear chain conditional random field (Lafferty et al., 2001, CRF henceforth) using posterior regularization. In the next subsection, we turn to a brief summary of this final step of estimating parameters of a discriminative model with posterior regularization.

3.2 Learning with Posterior Regularization

In this work, we utilize discriminative CRF models, and use posterior regularization (PR) to optimize their parameters. As a framework, posterior regularization is described in detail by Ganchev et al. (2010). However in our work, we adopt a different optimization technique; in what follows, we summarize the optimization algorithm in the context of CRF models.

Let $\mathbf{x}$ be an input sentence with a set of possible labelings $\mathcal{Y}(\mathbf{x})$ and let $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$ be a particular labeling for sentence $\mathbf{x}$. We use bold capital letters $\mathbf{X} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ and $\mathbf{Y} = \{\mathbf{y}_1 \dots \mathbf{y}_n\}$ to denote

a corpus of sentences and labelings for the corpus respectively. A CRF models the probability distribution over possible labels for a sentence $p_\theta(\mathbf{y}|\mathbf{x})$ as:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) \propto \exp(\theta \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})) \quad (1)$$

where θ are the model parameters and $\mathbf{f}(.)$ is a feature function. The model examines sentences in isolation, and the probability of a particular labeling for a corpus is defined as a product over the individual sentences:

$$p_\theta(\mathbf{Y} \mid \mathbf{X}) = \prod_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} p_\theta(\mathbf{y} \mid \mathbf{x}). \quad (2)$$

Traditionally, CRF models have been trained to optimize the regularized log-likelihood of the training data

$$\max_\theta \ \mathcal{L}(\theta) = \max_\theta \ \log(p_\theta(\mathbf{Y} \mid \mathbf{X})) - \gamma \, ||\theta|| \quad (3)$$

In our setting, we do not have a fully labeled corpus, but we have constraints on the distribution of labels. For example, we may know that a particular token could be labeled only by a label inventory licensed by a dictionary, or that a labeling projected from a source language is usually (but not always) correct. We define these constraints in terms of feature expectations. Let $q(\mathbf{Y})$ be a distribution over all possible labelings of our corpus $\mathcal{Y}(\mathbf{X})$. Let $\mathcal{Q}$ be a set of distributions defined by:

$$\mathcal{Q} = \{q(\mathbf{Y}) : \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] \leq \mathbf{b}\}, \quad (4)$$

where ϕ is a *constraint* feature function and $\mathbf{b}$ is a vector of non-negative values that serve as upper bounds to the expectations of every constraint feature. The vector $\mathbf{b}$ is used to encode our prior knowledge about desirable distributions $q(\mathbf{Y})$. Note that the constraint features ϕ are not related to the model features $\mathbf{f}$. The model features, together with the model parameters θ define the CRF model; the model features need to be computed at inference time for prediction. By contrast, the constraint features and their corresponding constraint values are used to define our training objective function (and are only used during learning). The PR objective with no labeled data is defined with respect to $\mathcal{Q}$ as:

$$\textbf{PR: } \max_\theta \ \mathcal{J}_\mathcal{Q}(\theta) = \max_\theta \ -\mathbf{KL}(\mathcal{Q}||p_\theta(\mathbf{Y} \mid \mathbf{X})) - \gamma \, ||\theta|| \quad (5)$$

where $\mathbf{KL}(\mathcal{Q}||p) = \min_{q\in\mathcal{Q}} \mathbf{KL}(q||p)$ is the KL-divergence (Kullback and Leibler, 1951) from a set to a point. Note that as we add more constraints, $\mathcal{Q}$ becomes a smaller set. In the limit, $\mathcal{Q} = \{q(\mathbf{Y}) : q(\hat{\mathbf{Y}}) = 1\}$ contains just one distribution concentrated on a single labeling $\hat{\mathbf{Y}}$. In this limit, posterior regularization degenerates into the convex log-likelihood objective normally used for supervised data $\mathcal{J}_\mathcal{Q}(\theta) = \mathcal{L}(\theta)$. However, in the general case, the PR objective $\mathcal{J}_\mathcal{Q}$ is not necessarily convex. Prior work, including that of Ganchev et al. propose an algorithm similar to Expectation-Maximization (Dempster et al., 1977, EM henceforth) to optimize $\mathcal{J}_\mathcal{Q}$, but we follow Liang et al. (2009) in using a schochastic update-based algorithm described below.

Note: To make it easier to reason about constraint values $\mathbf{b}$, we scale constraint features $\phi(\mathbf{X}, \mathbf{Y})$ to lie in $[0, 1]$ by computing $\max_\mathbf{Y} \phi(\mathbf{X}, \mathbf{Y})$ for the corpus to which ϕ is applied.

3.3 Optimization

The optimization procedure proposed by Ganchev et al. is similar to the EM algorithm, and computes the minimization $\min_{q\in\mathcal{Q}} \mathbf{KL}(q||p)$ at each step, using its dual form; this minimization is convex, so there is no duality gap. They show that the optimal primal variables $q^*(\mathbf{Y})$ are related to the optimal dual variables λ^* by:

$$q^*(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X})e^{-\lambda^*\cdot\phi(\mathbf{X},\mathbf{Y})}}{Z(\lambda^*)}. \quad (6)$$

where $Z(\lambda^*)$ is the normalizer. The dual problem is given by:

$$\max_{\lambda\geq 0} -\mathbf{b}\cdot\lambda - \log Z(\lambda). \quad (7)$$

Substituting Eq. 7 into the objective in Eq. 5, we get the saddle-point problem:

$$\max_\theta \ \min_{\lambda\geq 0} \ \mathbf{b}\cdot\lambda + \log \sum_\mathbf{Y} p_\theta(\mathbf{Y}|\mathbf{X})e^{-\lambda^*\cdot\phi(\mathbf{X},\mathbf{Y})} - \gamma \, ||\theta|| \, . \quad (8)$$

To optimize the above objective function, we need to compute partial derivatives with respect to both θ and λ. First, to compute the partial derivatives of Eq. 8

with respect to θ, we need to find expectations of the model features $\mathbf{f}$ given the current distribution p_θ and the constraint distribution q. To perform tractable inference, a linear-chain CRF model assumes that the feature function factorizes according to smaller parts; in particular the factorization uses the following structure:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{f}(\mathbf{x}, y_i, y_{i-1}) \quad (9)$$

where i ranges over the tokens in the sentence. This factorization allows us to efficiently compute expectations over the labels y_i and label-pairs (y_i, y_{i+1}). To compute the partial gradient of Eq. 8 with respect to λ, we need to find the expectations of the constraint features ϕ. In order to be tractable here too, we ensure that ϕ also factorize according to the same structure as $\mathbf{f}$. Therefore, the gradient computation w.r.t. λ turns out to be straightforward.

For all the experiments in this paper, we optimize Eq. 8 using stochastic projected gradient. For each training sentence, we compute the gradient of θ and λ with respect to Eq. 8, take a gradient step in each one, and truncate the negative entries in λ to zero. We use a step size of 1 for all experiments.[1]

3.4 Relationship with Täckström et al. (2013)

In this subsection, we focus briefly on the relationship between this work and the work of Täckström et al. (2013), who focused on constrained learning of POS taggers. Täckström et al. define constrained lattices and train by optimizing marginal conditional log-likelihood. In our notation, they define their objective as:

$$\max_\theta \log \sum_{\mathbf{Y} \in \widehat{\mathcal{Y}}(\mathbf{X})} p_\theta(\mathbf{Y}|\mathbf{X}) - \gamma\|\theta\| \quad (10)$$

where $\widehat{\mathcal{Y}}(\mathbf{X})$ are the constrained lattices of label sequences that agree with both a dictionary and cross-lingually projected POS tags for each sentence of the training corpus. Let us define a constraint feature $\phi(\mathbf{X}, \mathbf{Y})$ which counts the number of tags in $\mathbf{Y}$ which are outside the constraint set $\widehat{\mathcal{Y}}(\mathbf{X})$ and require $\phi(\mathbf{X}, \mathbf{Y}) \leq 0$. Note that,

$$\arg\min_q \mathbf{KL}(q||p_\theta(\mathbf{Y}|\mathbf{X})) \text{ s.t. } \phi(\mathbf{X}, \mathbf{Y}) \leq 0$$

gives the same distribution as Eq. 10. Given this equivalence, it is easy to see that the gradient of Eq. 5 with respect to θ is the same as that of Eq. 10. By using such constrained lattices, Täckström et al. avoid maintaining a parameter for the constraint, but lose the ability to relax the constraint value and allow some probability mass outside the pruned lattice. Their paper also differs from ours in that they use L-BFGS (Liu and Nocedal, 1989), while we use an online optimization procedure. Since the objectives are non-convex, the two optimization techniques could lead to different local optima even when the constraint is not relaxed ($\mathbf{b} = 0$).

4 Tasks and Data

In this section, we focus on the nature of the two tasks that we attempt to solve, describe the source language datasets we use to train our supervised models for transfer, the target language datasets on which we evaluate our models and the parallel data we use for cross-lingual transfer.

4.1 Part-of-Speech Tagging

First, we focus on the task of part-of-speech tagging. Following previous work on cross-lingual POS tagging (Das and Petrov, 2011; Täckström et al., 2013), we adopt the POS tags of Petrov et al. (2012), version 1.03;[2] we use the October 2012 version of Wiktionary[3] as our tag dictionary.

After pruning the search space with the dictionary, we place soft constraints derived by projecting POS tags across word alignments. The alignments are filtered for confidence (see §5.1.2), but we also filter any projected tags that are not licensed by the dictionary. The example in Figure 1 illustrates why this dictionary filtering step is important. Consider the English-Spanish phrase pair from Figure 1, which we observed in our training data. Our supervised tagger correctly tags *Asian* with the ADJ tag as shown in the figure. *Asian* is aligned to the Spanish word *Asia*, which should be tagged NOUN. Because the Spanish Wiktionary only allows the NOUN tag for *Asia*, we do not project the ADJ tag from the English word *Asian*. By contrast, we do project the NOUN tag from the English word *sponges* to the Spanish

[1] Note that we did not implement regularization of θ in the stochastic optimizer, hence our PR objective (Eq. 8) was unregularized; however, the baseline models use ℓ_2 regularization.

[2] http://code.google.com/p/universal-pos-tags

[3] http://meta.wikimedia.org/wiki/Wiktionary

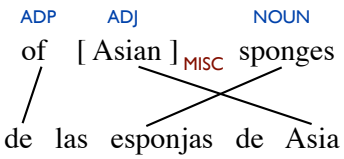


Figure 1: An English (top) – Spanish (bottom) phrase pair from our parallel data. The correct POS tags and NER annotations are shown for the English phrase. Word alignments are shown as links between English and Spanish words.

word *esponjas* because this tag is in our dictionary for the latter word.

For all our POS experiments, we evaluate on seventeen target languages. Fifteen of these languages were part of the experiments conducted by Täckström et al. (2013); we add Arabic and Hungarian to the set. The first column of Table 1 lists all seventeen languages using their two-letter abbreviation codes from the ISO 639-1 standard. The evaluation datasets correspond to the test sets from the CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). For French we use the treebank of Abeillé et al. (2003). English serves as our source language and we use the Penn Treebank (Marcus et al., 1993, with tags mapped to the universal tags) to train our supervised source-side model.

4.2 Named-Entity Segmentation

Second, we investigate the task of named-entity segmentation. The goal of this task is to identify the boundaries of named-entities for a given language without classifying them by type. This is the *unlabeled* version of named-entity recognition, and is more amenable to cross-lingual supervision. To understand why that is, consider again the example from Figure 1. The English supervised NE tagger correctly identifies *Asian* as a named entity of type MISC (miscellaneous). The word-alignments suggest we should transfer this annotation to the Spanish word *Asia* which is also an entity. However, this should be labeled LOC (location) according to the CoNLL annotation guidelines (Tjong Kim Sang and De Meulder, 2003). Because syntactic variations of this kind are common, it makes cross-lingual detection of NE boundaries as well as types hard.[4] In this paper, we focus on named-entity segmentation alone, consider the full NER task out of scope. We use English as a source language and train a supervised English named-entity tagger with the labels in place, using the CoNLL 2003 shared task data (Tjong Kim Sang and De Meulder, 2003). We project the spans using the maximal-span heuristic (Yarowsky and Ngai, 2001). We project into Dutch, German and Spanish and evaluate on the standard CoNLL 2002 and 2003 shared task data sets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003).

4.3 Parallel Data

For both tasks we use parallel data gathered automatically from the web using the method of Uszkoreit et al. (2010), as well as data from Europarl (Koehn, 2005) and the UN parallel corpus (UN, 2006), for languages covered by the latter two corpora. The parallel sentences are word aligned with the aligner of DeNero and Macherey (2011). The size of the parallel corpus is larger than we need for our tasks, so we follow Täckström et al. (2013) in sampling 500k tokens for POS tagging and 10k sentences for named-entity segmentation (see §5.1.2 and §5.2.2).

5 Experimental Details

In this section, we provide details about task-specific implementations of the supervised source-side model and the word-alignment filtering techniques (steps 2 and 3 in Algorithm 1 respectively); we also briefly describe the setup of the cross-lingual experiments for each task.

5.1 Part-of-Speech Tagging

We first focus on the experimental setup for the POS tagging task. When describing feature sets we refer to features conjoined with just a single tag as *emission features* and with consecutive tag pairs as *transition features*.

[4] We tried using English and German gazetteers from the CoNLL 2002 and 2003 shared tasks as a label dictionary similar to the way we use Wiktionary for POS tagging. This did not work well because the CoNLL gazetteers do not have good coverage on our parallel datasets, which we use for training.

5.1.1 Supervised Source-Side Model

We tag the English side of our parallel data with a supervised first-order linear-chain CRF POS tagger. We use standard features for tagging. Our emission features are a bias feature, the current word, its suffixes up to length 3, its capitalization shape, whether it contains a hyphen, digit or punctuation and its cluster identity. Our transition features are a bias feature and the cluster identities of each word in the transition. For the cluster-based features, we use monolingual word clusters induced with the exchange algorithm of Uszkoreit and Brants (2008), which implements the same objective as Brown et al. (1992); these clusters have shown improvements for sequence labeling tasks (Turian et al., 2010; Täckström et al., 2012). We set the number of clusters to 256 for both the source side tagger and all the other languages. On Section 23 of the WSJ section of the Penn Treebank, the source side tagger achieves an accuracy of 96.2%.

5.1.2 Word Alignment Filtering

Following Täckström et al. (2013), we tag the English side of our parallel data using the source-side POS tagger, intersect the word alignments and filter alignments with confidence below 0.95. We sample 500,000 tokens of target side sentences for each language, and use this as training data for learning weakly-supervised taggers.

5.1.3 Setup for Cross-Lingual Experiments

Following Täckström et al. (2013) we use a reduced feature set for the cross-lingual models. The emission features are the same as the supervised model but without the punctuation feature,[5] and we use only the bias transition feature. Because this limits the ability of the model to use context, we also experiment with an extended feature set that has transition features for the clusters of each word in the transition, and their suffixes up to length 3. We refer to the extended-feature models as "BASE+" and "PR+" to distinguish them from the models with fewer features, labeled "BASE" and "PR".

We train BASE and BASE+ using L-BFGS with an ℓ_2 regularization weight of 1 for 100 iterations to reproduce the setup used by Täckström et al. (2013). We have only one constraint feature in our posterior regularization models that fires for the unpruned projected tags on words x_i. This feature controls how often our model trusts a projected tag; we explain how its strength is chosen in §6.1. The PR and PR+ models are trained using the stochastic gradient method described in §3.3.

[5]The dictionary licenses punctuations, only by the '.' tag.

5.2 Named-Entity Segmentation

In this subsection, we turn to the experimental details of the named-entity segmentation system.

5.2.1 Supervised Source-Side Model

To train our supervised source-side NER model, we implemented a linear-chain first order CRF model. Our feature set was inspired by the model of Kazama and Torisawa (2007, §6.1); we used all the local features from their model except the gazetteer features, and added cluster emission features for offsets in the range [-2, 2] and transition features for offsets in the range [-1, 1] as well as a sentence-start feature. We use automatic POS tags for all the experiments.

We use a BIO encoding of the four NER labels (PER, LOC, ORG and MISC). We also experimented with omitting the NE labels from the tagger, still with a BIO encoding for segments, but the results were worse on average than what we report in Table 2. We train the source-side model on the CoNLL 2003 English training set with log-loss using L-BFGS for 100 iterations with ℓ_2 regularization weight of 0.1. The model gets 90.9% and 87.5% labeled F_1 on the CoNLL development and test sets respectively.[6]

5.2.2 Word-Alignment Filtering

Projecting named entities across languages can be error prone for several reasons. Mistakes introduced by the automatic word aligner is one of them. Word alignment errors are particularly problematic for entity mentions because of the garbage collector effect (Brown et al., 1993); due to differences in the word order between languages, a few alignment errors can result in many errors in the other language. Additionally, entities can occur on just one side of the bitext.[7] Another source of error is the automatic

[6]These performance values would place us among the top three competitors of the CoNLL 2003 shared task.

[7]For example, "*It's all Greek to me.*" in one language and "*I don't understand it.*" in another.

labeling on the source side, which is inaccurate if the parallel corpus is out of domain. To mitigate these errors, we aggressively filter the training data for this task. We discard sentence pairs where more than 30% of the source language tokens are unaligned, where any source entities are unaligned or where any source entities are more than 4 tokens long. We also compute a confidence score over entity annotations as the minimum posterior over the tags that comprise the entity and discard sentence pairs that have an entity with confidence below 0.9. Finally, we discard any sentences that contain no projected entities. These filtering steps allow us to keep 7.4%, 9.7% and 10.4% of the aligned sentence pairs for German, Spanish and Dutch, respectively, resulting in very high-precision named-entity projections (see Table 2). For comparison, we also perform experiments without this filtering step.

5.2.3 Setup for Cross-Lingual Experiments

We use a CRF with the same feature set and BIO encoding for the cross-lingual models as the source-side NER model. We compare our approach ("PR" in Table 2) to a baseline ("BASE" in Table 2) which treats the projected annotations as fully observed. The PR model treats the projected NE spans of a sentence as observed, and allows all labels on the remaining tokens. Since the "O" tag is never seen, an unconstrained model would learn to never predict it. We add two features that fire when the current word is tagged "O": a bias feature and a feature that fires when the automatic POS tag is a proper noun. We set up $\mathcal{Q}$ so the desired expectations are at least 0.98 and at most 0.1 for these constraint features respectively.

6 Results

In this section, we turn to our experimental results; first, we focus on POS tagging and then turn to the NE segmentation task.

6.1 Part-of-Speech Tagging

Constraint Strength: As discussed in §4.1, it is important to filter out projected annotations not licensed by Wiktionary. Thus, the quality of weakly-supervised POS taggers learned from projections is closely correlated with the coverage of the Wiktionary. To quantify the effect of Wiktionary coverage, we counted the expected number of possible tags per token (TpT) for our unlabeled corpora. Specifically, for each token, we counted the number of tags licensed by the dictionary, or all tags for word forms not in the dictionary. For each language, we also ran our system with constraint strengths in {0.7, 0.75, 0.8, 0.85, 0.9, 0.92, 0.95, 0.98, 1.00}, and computed the optimal constraint strength from this set. We found that the best constraint strength is closely correlated with the average number of tags available for each token. Figure 2 shows the best constraint strength as a function of the inverse of the number of unpruned tags per token. As observed in the figure, the relationship between the optimal strength and 1/TpT is roughly linear. Figure 2 also shows a linear approximation to the data plotted. When applying this technique to a new language, we would not be able to estimate the optimal constraint strength, but we could use the linear approximation and knowledge of 1/TpT to estimate it. For our experiments below, we perform this estimation for each language using the linear approximation computed from the remaining languages.

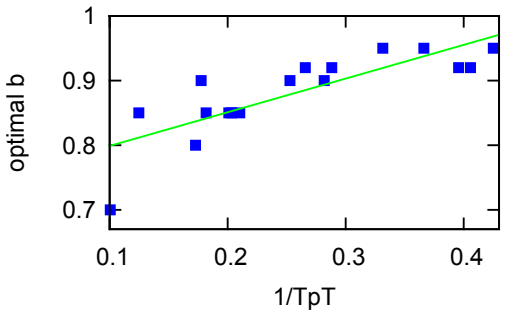


Figure 2: Correlation between optimal constraint value $\mathbf{b}$ and dictionary pruning efficiency. Each blue square is a language, the green line is a linear approximation of the data.

Results: The results for our part-of-speech tagging experiments are in Table 1. We compare our results to BASE, which corresponds to reruns of the best model of Täckström et al. (2013, Column 9 of Table 2), and closely aligns with the numbers reported by the authors. We see in Table 1 that for both feature sets (i.e., with and without the '+' extension), our estimated constraint strength is usually better than using a constraint strength of 1. The results in the column labeled PR are better than BASE for 12 out of 17 languages, and the results for PR+ are

	BASE	BASE+	PR	PR+
ar	37.84	44.96*	49.04*	**50.10***
bg	88.04	87.93	88.02	**88.42***
cs	79.67	80.01*	80.20*	**80.68***
da	88.14	87.92	**88.24***	87.90
de	90.32	89.97	**90.41***	90.29
el	90.03	89.03	**90.63***	90.24*
es	86.99	86.81	87.20*	**87.21***
fr	87.07	**87.53***	87.44*	87.48*
hu	82.05	82.05	82.14*	**83.13***
it	89.48	**89.89***	89.52	89.72*
ja	**80.63**	78.54	80.02	79.68
nl	85.89	85.77	85.59	**85.98***
pt	90.93	**91.60***	91.48*	91.56*
sl	82.46	82.08	83.16*	**83.49***
sv	89.06	88.72	**89.25***	88.77
tr	64.39	65.74*	63.88	**66.47***
zh	73.98	72.82	**74.51***	68.43
Avg	81.59	81.85*	**82.40***	82.33*
-zh-ar	85.01	84.91	85.15*	**85.40***

Table 1: POS tagging results. BASE represents the best model of Täckström et al. (2013). PR is a system with the same features but with relaxed constraints. BASE+ and PR+ add additional model features (see §5.2.3). * indicates improvements over the previous state of the art (BASE), and bold values indicate the best score for a language. "Avg" indicates averaged results for all 17 languages, while "-zh-ar" shows averaged results without Chinese and Arabic.

better than BASE+ for 13 out of 17 languages. Additionally, adding features does not tend to help the baseline model to a large extent (the wins are for 6 languages), but does tend to help the PR model (for 11 languages); however, there is a large drop in performance for Chinese.

Error Analysis: Here, we analyze the nature of improvements that the PR models get. For the languages where PR results in large improvements, it stems from the ability to allow the sentential context to sometimes override the tag projected via the parallel data. For example, the Czech word *se* can either be a reflexive pronoun (such as *ourselves* in English) or translate to the preposition *with*. The pronominal sense comprises about 95% of occurrences in the Czech annotations, but it would not appear in an English translation. For example, the phrase "*podívali jsme se*" translates to "*we looked*", and the word *jsme* would typically be aligned to *we*; *se*, which serves as a reflexive pronoun here, remains unaligned. Consequently, in our data, over 7000 occurrences of *se* appear, but only 17 instances have a tag projection that is not filtered by Wiktionary. Since the remaining are tagged with the preposition tag, the hard-constrained baseline always tags *se* as a preposition. By contrast, the soft-constrained PR model predicts the pronominal sense in cases where the context is most indicative of a pronoun – 38% of the time. It still mistags many of the pronominal cases where the contextual evidence is not strong enough. We get very similar behavior with the Hungarian word *hogy* which can translate to the conjunction *that* (as in "*I see that you are here*") or the adverb *how*.

We found that the drastic drop in performance for Chinese under the PR+ model is due to the possessive marker "的" which serves exclusively as a particle in the test data. Wiktionary also allows the noun and adverb tags. The adverbial use is actually a different token (的确 ↔ *really, truly*) containing the same character. Because the cross-lingual training data is based on machine-learned alignments, 99.4% of the training examples of 的 have no annotations, and only 0.6% have the particle annotation projected from the English *'s* possessive marker. If we remove the noun and adverb senses from the Wiktionary performance of PR+ improves to 72.87%. Alternatively, we could add another constraint to prefer closed-class words over open-class words when both are licensed by the dictionary. When we add such a constraint to Chinese with a constraint value of 0.95, we recover most of the loss (68.43 → 72.94); however, we do not report this specific change to the Chinese experimental setup in Table 1 to maintain generality.

6.2 Named-Entity Segmentation

Results: Table 2 shows the results for the named entity segmentation experiments. First, we observe that the word alignment filtering step (§5.2.2) improves results for all three languages by significant margins, for both the BASE and PR models. Both with and without filtering, we observe that the baseline models are very strongly biased towards precision. The filtering step tends to help with recall more than precison for both models. By having a soft constraint via PR and allowing some segmentations to fall outside of the transferred one, we get an increase in recall,

		No Filtering		Filtering (§5.2.2)	
Lang	Metric	BASE	PR	BASE	PR
de	Prec	74.29	73.85	75.36	**76.47**
	Recall	41.69	54.50	54.71	**64.61**
	F_1	53.41	62.71	63.39	**70.04**
es	Prec	74.53	62.10	**82.50**	70.22
	Recall	56.39	78.33	67.27	**81.10**
	F_1	64.20	69.28	74.11	**75.27**
nl	Prec	81.90	75.12	**86.39**	76.09
	Recall	50.54	76.11	65.45	**79.11**
	F_1	62.51	75.61	74.47	**77.57**
Above: dev, below: test					
de	Prec	73.23	71.67	69.90	**70.94**
	Recall	39.70	51.81	52.52	**61.42**
	F_1	51.49	60.14	59.97	**65.84**
es	Prec	75.38	65.40	**83.50**	73.68
	Recall	56.00	80.30	67.55	**83.31**
	F_1	64.26	72.09	74.68	**78.20**
nl	Prec	79.45	73.55	**86.01**	77.05
	Recall	47.45	75.37	65.16	**80.11**
	F_1	59.42	74.45	74.14	**78.55**

Table 2: Result for the named-entity segmentation experiments. The highest score in each category is shown in bold. Note that "No Filtering" still discards sentences with no projected entities.

and in turn an improved F_1 score. On average the PR model improves F-score by 3.6% on the development set and 4.6% on the test set over the baseline (when filtering is used). Note that because we focus on named entity segmentation, our results are not directly comparable to those of Täckström (2012), who train a de-lexicalized named entity recognizer on one language and apply it to other languages.

Error Analysis: In order to get a sense for the types of errors made by the baseline which are corrected by the PR model, we collected statistics about the most frequent errors in the segments extracted by the baseline and by our model. We divided the errors into missing segments, extraneous segments and overlapping segments.

From Table 2, it is clear that the most common errors for the baseline models are missing entities. From our analysis of the CoNLL development data, we found that the entities that occur with little context (such as the location and publisher of an item) at the onset of news articles are most frequently missed. For German, *dpa* (Deutsche Presse-Agentur) and *Reuter* are the two most common missing segmentations; the Spanish counterparts are *Gobierno* (Government) and *Barcelona*, while for Dutch they are *De Morgen* and *Brussel*. While filtering parallel sentences and using a soft constraint both increase recall, even our strongest model does not get enough information to predict these entities, and they continue to be major sources of error. By contrast, the names mentioned in context are the ones that are most frequently added to the analysis when PR is used. In a sense this is desirable, since a machine-learned named-entity segmentation system is most useful for the long tail of entity mentions.

If we filter the training data and use the PR model to further increase recall, precision errors tend to become relatively more frequent (this trend is observable in Table 2). For German, the most frequent precision error is *Mark* referring to the *Deutsche Mark*. For Spanish, the most frequent precision errors are due to boundary errors. The Spanish annotation guidelines include enclosing quotes as part of the entity name, and failing to include them accounts for just under 1% of the precision errors of the PR system that uses filtering. The second most frequent error is failing to segment *Inter de Milán*. The model segments out either *Inter* or *Milán* or both by themselves depending on context.

7 Conclusions

In this paper, we presented a framework for cross-lingual transfer of sequence information from a resource-rich source language to a resource-poor target language. Our framework incorporates soft constraints while training with projected information via posterior regularization. We presented the efficacy of our framework on two very useful natural language tasks: POS tagging and named-entity segmentation. The soft constraints used in our work model intuitions about a given task. For the POS tagging problem, we designed constraints that also incorporate projected token-level information, and presented a principled method for choosing the extent to which this information should be trusted within the PR framework. This approach generalizes the state of the art in cross-lingual projection work in the context of POS tagging, and improves upon it.

Across seventeen languages, our models outperform the previous state of the art by an average of 0.8% (greater than 4% error reduction), and outperforms it on twelve out of seventeen languages. For named-entity segmentation, our model results in 3.6% and 4.6% absolute improvements in F_1-score on our development and test sets respectively, when averaged across three languages.

Acknowledgments

We would like to thank Ryan McDonald, Fernando Pereira, Slav Petrov and Oscar Täckström for numerous discussions on this topic and providing detailed feedback on early drafts of this paper. We are also grateful to the four anonymous reviewers for their valuable comments.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a Treebank for French. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 10. Kluwer.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Meredith J. Goldsmith, Jan Hajic, Robert L. Mercer, and Surya Mohanty. 1993. But dictionaries are data too. In *Proceedings of the Workshop on Human Language Technology*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of EMNLP*.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL-HLT*.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of ACL-HLT*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL-IJCNLP*.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(03):311–325.

Jun'ichi Kazama and Kentaro Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of EMNLP*.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of ACL*.

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

Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of EMNLP-CoNLL*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proceedings of ICML*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC*.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of NAACL-HLT*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Oscar Täckström. 2012. Nudging the envelope of direct transfer methods for multilingual named entity recognition. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of CoNLL*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.

UN. 2006. ODS UN parallel corpus.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-HLT*.

Jakob Uszkoreit, Jay Ponte, Ashok Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of COLING*.

Andreas Vlachos. 2006. Active annotation. *Proceedings of EACL*.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of IJCNLP Workshop: NLP for Less Privileged Languages*.

Author Index

Acree, Brice D. L., 91
Addanki, Karteek, 102
Ageev, Mikhail, 1011
Agichtein, Eugene, 1011
Aguilar, Jacqui, 1643
Alshawi, Hiyan, 1983
Altman, Eitan, 1943
Altun, Yasemin, 1481
Anderson, Andrew J., 1960
Archambeau, Cedric, 233
Artzi, Yoav, 1545, 1914
Asher, Nicholas, 357
Auli, Michael, 1044

Bach, Francis, 233
Bailly, Raphaël, 624
Balasubramanian, Niranjan, 1721
Barbosa, Denilson, 447
Baroni, Marco, 141, 1908, 1960
Basile, Valerio, 1422
Batchelor, Colin, 747
Batra, Dhruv, 1100
Beckley, Russell, 1584
Beloucif, Meriem, 102
Benamara, Farah, 357
Benedí, José-Miguel, 244
Berant, Jonathan, 1533, 1710
Berg-Kirkpatrick, Taylor, 874
Bernardi, Raffaella, 769
Bethard, Steven, 821
Bhosale, Shruti, 1851
Biemann, Chris, 884
Black, Alan W, 73
Blanco, Eduardo, 1235
Blunsom, Phil, 345, 1700
Bonial, Claire, 1438
Bordes, Antoine, 1366
Bordignon, Ulisse, 1960
Bos, Johan, 1422
Botha, Jan A., 345
Bouamor, Dhouha, 479
Bouchard, Guillaume, 233
Bruni, Elia, 1960
Burges, Christopher J.C., 55, 193
Busa-Fekete, Róbert, 1926

Cadilhac, Anais, 357
Callison-Burch, Chris, 590
Canny, John, 1898
Cardie, Claire, 1933
Carreras, Xavier, 624
Casacuberta, Francisco, 244
Castellanos, Malu, 1655
Cer, Daniel, 1393
Chahuneau, Victor, 1677
Chai, Joyce Y., 392
Chakrabarti, Soumen, 436
Chambers, Nathanael, 1797
Chang, Baobao, 1
Chang, Kai-Wei, 601, 1602
Chang, Yin-Wen, 210
Charniak, Eugene, 1433, 1765
Chen, Bin, 12
Chen, Chen, 1360
Chen, Enhong, 935
Chen, Hanyang, 647
Chen, Hongbo, 1741
Chen, Huan, 946
Chen, Jianfu, 1246
Chen, Wenliang, 1303
Chen, Xilun, 1055
Chen, Yidong, 535
Chen, Zhiyuan, 1655
Cheng, Xiao, 1787
Cherry, Colin, 1948
Chiang, David, 1387

Chikayama, Takashi, 1372
Chiticariu, Laura, 827
Choe, Do Kook, 1433
Choi, Eunsol, 1545
Choi, Yejin, 1246, 1443, 1753
Chou, Andrew, 1533
Chrupała, Grzegorz, 1422
Chuang, Jason, 1631
Cimiano, Philipp, 1732
Ciot, Morgane, 1136
Clark, Peter, 590, 1710
Clark, Stephen, 1427
Cmejrek, Martin, 545
Cohn, Trevor, 977
Collins, Michael, 210, 1574
Conrad, Stefan, 1828
Cross, James, 758
Cui, Lei, 1055
Curran, James R., 989

Damani, Om P., 163
Das, Dipanjan, 1996
Daumé III, Hal, 1077, 1455
Daxenberger, Johannes, 578
de Marneffe, Marie-Catherine, 1625
De Saeger, Stijn, 693
De Silva, Lalindra, 704
DeNero, John, 636
Devlin, Jacob, 556
Diao, Qiming, 1869
Ding, Xiao, 468
Ding, Yang, 1563
Dobnik, Simon, 747
Dogruoz, A. Seza, 857
Dou, Qing, 1668
Dou, Zhicheng, 468
Dridan, Rebecca, 1201
Dubuisson, Jimmy, 669
Duh, Kevin, 130
Durrett, Greg, 1971
Dyer, Chris, 73, 1100, 1677

Eckmann, Jean-Pierre, 669
Eisenstein, Jacob, 61, 891
Eisner, Jason, 1455
El-Beze, Marc, 1943
Elliott, Desmond, 1292
Elming, Jakob, 1476
Elsner, Micha, 42
Engonopoulos, Nikos, 1354
Eskander, Ramy, 1032
Ethis, Emmanuel, 1943
Etzioni, Oren, 1721
Evang, Kilian, 1422

Fang, Rui, 392
Faralli, Stefano, 170
Feldman, Naomi, 42
Feng, Shi, 897
Feng, Song, 1753
Feng, Wei, 12
Filippova, Katja, 1481
FitzGerald, Nicholas, 1914
Fossum, Victoria, 1387
Foster, Jennifer, 1158
Foulds, James, 113
Frank, Stella, 30
Frostig, Roy, 1533
Fu, Ruiji, 1224
Fukumizu, Kenji, 613

Gaillard, Julien, 1943
Galley, Michel, 1044, 1948
Ganchev, Kuzman, 1996
Ganjigunte Ashok, Vikas, 1753
Gardent, Claire, 808
Gardner, Matt, 833
Garron, Anderson, 1348
Gatti, Lorenzo, 1259
Ge, Tao, 1
Ghonge, Shweta, 163
Ghosh, Riddhiman, 1655
Gilbert, Nathan, 704
Gimpel, Kevin, 1100
Glass, James, 182
Goldwater, Sharon, 30, 42
González-Rubio, Jesús, 244
Goto, Isao, 845
Gottipati, Swapna, 1858
Gravier, Guillaume, 1314
Gross, Justin H., 91
Gubbins, Joseph, 1405

Guerini, Marco, 1259
Guo, Yuhang, 863
Guo, Yuhong, 152, 1465
Gurevych, Iryna, 578

Habash, Nizar, 1032
Hajishirzi, Hannaneh, 289
Hall, David, 1898
Hall, Keith, 879
Hangyo, Masatsugu, 924
Hara, Kazuo, 613
Harding, Brittany, 1710
Hardmeier, Christian, 380
Hartshorne, Joshua K., 1438
Hashimoto, Chikara, 693
Hashimoto, Kazuma, 1372
Hassan, Ahmed, 1000
Hayashi, Katsuhiko, 1382
He, Ben, 1741
He, He, 1455
He, Xiangnan, 780
He, Zhengyan, 426
He, Zhongjun, 524
Hieber, Felix, 1688
Hirao, Tsutomu, 1515
Hirst, Graeme, 300
Hon, Hsiao-Wuen, 85
Hou, Yufang, 814
Hovy, Dirk, 1411
Hovy, Eduard, 1411
Hsu, Meichun, 1655
Hu, Zhichao, 369
Huang, Lian'en, 726
Huang, Liang, 758, 908, 1112
Huang, Lifu, 726
Huang, Ruihong, 704, 1557
Huang, Xiaoqiu, 903
Huang, Xuanjing, 658, 946
Huang, Zhongqiang, 556
Hüllermeier, Eyke, 1926

Irvine, Ann, 1077

Jain, Siddhanth, 436
Jehl, Laura, 1688
Ji, Yangfeng, 891
Ji, Zongcheng, 535
Jiang, Jing, 1858, 1869
Jin, Yiping, 780
Jindal, Prateek, 1808
Johannsen, Anders, 1476
Jurafsky, Daniel, 1983

Kalchbrenner, Nal, 1700
Kan, Min-Yen, 12, 780
Kang, Jihua, 946
Kang, Jun Seok, 1443
Kartsaklis, Dimitri, 1590
Kawahara, Daisuke, 924, 1213
Kehler, Andrew, 914
Keller, Frank, 30, 1292
Kessler, Wiltrud, 1892
Kiela, Douwe, 1427
Kim, Joo-Kyung, 1625
King, Irwin, 1521
Kisiel, Bryan, 833
Klakow, Dietrich, 24
Klein, Dan, 265, 874, 1898, 1971
Klinger, Roman, 1732
Kloetzer, Julien, 693
Knight, Kevin, 1668
Kolhatkar, Varada, 300
Koller, Alexander, 1354
Kominek, John, 1325
Kong, Fang, 278, 715
Konstas, Ioannis, 989, 1503
Koppel, Moshe, 1449, 1880
Koprinska, Irena, 989
Kuhn, Jonas, 333, 1892
Kummerfeld, Jonathan K., 265
Kurohashi, Sadao, 924, 1213
Kuznetsova, Polina, 1246, 1443
Kwiatkowski, Tom, 1545

Lagun, Dmitry, 1011
Lapata, Mirella, 415, 1503
Lascarides, Alex, 357
Laws, M. Barton, 1765
Lazaridou, Angeliki, 1908
Le Roux, Joseph, 1158
Le, Dieu-Thu, 769
Lee, Chia-ying, 182
Lerner, Uri, 513

Lewis, Justin, 1710
Lewis, Mike, 681
Li, Baichuan, 1521
Li, Binyang, 897
Li, Chen, 490
Li, Hang, 935
Li, Jiwei, 1933
Li, Mu, 426, 1055
Li, Peng, 567
Li, Sheng, 863
LI, Shoushan, 715
Li, Sujian, 1
Li, Xiaoming, 1337
Li, Yunyao, 827
Liakata, Maria, 747
Liang, Percy, 1170, 1533
Lin, Chin-Yew, 85, 1521
Lin, Edward, 1325
Lin, Ziheng, 12
Ling, Wang, 73
Liu, Bing, 1124, 1655
Liu, Changsong, 392
Liu, Fei, 490
Liu, Jing, 85, 1521
Liu, Qun, 535, 1066
Liu, Shujie, 426, 1055
Liu, Ting, 468, 863, 1224
Liu, Yang, 490, 567, 1492
Lopez de Lacalle, Oier, 415
Lu, Bao-Liang, 845
Lü, Yajuan, 1066
Lu, Zhengdong, 935
Luca, Michael, 1443
Luque, Franco M., 624
Lyu, Michael R., 1521

Ma, Tengfei, 736
Mairal, Julien, 233
Makazhanov, Aibek, 1022
Makhambetov, Olzhas, 1022
Manning, Christopher D., 1170, 1393, 1631, 1710
Mansur, Mairgup, 311
Markert, Katja, 814
Martinez, Hector, 1476
Matkarimov, Bakhyt, 1022
Matsumoto, Yuji, 130
Matthies, Franz, 803
Mausam, 1721
McCrae, John Philip, 1732
McDonald, Ryan, 908
Meek, Christopher, 1602
Meng, Fandong, 1066
Mesquita, Filipe, 447
Mi, Haitao, 545, 1112
Mitchell, Margaret, 1643
Mitchell, Tom, 833
Miwa, Makoto, 1372
Miyao, Yusuke, 1180
Mochihashi, Daichi, 1180
Moens, Marie-Francine, 1613
Mohapatra, Hrushikesh, 436
Moldovan, Dan, 1235
Mooney, Raymond, 1851
Moore, Joshua, 55
Mori, Shinsuke, 204
Moriceau, Véronique, 958
Moschitti, Alessandro, 458
Mueller, Thomas, 322
Mukherjee, Arjun, 1655
Munishkina, Larissa, 369
Muzny, Grace, 1417

Nagata, Masaaki, 204, 1382, 1515
Nakagawa, Hiroshi, 736
Navigli, Roberto, 170
Nelakanti, Anil Kumar, 233
Ney, Hermann, 1377
Ng, Andrew, 1631
Ng, Hwee Tou, 278
Ng, Jun-Ping, 12, 780
Ng, Vincent, 1360
Nguyen, Dong, 857
Nishino, Masaaki, 1515
Nivre, Joakim, 380
Noji, Hiroshi, 1180
Norimatsu, Jun-ya, 222

O'Keefe, Tim, 989
Oard, Douglas W., 1270
Oh, Jong-Hoon, 693
Ohtake, Kiyonori, 693
Okumura, Manabu, 1213

Ortíz-Martinez, Daniel, 244
Ott, Myle, 1933

Palmer, Martha, 1438
Pareti, Silvia, 989
Pasca, Marius, 403
Patterson, Gary, 914
Peitz, Stephan, 1377
Perelygin, Alex, 1631
Petrov, Slav, 513
Pichotta, Karl, 636
Poesio, Massimo, 1960
Popescu, Adrian, 479
Potts, Christopher, 1631
Preoţiuc-Pietro, Daniel, 977

Qadir, Ashequl, 704
Qian, Jin, 946
Qian, Tieyun, 1124
Qian, Xian, 1492
Qin, Bing, 468, 863, 1224
Qiu, Minghui, 1858
Qiu, Xipeng, 658
Quattoni, Ariadna, 624
Quirk, Chris, 1044, 1077, 1948

Rahimtoroghi, Elahe, 369
Rajput, Nitendra, 1270
Rambow, Owen, 1032
Rappoport, Ari, 1880
Rasooli, Mohammad Sadegh, 124
Reberšek, Peter, 1399
Rebholz-Schuhmann, Dietrich, 747
Reiss, Frederick R., 827
Renshaw, Erin, 55, 193
Resnik, Philip, 1348
Resnik, Rebecca, 1348
Richardson, Matthew, 193
Riedl, Martin, 884
Rietig, Felix, 1377
Riezler, Stefan, 1688
Riloff, Ellen, 704, 1557
Roark, Brian, 1584
Rojas Barahona, Lina M., 808
Roller, Stephen, 1146
Roth, Benjamin, 24
Roth, Dan, 601, 791, 1787, 1808
Rozenknop, Antoine, 1158
Rozovskaya, Alla, 791
Rush, Alexander, 210
Ruths, Derek, 1136

Sabyrgaliyev, Islam, 1022
Sadrzadeh, Mehrnoosh, 1590
Saerens, Marco, 613
Saers, Markus, 102
Saha, Shyamasree, 747
Samdani, Rajhans, 601
Sankaran, Baskaran, 1089
Sano, Motoki, 693
Sarkar, Anoop, 1089
Sasano, Ryohei, 1213
Scheible, Christian, 669
Schlinger, Eva, 1677
Schmid, Helmut, 322
Schmidek, Jordan, 447
Scholz, Thomas, 1828
Schulte im Walde, Sabine, 1146
Schütze, Hinrich, 322, 669
Schwartz, Roy, 1880
Sébillot, Pascale, 1314
Seeker, Wolfgang, 333
Seidman, Shachar, 1449
Semmar, Nasredine, 479
Setiawan, Hendra, 501
Severyn, Aliaksei, 458
Søgaard, Anders, 803, 1476
Shakhnarovich, Gregory, 1100
Sharafudinov, Anuar, 1022
She, Lanbo, 392
Shen, Libin, 839
Shimbo, Masashi, 130, 613
Siahbani, Maryam, 1089
Sim, Yanchuan, 91, 1858
Simion, Andrei, 1574
Simon, Anca-Roxana, 1314
Smith, Noah A., 91, 1677, 1858
Smyth, Padhraic, 113
Socher, Richard, 1393, 1631
Soderland, Stephen, 1721
Sokokov, Artem, 1688
Sonderegger, Morgan, 1136
Song, Linfeng, 1066

Song, Yang, 426
Spitkovsky, Valentin I., 1983
Sproat, Richard, 879
Srivastava, Shashank, 1411
Steedman, Mark, 681
Stein, Cliff, 1574
Strube, Michael, 814
Su, Jian, 12
Sudoh, Katsuhito, 204, 1382
Sui, Zhifang, 1
Sumita, Eiichro, 845
Sun, Maosong, 567
Sun, Xu, 311
Surve, Prafulla, 704
Suzuki, Ikumi, 613
Suzuki, Jun, 1382
Swanson, Reid, 369
Szarvas, György, 1926

Talukdar, Partha Pratim, 833
Tan, Chew Lim, 12, 1563
Tan, Ming, 851
Tanaka, Toru, 222
Tannier, Xavier, 958
Tetreault, Joel, 124
Thalappillil Scaria, Aju, 1710
Tiedemann, Jörg, 380
Titov, Ivan, 1354
Torisawa, Kentaro, 693
Toutanova, Kristina, 1948
Trancoso, Isabel, 73
Trikalinos, Thomas A, 1765
Tsubaki, Masashi, 130
Tsukada, Hajime, 1382
Tsur, Oren, 1880
Tsuruoka, Yoshimasa, 1372
Turchi, Marco, 1259

Uijlings, Jasper, 769
Usunier, Nicolas, 1366
Utiyama, Masao, 845

Van Durme, Benjamin, 590, 1643
Vaswani, Ashish, 1387
Vecchi, Eva Maria, 141, 1908
Verlic, Mateja, 1399
Villalba, Martin, 1354
Vinicombe, Heath, 1851
Vlachos, Andreas, 1405
Volkova, Svitlana, 1815
Vozila, Paul, 1191
Vulić, Ivan, 1613

Wager, Stefan, 1170
Walker, Marilyn A., 369
Wallace, Byron C., 1765
Wan, Xiaojun, 1840
Wang, Daling, 897
Wang, Di, 869
Wang, Haifeng, 524
Wang, Hao, 935
Wang, Houfeng, 311, 426
Wang, Jinpeng, 1337
Wang, Mengqiu, 1170, 1710
Wang, Quan, 85
Wang, Rui, 845
Wang, Shanshan, 1281
Wang, Shaojun, 535, 851
Wang, Sida, 1170
Wang, Wei, 903
Wang, William Yang, 869, 1325
Wang, Zhongqing, 715
Wei, Haitian, 1337
Weld, Daniel S., 289, 1776
Wen, Ji-rong, 468
Weng, Fuliang, 490
Weston, Jason, 1366
White, Jerome, 1270
Wilson, Ira B., 1765
Wilson, Theresa, 1643, 1815
Wong, Kam-Fai, 897
Wood, Frank, 42
Wu, Dekai, 102
Wu, Hua, 524
Wu, Jean, 1631
Wuebker, Joern, 1377

Xia, Tian, 535, 851
Xiang, Bing, 501
Xiao, Jianguo, 1840
Xiao, Min, 152, 1465
Xiao, Xinyan, 255
Xie, Jun, 1066

Xiong, Chenyan, 869
Xiong, Deyi, 255, 1563
Xu, Hua, 903
Xu, Shize, 1281
Xu, Tianyu, 647

Yakhnenko, Oksana, 1366
Yamamoto, Mikio, 222
Yan, Hongfei, 1337
Yang, Fan, 1191
Yang, Yi, 61
Yao, Xuchen, 590
Yarowsky, David, 1815
Yasuda, Norihito, 1515
Yasuhara, Makoto, 222
Yessenbayev, Zhandos, 1022
Yih, Wen-tau, 55, 1602
Yoshida, Yasuhisa, 1515
Yu, Ge, 897
Yu, Heng, 1112

Zalk, Marion, 1270
Zamparelli, Roberto, 141
Zbib, Rabih, 556
Zettlemoyer, Luke, 289, 1417, 1545, 1914
Zhai, Shaodan, 535
Zhang, Congle, 1776
Zhang, Dongdong, 1055
Zhang, Hao, 908
Zhang, Le, 897
Zhang, Longkai, 311
Zhang, Min, 1303, 1563
Zhang, Qi, 946
Zhang, Yan, 1281
Zhang, Yu, 182
Zhang, Yue, 1303
Zhao, Feipeng, 152
Zhao, Hai, 845
Zhao, Jiayi, 658
Zhao, Kai, 758, 908, 1112
Zhao, Tiejun, 524
Zhao, Wayne Xin, 1337
Zhao, Yinggong, 1387
Zheng, Xiaoqing, 647
Zhou, Bowen, 501, 545, 839, 851
Zhou, Guodong, 715, 968
Zhou, Ming, 426, 1055
Zhou, Xinjie, 1840
Zhu, Conghui, 524
Zhu, Qiaoming, 968
Zhu, Xiaoning, 524
Zilles, Leila, 289
Zinsmeister, Heike, 300
Zou, Bowei, 968
Zou, Will Y., 1393
Zweig, Geoffrey, 1044
Zweigenbaum, Pierre, 479