ACL-08: HLT

46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies

Short Papers

June 16–17, 2008
The Ohio State University
Columbus, Ohio, USA

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

Order copies of this and other ACL proceedings from:

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

Table of Contents

Language Dynamics and Capitalization using Maximum Entropy
Fernando Batista, Nuno Mamede and Isabel Trancoso 1

Surprising Parser Actions and Reading Difficulty
Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl and Shravan Vasishth 5

Improving the Performance of the Random Walk Model for Answering Complex Questions
Yllias Chali and Shafiq Joty 9

Dimensions of Subjectivity in Natural Language
Wei Chen 13

Extractive Summaries for Educational Science Content
Sebastian de la Chica, Faisal Ahmad, James H. Martin and Tamara Sumner 17

Dialect Classification for Online Podcasts Fusing Acoustic and Language Based Structural and Semantic Information
Rahul Chitturi and John Hansen 21

The Complexity of Phrase Alignment Problems
John DeNero and Dan Klein 25

Novel Semantic Features for Verb Sense Disambiguation
Dmitriy Dligach and Martha Palmer 29

Icelandic Data Driven Part of Speech Tagging
Mark Dredze and Joel Wallenberg 33

Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking
Kevin Duh and Katrin Kirchhoff 37

Coreference-inspired Coherence Modeling
Micha Elsner and Eugene Charniak 41

Enforcing Transitivity in Coreference Resolution
Jenny Rose Finkel and Christopher D. Manning 45

Simulating the Behaviour of Older versus Younger Users when Interacting with Spoken Dialogue Systems
Kallirroi Georgila, Maria Wolters and Johanna Moore 49

Active Sample Selection for Named Entity Transliteration
Dan Goldwasser and Dan Roth 53

Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation
Nizar Habash 57

Combined One Sense Disambiguation of Abbreviations
Yaakov HaCohen-Kerner, Ariel Kass and Ariel Peretz 61

Assessing the Costs of Sampling Methods in Active Learning for Annotation
Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll and McClanahan Peter 65

Blog Categorization Exploiting Domain Dictionary and Dynamically Estimated Domains of Unknown Words
Chikara Hashimoto and Sadao Kurohashi 69

Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management
James Henderson and Oliver Lemon 73

Recent Improvements in the CMU Large Scale Chinese-English SMT System
Almut Silja Hildebrand, Kay Rottmann, Mohamed Noamany, Quin Gao, Sanjika Hewavitharana, Nguyen Bach and Stephan Vogel 77

Machine Translation System Combination using ITG-based Alignments
Damianos Karakos, Jason Eisner, Sanjeev Khudanpur and Markus Dreyer 81

Dictionary Definitions based Homograph Identification using a Generative Hierarchical Model
Anagha Kulkarni and Jamie Callan 85

A Novel Feature-based Approach to Chinese Entity Relation Extraction
Wenjie Li, Peng Zhang, Furu Wei, Yuexian Hou and Qin Lu 89

Using Structural Information for Identifying Similar Chinese Characters
Chao-Lin Liu and Jen-Hsiang Lin 93

You've Got Answers: Towards Personalized Models for Predicting Success in Community Question Answering
Yandong Liu and Eugene Agichtein 97

Self-Training for Biomedical Parsing
David McClosky and Eugene Charniak 101

A Unified Syntactic Model for Parsing Fluent and Disfluent Speech
Tim Miller and William Schuler 105

The Good, the Bad, and the Unknown: Morphosyllabic Sentiment Tagging of Unseen Words
Karo Moilanen and Stephen Pulman 109

Kernels on Linguistic Structures for Answer Extraction
Alessandro Moschitti and Silvia Quarteroni 113

Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking
Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab and Cynthia Rudin 117

Using Automatically Transcribed Dialogs to Learn User Models in a Spoken Dialog System
Umar Syed and Jason Williams 121

Robust Extraction of Named Entity Including Unfamiliar Word
Masatoshi Tsuchiya, Shinya Hida and Seiichi Nakagawa 125

In-Browser Summarisation: Generating Elaborative Summaries Biased Towards the Reading Context
Stephen Wan and Cécile Paris 129

Lyric-based Song Sentiment Classification with Sentiment Vector Space Model
Yunqing Xia, Linlin Wang, Kam-Fai Wong and Mingxing Xu 133

Mining Wikipedia Revision Histories for Improving Sentence Compression
Elif Yamangil and Rani Nelken 137

Smoothing a Tera-word Language Model
Deniz Yuret 141

Event Matching Using the Transitive Closure of Dependency Relations
Daniel M. Bikel and Vittorio Castelli 145

A Linguistically Annotated Reordering Model for BTG-based Statistical Machine Translation
Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li 149

Segmentation for English-to-Arabic Statistical Machine Translation
Ibrahim Badr, Rabih Zbib and James Glass 153

Exploiting N-best Hypotheses for SMT Self-Enhancement
Boxing Chen, Min Zhang, Aiti Aw and Haizhou Li 157

Partial Matching Strategy for Phrase-based Statistical Machine Translation
Zhongjun He, Qun Liu and Shouxun Lin 161

Unsupervised Learning of Acoustic Sub-word Units
Balakrishnan Varadarajan, Sanjeev Khudanpur and Emmanuel Dupoux 165

High Frequency Word Entrainment in Spoken Dialogue
Ani Nenkova, Agustín Gravano and Julia Hirschberg 169

Distributed Listening: A Parallel Processing Approach to Automatic Speech Recognition
Yolanda McMillian and Juan Gilbert 173

Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations
Steven Bethard and James H. Martin 177

Evolving New Lexical Association Measures Using Genetic Programming
Jan Šnajder, Bojana Dalbelo Bašić, Saša Petrović and Ivan Sikirić 181

Semantic Types of Some Generic Relation Arguments: Detection and Evaluation
Sophia Katrenko and Pieter Adriaans 185

Mapping between Compositional Semantic Representations and Lexical Semantic Resources: Towards Accurate Deep Semantic Parsing
Sergio Roa, Valia Kordoni and Yi Zhang 189

Query-based Sentence Fusion is Better Defined and Leads to More Preferred Results than Generic Sentence Fusion
Emiel Krahmer, Erwin Marsi and Paul van Pelt 193

Intrinsic vs. Extrinsic Evaluation Measures for Referring Expression Generation
Anja Belz and Albert Gatt 197

Correlation between ROUGE and Human Evaluation of Extractive Meeting Summaries
Feifan Liu and Yang Liu 201

FastSum: Fast and Accurate Query-based Multi-document Summarization
Frank Schilder and Ravikumar Kondadadi 205

Construct State Modification in the Arabic Treebank
Ryan Gabbard and Seth Kulick 209

Unlexicalised Hidden Variable Models of Split Dependency Grammars
Gabriele Antonio Musillo and Paola Merlo 213

Computing Confidence Scores for All Sub Parse Trees
Feng Lin and Fuliang Weng 217

Adapting a WSJ-Trained Parser to Grammatically Noisy Text
Jennifer Foster, Joachim Wagner and Josef van Genabith 221

Enriching Spoken Language Translation with Dialog Acts
Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore and Shrikanth Narayanan 225

Speakers' Intention Prediction Using Statistics of Multi-level Features in a Schedule Management Domain
Donghyun Kim, Hyunjung Lee, Choong-Nyoung Seon, Harksoo Kim and Jungyun Seo 229

Active Learning with Confidence
Mark Dredze and Koby Crammer 233

splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications
Yoav Goldberg and Michael Elhadad 237

Extracting a Representation from Text for Semantic Analysis
Rodney D. Nielsen, Wayne Ward, James H. Martin and Martha Palmer 241

Efficient Processing of Underspecified Discourse Representations
Michaela Regneri, Markus Egg and Alexander Koller 245

Choosing Sense Distinctions for WSD: Psycholinguistic Evidence
Susan Windisch Brown 249

Decompounding query keywords from compounding languages
Enrique Alfonseca, Slaven Bilac and Stefan Pharies 253

Multi-domain Sentiment Classification
Shoushan Li and Chengqing Zong 257

Evaluating Word Prediction: Framing Keystroke Savings
Keith Trnka and Kathleen McCoy 261

Pairwise Document Similarity in Large Collections with MapReduce
Tamer Elsayed, Jimmy Lin and Douglas Oard 265

Text Segmentation with LDA-Based Fisher Kernel
Qi Sun, Runxin Li, Dingsheng Luo and Xihong Wu 269

Conference Program

Monday, June 16, 2008

9:00–9:10 Opening Session

9:10–10:10 Invited Talk: Marc Swerts, *Facial Expressions in Human-Human and Human-Machine Interactions*

10:10–10:40 **Break**

Session 1A: Information Extraction 1

10:40–11:05 *Mining Wiki Resources for Multilingual Named Entity Recognition*
Alexander E. Richman and Patrick Schone

11:05–11:30 *Distributional Identification of Non-Referential Pronouns*
Shane Bergsma, Dekang Lin and Randy Goebel

11:30–11:55 *Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs*
Marius Paşca and Benjamin Van Durme

11:55–12:20 *The Tradeoffs Between Open and Traditional Relation Extraction*
Michele Banko and Oren Etzioni

Session 1B: Language Resources and Evaluation

10:40–11:05 *PDT 2.0 Requirements on a Query Language*
Jiří Mírovský

11:05–11:30 *Task-oriented Evaluation of Syntactic Parsers and Their Representations*
Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki and Jun'ichi Tsujii

11:30–11:55 *MAXSIM: A Maximum Similarity Metric for Machine Translation Evaluation*
Yee Seng Chan and Hwee Tou Ng

11:55–12:20 *Contradictions and Justifications: Extensions to the Textual Entailment Task*
Ellen M. Voorhees

Monday, June 16, 2008 (continued)

Session 1C: Machine Translation 1

10:40–11:05 *Cohesive Phrase-Based Decoding for Statistical Machine Translation*
Colin Cherry

11:05–11:30 *Phrase Table Training for Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair?*
Yonggang Deng, Jia Xu and Yuqing Gao

11:30–11:55 *Measure Word Generation for English-Chinese SMT Systems*
Dongdong Zhang, Mu Li, Nan Duan, Chi-Ho Li and Ming Zhou

11:55–12:20 *Bayesian Learning of Non-Compositional Phrases with Synchronous Parsing*
Hao Zhang, Chris Quirk, Robert C. Moore and Daniel Gildea

Session 1D: Speech Processing

10:40–11:05 *Applying a Grammar-Based Language Model to a Simplified Broadcast-News Transcription Task*
Tobias Kaufmann and Beat Pfister

11:05–11:30 *Automatic Editing in a Back-End Speech-to-Text System*
Maximilian Bisani, Paul Vozila, Olivier Divay and Jeff Adams

11:30–11:55 *Grounded Language Modeling for Automatic Speech Recognition of Sports Video*
Michael Fleischman and Deb Roy

11:55–12:20 *Lexicalized Phonotactic Word Segmentation*
Margaret M. Fleck

12:20–2:00 **Lunch**

Monday, June 16, 2008 (continued)

Session 2A: Information Retrieval 1

2:00–2:25 *A Re-examination of Query Expansion Using Lexical Resources*
Hui Fang

2:25–2:50 *Selecting Query Term Alternations for Web Search by Exploiting Query Contexts*
Guihong Cao, Stephen Robertson and Jian-Yun Nie

2:50–3:15 *Searching Questions by Identifying Question Topic and Question Focus*
Huizhong Duan, Yunbo Cao, Chin-Yew Lin and Yong Yu

Session 2B: Language Generation

2:00–2:25 *Trainable Generation of Big-Five Personality Styles through Data-Driven Parameter Estimation*
François Mairesse and Marilyn Walker

2:25–2:50 *Correcting Misuse of Verb Forms*
John Lee and Stephanie Seneff

2:50–3:15 *Hypertagging: Supertagging for Surface Realization with CCG*
Dominic Espinosa, Michael White and Dennis Mehay

Session 2C: Machine Translation 2

2:00–2:25 *Forest-Based Translation*
Haitao Mi, Liang Huang and Qun Liu

2:25–2:50 *A Discriminative Latent Variable Model for Statistical Machine Translation*
Phil Blunsom, Trevor Cohn and Miles Osborne

2:50–3:15 *Efficient Multi-Pass Decoding for Synchronous Context Free Grammars*
Hao Zhang and Daniel Gildea

Monday, June 16, 2008 (continued)

Session 2D: Semantics 1

2:00–2:25 *Regular Tree Grammars as a Formalism for Scope Underspecification*
Alexander Koller, Michaela Regneri and Stefan Thater

2:25–2:50 *Classification of Semantic Relationships between Nominals Using Pattern Clusters*
Dmitry Davidov and Ari Rappoport

2:50–3:15 *Vector-based Models of Semantic Composition*
Jeff Mitchell and Mirella Lapata

3:15–3:45 **Break**

Session 3A: Information Extraction 2

3:45–4:10 *Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition*
Andrew Arnold, Ramesh Nallapati and William W. Cohen

4:10–4:35 *Refining Event Extraction through Cross-Document Inference*
Heng Ji and Ralph Grishman

4:35–5:00 *Learning Document-Level Semantic Properties from Free-Text Annotations*
S.R.K. Branavan, Harr Chen, Jacob Eisenstein and Regina Barzilay

5:00–5:25 *Automatic Image Annotation Using Auxiliary Text Information*
Yansong Feng and Mirella Lapata

Monday, June 16, 2008 (continued)

Session 3B: Sentiment Analysis

3:45–4:10 *Hedge Classification in Biomedical Texts with a Weakly Supervised Selection of Keywords*
György Szarvas

4:10–4:35 *When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging*
Alina Andreevskaia and Sabine Bergler

4:35–5:00 *A Generic Sentence Trimmer with CRFs*
Tadashi Nomoto

5:00–5:25 *A Joint Model of Text and Aspect Ratings for Sentiment Summarization*
Ivan Titov and Ryan McDonald

Session 3C: Syntax and Parsing 1

3:45–4:10 *Improving Parsing and PP Attachment Performance with Sense Information*
Eneko Agirre, Timothy Baldwin and David Martinez

4:10–4:35 *A Logical Basis for the D Combinator and Normal Form in CCG*
Frederick Hoyt and Jason Baldridge

4:35–5:00 *Parsing Noun Phrase Structure with CCG*
David Vadas and James R. Curran

5:00–5:25 *Sentence Simplification for Semantic Role Labeling*
David Vickrey and Daphne Koller

Monday, June 16, 2008 (continued)

Session 3D: Student Research Workshop

3:45–4:10 *A Supervised Learning Approach to Automatic Synonym Identification Based on Distributional Features*
Masato Hagiwara

4:10–4:35 *An Integraged Architecture for Generating Parenthetical Constructions*
Eva Banik

4:35–5:00 *Inferring Activity Time in News through Event Modeling*
Vladimir Eidelman

5:00–5:25 *Combining Source and Target Language Information for Name Tagging of Machine Translation Output*
Shasha Liao

5:25–5:50 *A Re-examination on Features in Regression Based Approach to Automatic MT Evaluation*
Shuqi Sun, Yin Chen and Jufeng Li

5:25–6:00 **Break**

6:00–8:30 **Poster and Demo Session**

Long Paper Posters

Summarizing Emails with Conversational Cohesion and Subjectivity
Giuseppe Carenini, Raymond T. Ng and Xiaodong Zhou

Ad Hoc Treebank Structures
Markus Dickinson

A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing
Yoav Goldberg and Reut Tsarfaty

Which Words Are Hard to Recognize? Prosodic, Lexical, and Disfluency Factors that Increase ASR Error Rates
Sharon Goldwater, Dan Jurafsky and Christopher D. Manning

Name Translation in Statistical Machine Translation - Learning When to Transliterate
Ulf Hermjakob, Kevin Knight and Hal Daumé III

Monday, June 16, 2008 (continued)

Using Adaptor Grammars to Identify Synergies in the Unsupervised Acquisition of Linguistic Structure
Mark Johnson

Inducing Gazetteers for Named Entity Recognition by Large-Scale Clustering of Dependency Relations
Jun'ichi Kazama and Kentaro Torisawa

Evaluating Roget's Thesauri
Alistair Kennedy and Stan Szpakowicz

Unsupervised Translation Induction for Chinese Abbreviations using Monolingual Corpora
Zhifei Li and David Yarowsky

Which Are the Best Features for Automatic Verb Classification
Jianguo Li and Chris Brew

Collecting a Why-Question Corpus for Development and Evaluation of an Automatic QA-System
Joanna Mrozinski, Edward Whittaker and Sadaoki Furui

Solving Relational Similarity Problems Using the Web as a Corpus
Preslav Nakov and Marti A. Hearst

Combining Speech Retrieval Results with Generalized Additive Models
J. Scott Olsson and Douglas W. Oard

A Critical Reassessment of Evaluation Baselines for Speech Summarization
Gerald Penn and Xiaodan Zhu

Intensional Summaries as Cooperative Responses in Dialogue: Automation and Evaluation
Joseph Polifroni and Marilyn Walker

Word Clustering and Word Selection Based Feature Reduction for MaxEnt Based Hindi NER
Sujan Kumar Saha, Pabitra Mitra and Sudeshna Sarkar

Combining EM Training and the MDL Principle for an Automatic Verb Classification Incorporating Selectional Preferences
Sabine Schulte im Walde, Christian Hying, Christian Scheible and Helmut Schmid

Randomized Language Models via Perfect Hash Functions
David Talbot and Thorsten Brants

Applying Morphology Generation Models to Machine Translation
Kristina Toutanova, Hisami Suzuki and Achim Ruopp

Multilingual Harvesting of Cross-Cultural Stereotypes
Tony Veale, Yanfen Hao and Guofu Li

Semi-Supervised Convex Training for Dependency Parsing
Qin Iris Wang, Dale Schuurmans and Dekang Lin

Chinese-English Backward Transliteration Assisted with Mining Monolingual Web Pages
Fan Yang, Jun Zhao, Bo Zou, Kang Liu and Feifan Liu

Robustness and Generalization of Role Sets: PropBank vs. VerbNet
Beñat Zapirain, Eneko Agirre and Lluís Màrquez

A Tree Sequence Alignment-based Tree-to-Tree Translation Model
Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan and Sheng Li

Short Paper Posters

Language Dynamics and Capitalization using Maximum Entropy
Fernando Batista, Nuno Mamede and Isabel Trancoso

Surprising Parser Actions and Reading Difficulty
Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl and Shravan Vasishth

Improving the Performance of the Random Walk Model for Answering Complex Questions
Yllias Chali and Shafiq Joty

Dimensions of Subjectivity in Natural Language
Wei Chen

Extractive Summaries for Educational Science Content
Sebastian de la Chica, Faisal Ahmad, James H. Martin and Tamara Sumner

Dialect Classification for Online Podcasts Fusing Acoustic and Language Based Structural and Semantic Information
Rahul Chitturi and John Hansen

Monday, June 16, 2008 (continued)

The Complexity of Phrase Alignment Problems
John DeNero and Dan Klein

Novel Semantic Features for Verb Sense Disambiguation
Dmitriy Dligach and Martha Palmer

Icelandic Data Driven Part of Speech Tagging
Mark Dredze and Joel Wallenberg

Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking
Kevin Duh and Katrin Kirchhoff

Coreference-inspired Coherence Modeling
Micha Elsner and Eugene Charniak

Enforcing Transitivity in Coreference Resolution
Jenny Rose Finkel and Christopher D. Manning

Simulating the Behaviour of Older versus Younger Users when Interacting with Spoken Dialogue Systems
Kallirroi Georgila, Maria Wolters and Johanna Moore

Active Sample Selection for Named Entity Transliteration
Dan Goldwasser and Dan Roth

Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation
Nizar Habash

Combined One Sense Disambiguation of Abbreviations
Yaakov HaCohen-Kerner, Ariel Kass and Ariel Peretz

Assessing the Costs of Sampling Methods in Active Learning for Annotation
Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll and McClanahan Peter

Blog Categorization Exploiting Domain Dictionary and Dynamically Estimated Domains of Unknown Words
Chikara Hashimoto and Sadao Kurohashi

Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management
James Henderson and Oliver Lemon

Recent Improvements in the CMU Large Scale Chinese-English SMT System
Almut Silja Hildebrand, Kay Rottmann, Mohamed Noamany, Quin Gao, Sanjika Hewavitharana, Nguyen Bach and Stephan Vogel

Machine Translation System Combination using ITG-based Alignments
Damianos Karakos, Jason Eisner, Sanjeev Khudanpur and Markus Dreyer

Dictionary Definitions based Homograph Identification using a Generative Hierarchical Model
Anagha Kulkarni and Jamie Callan

A Novel Feature-based Approach to Chinese Entity Relation Extraction
Wenjie Li, Peng Zhang, Furu Wei, Yuexian Hou and Qin Lu

Using Structural Information for Identifying Similar Chinese Characters
Chao-Lin Liu and Jen-Hsiang Lin

You've Got Answers: Towards Personalized Models for Predicting Success in Community Question Answering
Yandong Liu and Eugene Agichtein

Self-Training for Biomedical Parsing
David McClosky and Eugene Charniak

A Unified Syntactic Model for Parsing Fluent and Disfluent Speech
Tim Miller and William Schuler

The Good, the Bad, and the Unknown: Morphosyllabic Sentiment Tagging of Unseen Words
Karo Moilanen and Stephen Pulman

Kernels on Linguistic Structures for Answer Extraction
Alessandro Moschitti and Silvia Quarteroni

Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking
Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab and Cynthia Rudin

Using Automatically Transcribed Dialogs to Learn User Models in a Spoken Dialog System
Umar Syed and Jason Williams

Robust Extraction of Named Entity Including Unfamiliar Word
Masatoshi Tsuchiya, Shinya Hida and Seiichi Nakagawa

In-Browser Summarisation: Generating Elaborative Summaries Biased Towards the Reading Context
Stephen Wan and Cécile Paris

Lyric-based Song Sentiment Classification with Sentiment Vector Space Model
Yunqing Xia, Linlin Wang, Kam-Fai Wong and Mingxing Xu

Mining Wikipedia Revision Histories for Improving Sentence Compression
Elif Yamangil and Rani Nelken

Smoothing a Tera-word Language Model
Deniz Yuret

Student Research Workshop Posters

The Role of Positive Feedback in Intelligent Tutoring Systems
Davide Fossati

Arabic Language Modeling with Finite State Transducers
Ilana Heintz

Impact of Initiative on Collaborative Problem Solving
Cynthia Kersey

An Unsupervised Vector Approach to Biomedical Term Disambiguation: Integrating UMLS and Medline
Bridget McInnes

A Subcategorization Acquisition System for French Verbs
Cédric Messiant

Adaptive Language Modeling for Word Prediction
Keith Trnka

A Hierarchical Approach to Encoding Medical Concepts for Clinical Notes
Yitao Zhang

Demonstrations

Demonstration of a POMDP Voice Dialer
Jason Williams

Generating Research Websites Using Summarisation Techniques
Advaith Siddharthan and Ann Copestake

BART: A Modular Toolkit for Coreference Resolution
Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang and Alessandro Moschitti

Demonstration of the UAM CorpusTool for Text and Image Annotation
Mick O'Donnell

Interactive ASR Error Correction for Touchscreen Devices
David Huggins-Daines and Alexander I. Rudnicky

Yawat: Yet Another Word Alignment Tool
Ulrich Germann

SIDE: The Summarization Integrated Development Environment
Moonyoung Kang, Sourish Chaudhuri, Mahesh Joshi and Carolyn P. Rosé

ModelTalker Voice Recorder—An Interface System for Recording a Corpus of Speech for Synthesis
Debra Yarrington, John Gray, Chris Pennington, H. Timothy Bunnell, Allegra Cornaglia, Jason Lilley, Kyoko Nagao and James Polikoff

The QuALiM Question Answering Demo: Supplementing Answers with Paragraphs drawn from Wikipedia
Michael Kaisser

Tuesday, June 17, 2008

Session: Outstanding Paper Award Presentations

9:00–9:10 Presentation of Awards

9:10–9:35 *Automatic Syllabification with Structured SVMs for Letter-to-Phoneme Conversion*
Susan Bartlett, Grzegorz Kondrak and Colin Cherry

9:35–10:00 *A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model*
Libin Shen, Jinxi Xu and Ralph Weischedel

10:00–10:25 *Forest Reranking: Discriminative Parsing with Non-Local Features*
Liang Huang

10:15–10:30 *Event Matching Using the Transitive Closure of Dependency Relations*
Daniel M. Bikel and Vittorio Castelli

Session 4A: Syntax and Parsing 2

11:10–11:35 *Simple Semi-supervised Dependency Parsing*
Terry Koo, Xavier Carreras and Michael Collins

11:35–12:00 *Optimal k-arization of Synchronous Tree-Adjoining Grammar*
Rebecca Nesson, Giorgio Satta and Stuart M. Shieber

12:00–12:25 *Enhancing Performance of Lexicalised Grammars*
Rebecca Dridan, Valia Kordoni and Jeremy Nicholson

Tuesday, June 17, 2008 (continued)

Session 4B: Dialogue

11:10–11:35 *Assessing Dialog System User Simulation Evaluation Measures Using Human Judges*
Hua Ai and Diane J. Litman

11:35–12:00 *Robust Dialog Management with N-Best Hypotheses Using Dialog Examples and Agenda*
Cheongjae Lee, Sangkeun Jung and Gary Geunbae Lee

12:00–12:25 *Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz Data: Bootstrapping and Evaluation*
Verena Rieser and Oliver Lemon

Session 4C: Machine Learning 2

11:10–11:35 *Phrase Chunking Using Entropy Guided Transformation Learning*
Ruy Luiz Milidiú, Cícero Nogueira dos Santos and Julio C. Duarte

11:35–12:00 *Learning Bigrams from Unigrams*
Xiaojin Zhu, Andrew B. Goldberg, Michael Rabbat and Robert Nowak

12:00–12:25 *Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data*
Jun Suzuki and Hideki Isozaki

Session 4D: Semantics 2

11:10–11:35 *Large Scale Acquisition of Paraphrases for Learning Surface Patterns*
Rahul Bhagat and Deepak Ravichandran

11:35–12:00 *Contextual Preferences*
Idan Szpektor, Ido Dagan, Roy Bar-Haim and Jacob Goldberger

12:00–12:25 *Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions*
Dmitry Davidov and Ari Rappoport

12:25–2:00 **Lunch**

Tuesday, June 17, 2008 (continued)

Session 5A: Short Papers 1 (Machine Translation)

2:00–2:15 *A Linguistically Annotated Reordering Model for BTG-based Statistical Machine Translation*
Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li

2:15–2:30 *Segmentation for English-to-Arabic Statistical Machine Translation*
Ibrahim Badr, Rabih Zbib and James Glass

2:30–2:45 *Exploiting N-best Hypotheses for SMT Self-Enhancement*
Boxing Chen, Min Zhang, Aiti Aw and Haizhou Li

2:45–3:00 *Partial Matching Strategy for Phrase-based Statistical Machine Translation*
Zhongjun He, Qun Liu and Shouxun Lin

Session 5B: Short Papers 2 (Speech)

2:00–2:15 No presentation

2:15–2:30 *Unsupervised Learning of Acoustic Sub-word Units*
Balakrishnan Varadarajan, Sanjeev Khudanpur and Emmanuel Dupoux

2:30–2:45 *High Frequency Word Entrainment in Spoken Dialogue*
Ani Nenkova, Agustín Gravano and Julia Hirschberg

2:45–3:00 *Distributed Listening: A Parallel Processing Approach to Automatic Speech Recognition*
Yolanda McMillian and Juan Gilbert

Tuesday, June 17, 2008 (continued)

Session 5C: Short Papers 3 (Semantics)

2:00–2:15 *Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations*
Steven Bethard and James H. Martin

2:15–2:30 *Evolving New Lexical Association Measures Using Genetic Programming*
Jan Šnajder, Bojana Dalbelo Bašić, Saša Petrović and Ivan Sikirić

2:30–2:45 *Semantic Types of Some Generic Relation Arguments: Detection and Evaluation*
Sophia Katrenko and Pieter Adriaans

2:45–3:00 *Mapping between Compositional Semantic Representations and Lexical Semantic Resources: Towards Accurate Deep Semantic Parsing*
Sergio Roa, Valia Kordoni and Yi Zhang

Session 5D: Short Papers 4 (Generation/Summarization)

2:00–2:15 *Query-based Sentence Fusion is Better Defined and Leads to More Preferred Results than Generic Sentence Fusion*
Emiel Krahmer, Erwin Marsi and Paul van Pelt

2:15–2:30 *Intrinsic vs. Extrinsic Evaluation Measures for Referring Expression Generation*
Anja Belz and Albert Gatt

2:30–2:45 *Correlation between ROUGE and Human Evaluation of Extractive Meeting Summaries*
Feifan Liu and Yang Liu

2:45–3:00 *FastSum: Fast and Accurate Query-based Multi-document Summarization*
Frank Schilder and Ravikumar Kondadadi

3:00–3:15 **Break**

Tuesday, June 17, 2008 (continued)

Session 5E: Short Papers 1 (Syntax)

3:15–3:30 *Construct State Modification in the Arabic Treebank*
Ryan Gabbard and Seth Kulick

3:30–3:45 *Unlexicalised Hidden Variable Models of Split Dependency Grammars*
Gabriele Antonio Musillo and Paola Merlo

3:45–4:00 *Computing Confidence Scores for All Sub Parse Trees*
Feng Lin and Fuliang Weng

4:00–4:15 *Adapting a WSJ-Trained Parser to Grammatically Noisy Text*
Jennifer Foster, Joachim Wagner and Josef van Genabith

Session 5F: Short Papers 2 (Dialog/Statistical Methods)

3:15–3:30 *Enriching Spoken Language Translation with Dialog Acts*
Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore and Shrikanth Narayanan

3:30–3:45 *Speakers' Intention Prediction Using Statistics of Multi-level Features in a Schedule Management Domain*
Donghyun Kim, Hyunjung Lee, Choong-Nyoung Seon, Harksoo Kim and Jungyun Seo

3:45–4:00 *Active Learning with Confidence*
Mark Dredze and Koby Crammer

4:00–4:15 *splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications*
Yoav Goldberg and Michael Elhadad

Tuesday, June 17, 2008 (continued)

Session 5G: Short Papers 3 (Semantics/Phonology)

3:15–3:30 *Extracting a Representation from Text for Semantic Analysis*
Rodney D. Nielsen, Wayne Ward, James H. Martin and Martha Palmer

3:30–3:45 *Efficient Processing of Underspecified Discourse Representations*
Michaela Regneri, Markus Egg and Alexander Koller

3:45–4:00 *Choosing Sense Distinctions for WSD: Psycholinguistic Evidence*
Susan Windisch Brown

4:00–4:15 *Decompounding query keywords from compounding languages*
Enrique Alfonseca, Slaven Bilac and Stefan Pharies

Session 5H: Short Papers 4 (Information Retrieval/Sentiment Analysis)

3:15–3:30 *Multi-domain Sentiment Classification*
Shoushan Li and Chengqing Zong

3:30–3:45 *Evaluating Word Prediction: Framing Keystroke Savings*
Keith Trnka and Kathleen McCoy

3:45–4:00 *Pairwise Document Similarity in Large Collections with MapReduce*
Tamer Elsayed, Jimmy Lin and Douglas Oard

4:00–4:15 *Text Segmentation with LDA-Based Fisher Kernel*
Qi Sun, Runxin Li, Dingsheng Luo and Xihong Wu

4:15–4:45 **Break**

Tuesday, June 17, 2008 (continued)

Session 6A: Question Answering

4:45–5:10 *Improving Search Results Quality by Customizing Summary Lengths*
Michael Kaisser, Marti A. Hearst and John B. Lowe

5:10–5:35 *Using Conditional Random Fields to Extract Contexts and Answers of Questions from Online Forums*
Shilin Ding, Gao Cong, Chin-Yew Lin and Xiaoyan Zhu

5:35–6:00 *Learning to Rank Answers on Large Online QA Collections*
Mihai Surdeanu, Massimiliano Ciaramita and Hugo Zaragoza

Session 6B: Phonology, Morphology 1

4:45–5:10 *Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis*
Meni Adler, Yoav Goldberg, David Gabay and Michael Elhadad

5:10–5:35 *Unsupervised Multilingual Learning for Morphological Segmentation*
Benjamin Snyder and Regina Barzilay

5:35–6:00 *EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)*
Yoav Goldberg, Meni Adler and Michael Elhadad

Session 6C: Machine Translation 3

4:45–5:10 *Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation*
Jakob Uszkoreit and Thorsten Brants

5:10–5:35 *Enriching Morphologically Poor Languages for Statistical Machine Translation*
Eleftherios Avramidis and Philipp Koehn

5:35–6:00 *Learning Bilingual Lexicons from Monolingual Corpora*
Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick and Dan Klein

Tuesday, June 17, 2008 (continued)

Session 6D: Semantics 3

4:45–5:10 *Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora*
Shiqi Zhao, Haifeng Wang, Ting Liu and Sheng Li

5:10–5:35 *Unsupervised Learning of Narrative Event Chains*
Nathanael Chambers and Dan Jurafsky

5:35–6:00 *Semantic Role Labeling Systems for Arabic using Kernel Methods*
Mona Diab, Alessandro Moschitti and Daniele Pighin

7:00–11:00 **Banquet**

Wednesday, June 18, 2008

9:00–10:00 Invited Talk: Susan Dumais, *Supporting Searchers in Searching*

10:00–10:30 **Break**

Session 7A: Summarization

10:30–10:55 *An Unsupervised Approach to Biography Production Using Wikipedia*
Fadi Biadsy, Julia Hirschberg and Elena Filatova

10:55–11:20 *Generating Impact-Based Summaries for Scientific Literature*
Qiaozhu Mei and ChengXiang Zhai

11:20–11:45 *Can You Summarize This? Identifying Correlates of Input Difficulty for Multi-Document Summarization*
Ani Nenkova and Annie Louis

Wednesday, June 18, 2008 (continued)

Session 7B: Discourse and Pragmatics

10:30–10:55 *You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement*
Micha Elsner and Eugene Charniak

10:55–11:20 *An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming*
Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu and Sheng Li

11:20–11:45 *Gestural Cohesion for Topic Segmentation*
Jacob Eisenstein, Regina Barzilay and Randall Davis

Session 7C: Machine Learning 2

10:30–10:55 *Multi-Task Active Learning for Linguistic Annotations*
Roi Reichart, Katrin Tomanek, Udo Hahn and Ari Rappoport

10:55–11:20 *Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields*
Gideon S. Mann and Andrew McCallum

11:20–11:45 *Analyzing the Errors of Unsupervised Learning*
Percy Liang and Dan Klein

Session 7D: Phonology, Morphology 2

10:30–10:55 *Joint Word Segmentation and POS Tagging Using a Single Perceptron*
Yue Zhang and Stephen Clark

10:55–11:20 *A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging*
Wenbin Jiang, Liang Huang, Qun Liu and Yajuan Lü

11:20–11:45 *Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion*
Sittichai Jiampojamarn, Colin Cherry and Grzegorz Kondrak

11:45–1:15 **ACL Business Meeting**

1:15–2:30 **Lunch**

Wednesday, June 18, 2008 (continued)

Session 8A: Information Retrieval 2

2:30–2:55 *A Probabilistic Model for Fine-Grained Expert Search*
Shenghua Bao, Huizhong Duan, Qi Zhou, Miao Xiong, Yunbo Cao and Yong Yu

2:55–3:20 *Credibility Improves Topical Blog Post Retrieval*
Wouter Weerkamp and Maarten de Rijke

3:20–3:45 *Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing*
Andras Csomai and Rada Mihalcea

3:45–4:10 *Resolving Personal Names in Email Using Context Expansion*
Tamer Elsayed, Douglas W. Oard and Galileo Namata

Session 8B: Syntax and Parsing 3

2:30–2:55 *Integrating Graph-Based and Transition-Based Dependency Parsers*
Joakim Nivre and Ryan McDonald

2:55–3:20 *Efficient, Feature-based, Conditional Random Field Parsing*
Jenny Rose Finkel, Alex Kleeman and Christopher D. Manning

3:20–3:45 *A Deductive Approach to Dependency Parsing*
Carlos Gómez-Rodríguez, John Carroll and David Weir

3:45–4:10 *Evaluating a Crosslinguistic Grammar Resource: A Case Study of Wambaya*
Emily M. Bender

Wednesday, June 18, 2008 (continued)

Session 8C: Machine Translation 2

2:30–2:55 *Better Alignments = Better Translations?*
Kuzman Ganchev, João V. Graça and Ben Taskar

2:55–3:20 *Mining Parenthetical Translations from the Web by Word Alignment*
Dekang Lin, Shaojun Zhao, Benjamin Van Durme and Marius Paşca

3:20–3:45 *Soft Syntactic Constraints for Hierarchical Phrased-Based Translation*
Yuval Marton and Philip Resnik

3:45–4:10 *Generalizing Word Lattice Translation*
Christopher Dyer, Smaranda Muresan and Philip Resnik

Session 8D: Semantics 4

2:30–2:55 *Combining Multiple Resources to Improve SMT-based Paraphrasing Model*
Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu and Sheng Li

2:55–3:20 *Extraction of Entailed Semantic Relations Through Syntax-Based Comma Resolution*
Vivek Srikumar, Roi Reichart, Mark Sammons, Ari Rappoport and Dan Roth

3:20–3:45 *Finding Contradictions in Text*
Marie-Catherine de Marneffe, Anna N. Rafferty and Christopher D. Manning

3:45–4:10 *Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs*
Zornitsa Kozareva, Ellen Riloff and Eduard Hovy

4:10–4:40 **Break**

4:40–5:50 Lifetime Achievement Award Presentation

5:50–6:00 Closing Session

Language Dynamics and Capitalization using Maximum Entropy

Fernando Batista[a,b], **Nuno Mamede**[a,c] **and Isabel Trancoso**[a,c]
[a] L^2F – Spoken Language Systems Laboratory - INESC ID Lisboa
R. Alves Redol, 9, 1000-029 Lisboa, Portugal
http://www.l2f.inesc-id.pt/
[b] ISCTE – Instituto de Ciências do Trabalho e da Empresa, Portugal
[c] IST – Instituto Superior Técnico, Portugal.
{fmmb,njm,imt}@l2f.inesc-id.pt

Abstract

This paper studies the impact of written language variations and the way it affects the capitalization task over time. A discriminative approach, based on maximum entropy models, is proposed to perform capitalization, taking the language changes into consideration. The proposed method makes it possible to use large corpora for training. The evaluation is performed over newspaper corpora using different testing periods. The achieved results reveal a strong relation between the capitalization performance and the elapsed time between the training and testing data periods.

1 Introduction

The capitalization task, also known as truecasing (Lita et al., 2003), consists of rewriting each word of an input text with its proper case information. The capitalization of a word sometimes depends on its current context, and the intelligibility of texts is strongly influenced by this information. Different practical applications benefit from automatic capitalization as a preprocessing step: when applied to speech recognition output, which usually consists of raw text, automatic capitalization provides relevant information for automatic content extraction, named entity recognition, and machine translation; many computer applications, such as word processing and e-mail clients, perform automatic capitalization along with spell corrections and grammar check.

The capitalization problem can be seen as a sequence tagging problem (Chelba and Acero, 2004; Lita et al., 2003; Kim and Woodland, 2004), where each lower-case word is associated to a tag that describes its capitalization form. (Chelba and Acero, 2004) study the impact of using increasing amounts of training data as well as a small amount of adaptation. This work uses a Maximum Entropy Markov Model (MEMM) based approach, which allows to combine different features. A large written newspaper corpora is used for training and the test data consists of Broadcast News (BN) data. (Lita et al., 2003) builds a trigram language model (LM) with pairs (word, tag), estimated from a corpus with case information, and then uses dynamic programming to disambiguate over all possible tag assignments on a sentence. Other related work includes a bilingual capitalization model for capitalizing machine translation (MT) outputs, using conditional random fields (CRFs) reported by (Wang et al., 2006). This work exploits case information both from source and target sentences of the MT system, producing better performance than a baseline capitalizer using a trigram language model. A preparatory study on the capitalization of Portuguese BN has been performed by (Batista et al., 2007).

One important aspect related with capitalization concerns the language dynamics: new words are introduced everyday in our vocabularies and the usage of some other words decays with time. Concerning this subject, (Mota, 2008) shows that, as the time gap between training and test data increases, the performance of a named tagger based on co-training (Collins and Singer, 1999) decreases.

This paper studies and evaluates the effects of language dynamics in the capitalization of newspaper

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 1–4,
Columbus, Ohio, USA, June 2008.

corpora. Section 2 describes the corpus and presents a short analysis on the lexicon variation. Section 3 presents experiments concerning the capitalization task, either using isolated training sets or by retraining with different training sets. Section 4 concludes and presents future plans.

2 Newspaper Corpus

Experiments here described use the RecPub newspaper corpus, which consists of collected editions of the Portuguese "Público" newspaper. The corpus was collected from 1999 to 2004 and contains about 148 Million words. The corpus was split into 59 subsets of about 2.5 Million words each (between 9 to 11 per year). The last subset is only used for testing, nevertheless, most of the experiments here described use different training and test subsets for better understanding the time effects on capitalization. Each subset corresponds to about five weeks of data.

2.1 Data Analysis

The number of unique words in each subset is around 86K but only about 50K occur more than once. In order to assess the relation between the word usage and the time gap, we created a number of vocabularies with the 30K more frequent words appearing in each training set (roughly corresponds to a freq > 3). Then, the first and last corpora subsets were checked against each one of the vocabularies. Figure 1 shows the correspondent results, revealing that the number of OOVs (Out of Vocabulary Words) decreases as the time gap between the train and test periods gets smaller.

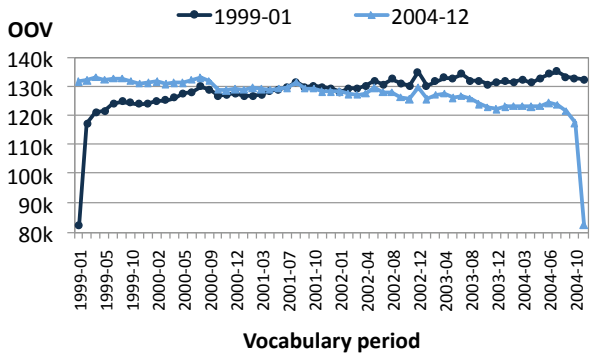


Figure 1: Number of OOVs using a 30K vocabulary.

3 Capitalization

The present study explores only three ways of writing a word: lower-case, all-upper, and first-capitalized, not covering mixed-case words such as "McLaren" and "SuSE". In fact, mixed-case words are also being treated by means of a small lexicon, but they are not evaluated in the scope of this paper.

The following experiments assume that the capitalization of the first word of each sentence is performed in a separated processing stage (after punctuation for instance), since its correct graphical form depends on its position in the sentence. Evaluation results may be influenced when taking such words into account (Kim and Woodland, 2004).

The evaluation is performed using the metrics: Precision, Recall and SER (Slot Error Rate) (Makhoul et al., 1999). Only capitalized words (not lowercase) are considered as slots and used by these metrics. For example: Precision is calculated by dividing the number of correct capitalized words by the number of capitalized words in the testing data.

The modeling approach here described is discriminative, and is based on maximum entropy (ME) models, firstly applied to natural language problems in (Berger et al., 1996). An ME model estimates the conditional probability of the events given the corresponding features. Therefore, all the information must be expressed in terms of features in a pre-processing step. Experiments here described only use features comprising word unigrams and bigrams: w_i (current word), $\langle w_{i-1}, w_i \rangle$ and $\langle w_i, w_{i+1} \rangle$ (bigrams). Only words occurring more than once were included for training, thus reducing the number of misspelled words. All the experiments used the `MegaM` tool (Daumé III, 2004), which uses conjugate gradient and a limited memory optimization of logistic regression. The following subsections describe the achieved results.

3.1 Isolated Training

In order to assess how time affects the capitalization performance, the first experiments consist of producing six isolated language models, one for each year of training data. For each year, the first 8 subsets were used for training and the last one was used for evaluation. Table 1 shows the corresponding capitalization results for the first and last testing sub-

Train	1999-12 test set			2004-12 test set		
	Prec	Rec	SER	Prec	Rec	SER
1999	**94%**	**81%**	**0.240**	92%	76%	0.296
2000	**94%**	**81%**	**0.242**	92%	77%	0.291
2001	94%	79%	0.262	93%	76%	0.291
2002	93%	79%	0.265	93%	78%	0.277
2003	94%	77%	0.276	93%	78%	0.273
2004	93%	77%	0.285	**93%**	**80%**	**0.264**

Table 1: Using 8 subsets of each year for training.

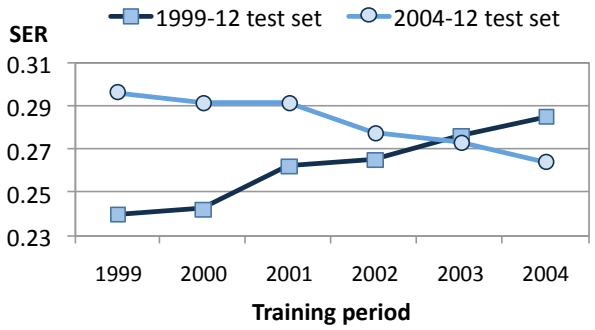


Figure 2: Performance for different training periods.

sets, revealing that performance is affected by the time lapse between the training and testing periods. The best results were always produced with nearby the testing data. A similar behavior was observed on the other four testing subsets, corresponding to the last subset of each year. Results also reveal a degradation of performance when the training data is from a time period after the evaluation data.

Results from previous experiment are still worse than results achieved by other work on the area (Batista et al., 2007) (about 94% precision and 88% recall), specially in terms of recall. This is caused by a low coverage of the training data, thus revealing that each training set (20 Million words) does not provide sufficient data for the capitalization task.

One important problem related with this discriminative approach concerns memory limitations. The memory required increases with the size of the corpus (number of observations), preventing the use of large corpora, such as RecPub for training, with

Evaluation Set	Prec	Rec	SER
2004-12 test set	93%	82%	0.233

Table 2: Training with all RecPub training data.

Checkpoint	LM #lines	Prec	Rec	SER
1999-12	1.27 Million	92%	77%	0.290
2000-12	1.86 Million	93%	79%	0.266
2001-12	2.36 Million	93%	80%	0.257
2002-12	2.78 Million	93%	81%	0.247
2003-12	3.10 Million	93%	82%	0.236
2004-08	3.36 Million	**93%**	**83%**	**0.225**

Table 3: Retraining from Jan. 1999 to Sep. 2004.

available computers. For example, four million events require about 8GB of RAM to process. This problem can be minimized using a modified training strategy, based on the fact that scaling the event by the number of occurrences is equivalent to multiple occurrences of that event. Accordingly to this, our strategy to use large training corpora consists of counting all n-gram occurrences in the training data and then use such counts to produce the corresponding input features. This strategy allows us to use much larger corpora and also to remove less frequent n-grams if desired. Table 2 shows the performance achieved by following this strategy with all the RecPub training data. Only word frequencies greater than 4 were considered, minimizing the effects of misspelled words and reducing memory limitations. Results reveal the expected increase of performance, specially in terms of recall. However, these results can not be directly compared with previous work on this subject, because of the different corpora used.

3.2 Retraining

Results presented so far use isolated training. A new approach is now proposed, which consists of training with new data, but starting with previously calculated models. In other words, previously trained models provide initialized models for the new train. As the training is still performed with the new data, the old models are iteratively adjusted to the new data. This approach is a very clean framework for language dynamics adaptation, offering a number of advantages: (1) new events are automatically considered in the new models; (2) with time, unused events slowly decrease in weight; (3) by sorting the trained models by their relevance, the amount of data used in next training stage can be limited without much impact in the results. Table 3 shows the re-

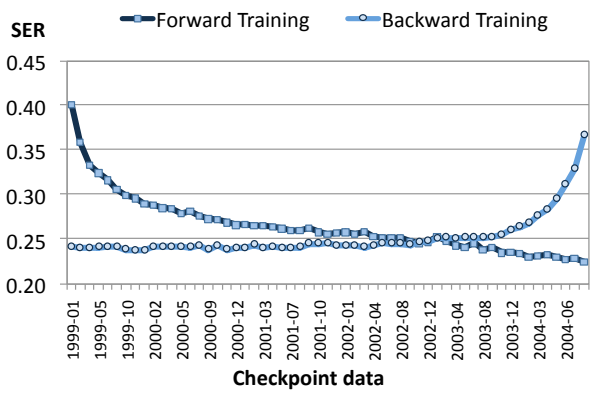


Figure 3: Training forward and backwards

sults achieved with this approach, revealing higher performance as more training data is available.

The next experiment shows that the training order is important. In fact, from previous results, the increase of performance may be related only with the number of events seen so far. For this reason, another experiment have been performed, using the same training data, but retraining backwards. Corresponding results are illustrated in Figure 3, revealing that: the backwards training results are worse than forward training results, and that backward training results do not allways increase, rather stabilize after a certain amount of data. Despite the fact that both training use all training data, in the case of forward training the time gap between the training and testing data gets smaller for each iteration, while in the backwards training is grows. From these results we can conclude that a strategy based on retraining is suitable for using large amounts of data and for language adaptation.

4 Conclusions and Future Work

This paper shows that maximum entropy models can be used to perform the capitalization task, specially when dealing with language dynamics. This approach provides a clean framework for learning with new data, while slowly discarding unused data. The performance achieved is almost as good as using generative approaches, found in related work. This approach also allows to combine different data sources and to explore different features. In terms of language changes, our proposal states that different capitalization models should be used for different time periods.

Future plans include the application of this work to BN data, automatically produced by our speech recognition system. In fact, subtitling of BN has led us into using a baseline vocabulary of 100K words combined with a daily modification of the vocabulary (Martins et al., 2007) and a re-estimation of the language model. This dynamic vocabulary provides an interesting scenario for our experiments.

Acknowledgments

This work was funded by PRIME National Project TECNOVOZ number 03/165, and FCT project CMU-PT/0005/2007.

References

F. Batista, N. J. Mamede, D. Caseiro, and I. Trancoso. 2007. A lightweight on-the-fly capitalization system for automatic speech recognition. In *Proc. of the RANLP 2007*, Borovets, Bulgaria, September.

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

C. Chelba and A. Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. *EMNLP04*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proc. of the Joint SIGDAT Conference on EMNLP*.

H. Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression.

J. Kim and P. C. Woodland. 2004. Automatic capitalisation generation for speech input. *Computer Speech & Language*, 18(1):67–90.

L. V. Lita, A. Ittycheriah, S. Roukos, and N. Kambhatla. 2003. tRuEcasIng. In *Proc. of the 41st annual meeting on ACL*, pages 152–159, Morristown, NJ, USA.

J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. 1999. Performance measures for information extraction. In *Proceedings of the DARPA Broadcast News Workshop*, Herndon, VA, Feb.

C. Martins, A. Teixeira, and J. P. Neto. 2007. Dynamic language modeling for a daily broadcast news transcription system. In *ASRU 2007*, December.

Cristina Mota. 2008. *How to keep up with language dynamics? A case study on Named Entity Recognition*. Ph.D. thesis, IST / UTL.

Wei Wang, Kevin Knight, and Daniel Marcu. 2006. Capitalizing machine translation. In *HLT-NAACL*, pages 1–8, Morristown, NJ, USA. ACL.

Surprising parser actions and reading difficulty

Marisa Ferrara Boston, John Hale
Michigan State University
USA
{mferrara,jthale}@msu.edu

Reinhold Kliegl, Shravan Vasishth
Potsdam University
Germany
{kliegl,vasishth}@uni-potsdam.de

Abstract

An incremental dependency parser's probability model is entered as a predictor in a linear mixed-effects model of German readers' eye-fixation durations. This dependency-based predictor improves a baseline that takes into account word length, *n*-gram probability, and Cloze predictability that are typically applied in models of human reading. This improvement obtains even when the dependency parser explores a tiny fraction of its search space, as suggested by narrow-beam accounts of human sentence processing such as Garden Path theory.

1 Introduction

A growing body of work in cognitive science characterizes human readers as some kind of probabilistic parser (Jurafsky, 1996; Crocker and Brants, 2000; Chater and Manning, 2006). This view gains support when specific aspects of these programs match up well with measurable properties of humans engaged in sentence comprehension.

One way to connect theory to data in this manner uses a parser's probability model to work out the *surprisal* or log-probability of the next word. Hale (2001) suggests this quantity as an index of psycholinguistic difficulty. When the transition from previous word to current word is low-probability, from the parser's perspective, the surprisal is high and the psycholinguistic claim is that behavioral measures should register increased cognitive difficulty. In other words, rare parser actions are cognitively costly. This basic notion has proved remarkably applicable across sentence types and languages (Park and Brew, 2006; Demberg and Keller, 2007; Levy, 2008).

The present work uses the time spent looking at a word during reading as an empirical measure of sentence processing difficulty. From the theoretical side, we calculate word-by-word surprisal predictions from a family of incremental dependency parsers for German based on Nivre (2004); these parsers differ only in the size k of the beam used in the search for analyses of longer and longer sentence-initial substrings. We find that predictions derived even from very narrow-beamed parsers improve a baseline eye-fixation duration model. The fact that any member of this parser family derives a useful predictor shows that at least some syntactic properties are reflected in readers' eye fixation durations. From a cognitive perspective, the utility of small k parsers for modeling comprehension difficulty lends credence to the view that the human processor is a single-path analyzer (Frazier and Fodor, 1978).

2 Parsing costs and theories of reading difficulty

The length of time that a reader's eyes spend fixated on a particular word in a sentence is known to be affected by a variety of word-level factors such as length in characters, n-gram frequency and empirical predictability (Ehrlich and Rayner, 1981; Kliegl et al., 2004). This last factor is the one measured when human readers are asked to guess the next word given a left-context string.

Any role for parser-derived syntactic factors

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 5–8,
Columbus, Ohio, USA, June 2008.

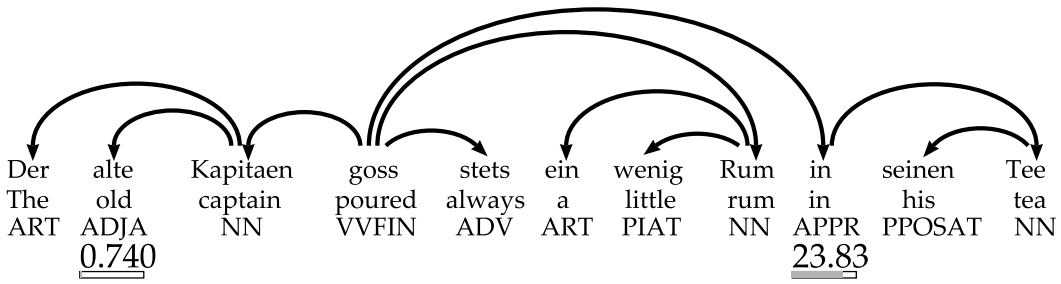


Figure 1: Dependency structure of a PSC sentence.

would have to go beyond these word-level influences. Our methodology imposes this requirement by fitting a kind of regression known as a linear mixed-effects model to the total reading times associated with each sentence-medial word in the Potsdam Sentence Corpus (PSC) (Kliegl et al., 2006). The PSC records the eye-movements of 272 native speakers as they read 144 German sentences.

3 The Parsing Model

The parser's outputs define a relation on word pairs (Tesnière, 1959; Hays, 1964). The structural description in Figure 1 is an example output that depicts this dependency relation using arcs. The word near the arrowhead is the *dependent*, the other word its *head* (or governor).

These outputs are built up by monotonically adding to an initially-empty set of dependency relations as analysis proceeds from left to right. To arrive at Figure 1 the Nivre parser passes through a number of intermediate states that aggregate four data structures, detailed below in Table 1.

σ	A stack of already-parsed unreduced words.
τ	An ordered input list of words.
h	A function from dependent words to heads.
d	A function from dependent words to arc types.

Table 1: Parser configuration.

The stack σ holds words that could eventually be connected by new arcs, while τ lists unparsed words. **h** and **d** are where the current set of dependency arcs reside. There are only four possible transitions from configuration to configuration. `Left-Arc` and `Right-Arc` transitions create dependency relations between the top elements in σ and τ, while `Shift` and `Reduce` transitions manipulate σ.

When more than one transition is applicable, the parser decides between them by consulting a probability model derived from the Negra and Tiger newspaper corpora (Skut et al., 1997; König and Lezius, 2003). This model is called Stack3 because it considers only the parts-of-speech of the top three elements of σ along with the top element of τ. On the PSC this model achieves 87.9% precision and 79.5% recall for unlabeled dependencies. Most of the attachments it gets wrong (Table 2) represent alternative readings that would require semantic guidance to rule out.

Error type	Amount
Noun attachment	4.2%
Prepositional Phrase attachment	3.0%
Conjunction	1.9%
Adverb ambiguity	1.8%
Other	1.1%
Total error	**12.1**%

Table 2: Parser errors by category.

To compare "serial" human sentence processing models against "parallel" models, our implementation does beam search in the space of Nivre-configurations. The number of configurations maintained at any point is a changeable parameter k.

3.1 Surprisal

In Figure 1 the thermometer beneath the German preposition "in" graphically indicates a high surprisal prediction derived from the dependency parser. Greater cognitive effort, reflected in reading time, should be observed on "in" as com-

pared to "alte." The difficulty prediction at "in" ultimately follows from the frequency of verbs taking prepositional complements that follow nominal complements in the training data. Equation 1 expresses the general theory: the surprisal of a word, on a language model, is the logarithm of the prefix probability eliminated in the transition from one word to the next.

$$\text{surprisal}(n) = \log_2 \left(\frac{\alpha_{n-1}}{\alpha_n} \right) \quad (1)$$

The prefix-probability α_n of an initial substring is the total probability of all grammatical analyses that derive $w = w_1 ... w_n$ as a left-prefix (Equation 2).

$$\alpha_n \quad = \sum_{d \in \mathcal{D}(G, wv)} \text{Prob}(d) \quad (2)$$

In a complete parser, every member of $\mathcal{D}$ is in correspondence with a state transition sequence. In the beam-search approximation, only the top *k* configurations are retained from prefix to prefix, which amounts to choosing a subset of $\mathcal{D}$.

4 Study

The study addresses whether surprisal is a significant predictor of reading difficulty and, if it is, whether the beam-size parameter *k* affects the usefulness of the calculated surprisal values in accounting for reading difficulty.

Using total reading time as a dependent measure, we fit a baseline linear mixed-effects model (Equation 3) that takes into account word-level predictors log frequency (*lf*), log bigram frequency (*bi*), word length (*len*), and human predictability given the left context (*pr*).

$$\log(TRT) = \quad (3)$$
$$5.4 - 0.02lf - 0.01bi - 0.59len^{-1} - 0.02pr$$

All of the word-level predictors were statistically significant at the α level 0.05.

Beyond this baseline, we fitted ten other linear mixed-effects models. To the inventory of word-level predictors, each of the ten regressions uniquely added the surprisal predictions calculated from a parser that retains at most *k*=1...9,100 analyses at each prefix. We evaluated the change in relative quality of fit due to surprisal with the *Deviance Information Criterion* (DIC) discussed in Spiegelhalter et al. (2002). Whereas the more commonly applied Akaike Information Criterion (1973) requires the number of estimated parameters to be determined exactly, the DIC facilitates the evaluation of mixed-effects models by relaxing this requirement. When comparing two models, if one of the models has a lower DIC value, this means that the model fit has improved.

4.1 Results and Discussion

Table 3 shows that the linear mixed-effects model of German reading difficulty improves when surprisal values from the dependency parser are used as predictors in addition to the word-level predictors. The coefficients on the baseline predictors remained unchanged (Equation 3) when any of the parser-based predictors was added.

Table 3 also suggests the returns to be had in accounting for reading time are greatest when the beam is limited to a handful of parses. Indeed, a parser that handles a few analyses at a time (*k*=1,2,3) is just as valuable as one that spends far greater memory resources (*k*=100). This observation is consistent with Brants and Crocker's (2000) observation that accuracy can be maintained even when restricted to 1% of the memory required for exhaustive parsing. The role of small *k* dependency parsers in determining the quality of statistical fit challenges the assumption that cognitive functions are global optima. Perhaps human parsing is boundedly rational in the sense of the bound imposed by Stack3 (Simon, 1955).

5 Conclusion

This study demonstrates that surprisal calculated with a dependency parser is a significant predictor of reading times, an empirical measure of cognitive difficulty. Surprisal is a significant predictor even when examined alongside the more commonly used predictors, word length, predictability, and n-gram frequency. The viability of parsers that consider just a small number of analyses at each increment is consistent with conceptions of the human comprehender that incorporate that restriction.

Model	Coefficient	Std. Error	t value	DIC
Baseline	-	-	-	144511.1
k=1	0.033691	0.002285	15	143964.9
k=2	0.038573	0.002510	15	143946.2
k=3	0.037320	0.002693	14	143990.4
k=4	0.041035	0.002853	14	143975.7
k=5	0.048692	0.002953	16	143910.9
k=6	0.046580	0.003063	15	143951.6
k=7	0.045008	0.003118	14	143974.4
k=8	0.042039	0.003165	13	144006.4
k=9	0.040657	0.003225	13	144023.9
k=100	0.029467	0.003878	8	144125.4

Table 3: Coefficients and standard errors from the multiple regressions using different versions of surprisal (baseline predictors' coefficients are not shown for space reasons). t values > 2 are statistically significant at $\alpha = 0.05$. The table also shows DIC values for the baseline model (Equation 3) and the models with baseline predictors plus surprisal.

References

H. Akaike. 1973. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Caski, editors, *2nd International Symposium on Information Theory*, pages 267–281, Budapest, Hungary.

T. Brants and M. Crocker. 2000. Probabilistic parsing and psychological plausibility. In *Proceedings of COLING 2000: The 18th International Conference on Computational Linguistics*.

N. Chater and C. Manning. 2006. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10:287–291.

M. W. Crocker and T. Brants. 2000. Wide-coverage probabilistic sentence processing. *Journal of Psycholinguistic Research*, 29(6):647–669.

V. Demberg and F. Keller. 2007. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. Manuscript, University of Edinburgh.

S. F. Ehrlich and K. Rayner. 1981. Contextual effects on word perception and eye movements during reading. *Journal of Verbal Learning and Verbal Behavior*, 20:641–655.

L. Frazier and J. D. Fodor. 1978. The sausage machine: a new two-stage parsing model. *Cognition*, 6:291–325.

J. Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of 2*$^{\text{nd}}$ *NAACL*, pages 1–8. Carnegie Mellon University.

D.G. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40:511–525.

D. Jurafsky. 1996. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20:137–194.

R. Kliegl, E. Grabner, M. Rolfs, and R. Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology*, 16:262–284.

R. Kliegl, A. Nuthmann, and R. Engbert. 2006. Tracking the mind during reading: The influence of past, present, and future words on fixation durations. *Journal of Experimental Psychology: General*, 135:12–35.

E. König and W. Lezius. 2003. The TIGER language - a description language for syntax graphs, Formal definition. Technical report, IMS, Universität Stuttgart, Germany.

R. Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

J. Nivre. 2004. Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together*, Barcelona, Spain. Association for Computational Linguistics.

J. Park and C. Brew. 2006. A finite-state model of human sentence processing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 49–56, Sydney, Australia.

H. Simon. 1955. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118.

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC.

D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. van der Linde. 2002. Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society*, 64(B):583–639.

L. Tesnière. 1959. *Eléments de syntaxe structurale*. Editions Klincksiek, Paris.

Improving the Performance of the Random Walk Model for Answering Complex Questions

Yllias Chali and Shafiq R. Joty
University of Lethbridge
4401 University Drive
Lethbridge, Alberta, Canada, T1K 3M4
{chali,jotys}@cs.uleth.ca

Abstract

We consider the problem of answering complex questions that require inferencing and synthesizing information from multiple documents and can be seen as a kind of topic-oriented, informative multi-document summarization. The stochastic, graph-based method for computing the relative importance of textual units (i.e. sentences) is very successful in generic summarization. In this method, a sentence is encoded as a vector in which each component represents the occurrence frequency (TF*IDF) of a word. However, the major limitation of the TF*IDF approach is that it only retains the frequency of the words and does not take into account the sequence, syntactic and semantic information. In this paper, we study the impact of syntactic and shallow semantic information in the graph-based method for answering complex questions.

1 Introduction

After having made substantial headway in factoid and list questions, researchers have turned their attention to more complex information needs that cannot be answered by simply extracting named entities like persons, organizations, locations, dates, etc. Unlike informationally-simple factoid questions, complex questions often seek multiple different types of information simultaneously and do not presupposed that one single answer could meet all of its information needs. For example, with complex questions like "What are the causes of AIDS?", the wider focus of this question suggests that the submitter may not have a single or well-defined information need and therefore may be amenable to receiving additional supporting information that is relevant to some (as yet) undefined informational goal. This type of questions require inferencing and synthesizing information from multiple documents. In Natural Language Processing (NLP), this information synthesis can be seen as a kind of topic-oriented, informative multi-document summarization, where the goal is to produce a single text as a compressed version of a set of documents with a minimum loss of relevant information.

Recently, the graph-based method (LexRank) is applied successfully to generic, multi-document summarization (Erkan and Radev, 2004). A topic-sensitive LexRank is proposed in (Otterbacher et al., 2005). In this method, a sentence is mapped to a vector in which each element represents the occurrence frequency (TF*IDF) of a word. However, the major limitation of the TF*IDF approach is that it only retains the frequency of the words and does not take into account the sequence, syntactic and semantic information thus cannot distinguish between "The hero killed the villain" and "The villain killed the hero". The task like *answering complex questions* that requires the use of more complex syntactic and semantics, the approaches with only TF*IDF are often inadequate to perform fine-level textual analysis.

In this paper, we extensively study the impact of syntactic and shallow semantic information in measuring similarity between the sentences in the random walk model for answering complex questions. We argue that for this task, similarity measures based on syntactic and semantic information performs better and can be used to characterize the

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 9–12,
Columbus, Ohio, USA, June 2008.

relation between a question and a sentence (answer) in a more effective way than the traditional TF*IDF based similarity measures.

2 Graph-based Random Walk Model for Text Summarization

In (Erkan and Radev, 2004), the concept of graph-based centrality is used to rank a set of sentences, in producing generic multi-document summaries. A similarity graph is produced where each node represents a sentence in the collection and the edges between nodes measure the cosine similarity between the respective pair of sentences. Each sentence is represented as a vector of term specific weights. The term specific weights in the sentence vectors are products of term frequency (tf) and inverse document frequency (idf). The degree of a given node is an indication of how much important the sentence is. To apply LexRank to query-focused context, a topic-sensitive version of LexRank is proposed in (Otterbacher et al., 2005). The score of a sentence is determined by a mixture model:

$$p(s|q) = d \times \frac{rel(s|q)}{\sum_{z \in C} rel(z|q)} + (1-d) \times \sum_{v \in C} \frac{sim(s,v)}{\sum_{z \in C} sim(z,v)} \times p(v|q) \quad (1)$$

Where, $p(s|q)$ is the score of a sentence s given a question q, is determined as the sum of its relevance to the question (i.e. $rel(s|q)$) and the similarity to other sentences in the collection (i.e. $sim(s,v)$). The denominators in both terms are for normalization. C is the set of all sentences in the collection. The value of the parameter d which we call "bias", is a trade-off between two terms in the equation and is set empirically. We claim that for a complex task like answering complex questions where the relatedness between the query sentences and the document sentences is an important factor, the graph-based random walk model of ranking sentences would perform better if we could encode the syntactic and semantic information instead of just the bag of word (i.e. TF*IDF) information in calculating the similarity between sentences. Thus, our mixture model for answering complex questions is:

$$p(s|q) = d \times TREESIM(s,q) + (1-d) \times \sum_{v \in C} TREESIM(s,v) \times p(v|q) \quad (2)$$

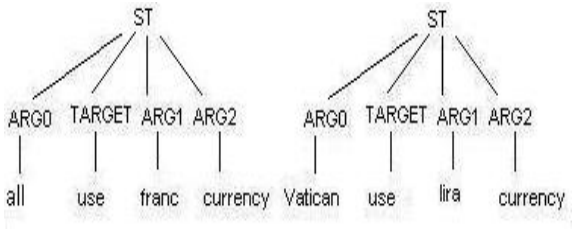


Figure 1: Example of semantic trees

Where *TREESIM(s,q)* is the normalized syntactic (and/or semantic) similarity between the query (q) and the document sentence (s) and C is the set of all sentences in the collection. In cases where the query is composed of two or more sentences, we compute the similarity between the document sentence (s) and each of the query-sentences (q_i) then we take the average of the scores.

3 Encoding Syntactic and Shallow Semantic Structures

Encoding syntactic structure is easier and straight forward. Given a sentence (or query), we first parse it into a syntactic tree using a syntactic parser (i.e. Charniak parser) and then we calculate the similarity between the two trees using the general tree kernel function (Section 4.1).

Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems like ASSERT (Hacioglu et al., 2003). For example, consider the PB annotation:

[ARG0 all][TARGET use][ARG1 the french franc][ARG2 as their currency]

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

[ARG0 the Vatican][TARGET use][ARG1 the Italian lira][ARG2 as their currency]

In order to calculate the semantic similarity between the sentences, we first represent the annotated sentence using the tree structures like Figure 1 which we call Semantic Tree (ST). In the semantic tree, arguments are replaced with the most important word-often referred to as the semantic head.

The sentences may contain one or more subordinate clauses. For example the sentence, "the Vatican, located wholly within Italy uses the Italian lira

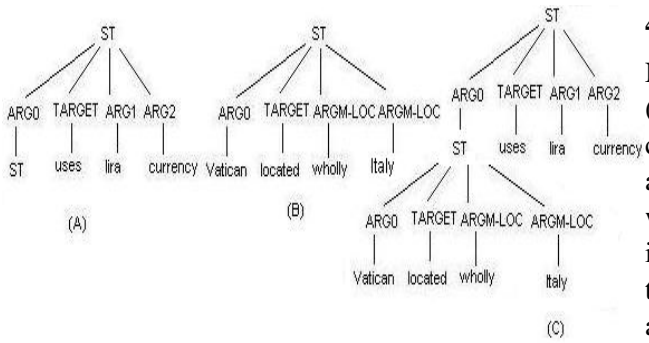


Figure 2: Two STs composing a STN

as their currency." gives the STs as in Figure 2. As we can see in Figure 2(A), when an argument node corresponds to an entire subordinate clause, we label its leaf with ST , e.g. the leaf of ARG0. Such ST node is actually the root of the subordinate clause in Figure 2(B). If taken separately, such STs do not express the whole meaning of the sentence, hence it is more accurate to define a single structure encoding the dependency between the two predicates as in Figure 2(C). We refer to this kind of nested STs as STNs.

4 Syntactic and Semantic Kernels for Text

4.1 Tree Kernels

Once we build the trees (syntactic or semantic), our next task is to measure the similarity between the trees. For this, every tree T is represented by an m dimensional vector $v(T) = (v_1(T), v_2(T), \cdots v_m(T))$, where the i-th element $v_i(T)$ is the number of occurrences of the i-th tree fragment in tree T. The tree fragments of a tree are all of its sub-trees which include at least one production with the restriction that no production rules can be broken into incomplete parts.

Implicitly we enumerate all the possible tree fragments $1, 2, \cdots, m$. These fragments are the axis of this m-dimensional space. Note that this could be done only implicitly, since the number m is extremely large. Because of this, (Collins and Duffy, 2001) defines the tree kernel algorithm whose computational complexity does not depend on m. We followed the similar approach to compute the tree kernel between two syntactic trees.

4.2 Shallow Semantic Tree Kernel (SSTK)

Note that, the tree kernel (TK) function defined in (Collins and Duffy, 2001) computes the number of common subtrees between two trees. Such subtrees are subject to the constraint that their nodes are taken with all or none of the children they have in the original tree. Though, this definition of subtrees makes the TK function appropriate for syntactic trees but at the same time makes it not well suited for the semantic trees (ST) defined in Section 3. For instance, although the two STs of Figure 1 share most of the subtrees rooted in the *ST* node, the kernel defined above computes no match.

The critical aspect of the TK function is that the productions of two evaluated nodes have to be identical to allow the match of further descendants. This means that common substructures cannot be composed by a node with only some of its children as an effective ST representation would require. Moschitti et al. (2007) solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST. We followed the similar approach to compute the SSTK.

5 Experiments

5.1 Evaluation Setup

The Document Understanding Conference (DUC) series is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments. We used the DUC 2007 datasets for evaluation.

We carried out automatic evaluation of our summaries using ROUGE (Lin, 2004) toolkit, which has been widely adopted by DUC for automatic summarization evaluation. It measures summary quality by counting overlapping units such as the n-gram (ROUGE-N), word sequences (ROUGE-L and ROUGE-W) and word pairs (ROUGE-S and ROUGE-SU) between the candidate summary and the reference summary. ROUGE parameters were set as the same as DUC 2007 evaluation setup. All the ROUGE measures were calculated by running ROUGE-1.5.5 with stemming but no removal of stopwords. The ROUGE run-time parameters are:

ROUGE-1.5.5.pl -2 -1 -u -r 1000 -t 0 -n 4 -w 1.2 -m -l 250 -a

The purpose of our experiments is to study the impact of the syntactic and semantic representation for complex question answering task. To accomplish this, we generate summaries for the topics of DUC 2007 by each of our four systems defined as below:

(1) TF*IDF: system is the original topic-sensitive LexRank described in Section 2 that uses the similarity measures based on tf*idf.

(2) SYN: system measures the similarity between the sentences using the *syntactic tree* and the *general tree kernel* function defined in Section 4.1.

(3) SEM: system measures the similarity between the sentences using the *shallow semantic tree* and the *shallow semantic tree kernel* function defined in Section 4.2.

(4) SYNSEM: system measures the similarity between the sentences using both the *syntactic* and *shallow semantic* trees and their associated *kernels*. For each sentence it measures the syntactic and semantic similarity with the query and takes the average of these measures.

5.2 Evaluation Results

The comparison between the systems in terms of their F-scores is given in Table 1. The SYN system improves the ROUGE-1, ROUGE-L and ROUGE-W scores over the TF*IDF system by 2.84%, 0.53% and 2.14% respectively. The SEM system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the TF*IDF system by 8.46%, 6.54%, 6.56%, and 11.68%, and over the SYN system by 5.46%, 5.98%, 4.33%, and 12.97% respectively. The SYNSEM system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the TF*IDF system by 4.64%, 1.63%, 2.15%, and 4.06%, and over the SYN system by 1.74%, 1.09%, 0%, and 5.26% respectively. The SEM system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the SYNSEM system by 3.65%, 4.84%, 4.32%, and 7.33% respectively which indicates that including syntactic feature with the semantic feature degrades the performance.

Systems	ROUGE 1	ROUGE L	ROUGE W	ROUGE SU
TF*IDF	0.359458	0.334882	0.124226	0.130603
SYN	0.369677	0.336673	0.126890	0.129109
SEM	0.389865	0.356792	0.132378	0.145859
SYNSEM	0.376126	0.340330	0.126894	0.135901

Table 1: ROUGE F-scores for different systems

6 Conclusion

In this paper, we have introduced the syntactic and shallow semantic structures and discussed their impacts in measuring the similarity between the sentences in the random walk framework for answering complex questions. Our experiments suggest the following: (a) similarity measures based on the syntactic tree and/or shallow semantic tree outperforms the similarity measures based on the TF*IDF and (b) similarity measures based on the shallow semantic tree performs best for this problem.

References

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of Neural Information Processing Systems*, pages 625–632, Vancouver, Canada.

G. Erkan and D. R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2003. Shallow Semantic Parsing Using Support Vector Machines. In *Technical Report TR-CSLR-2003-03*, University of Colorado.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of the international conference on Language Resources and Evaluation*, Las Palmas, Spain.

C. Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of Association for Computational Linguistics*, pages 74–81, Barcelona, Spain.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classificaion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 776–783, Prague, Czech Republic. ACL.

J. Otterbacher, G. Erkan, and D. R. Radev. 2005. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, Canada.

Dimensions of Subjectivity in Natural Language

Wei Chen
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
weichen@cs.cmu.edu

Abstract

Current research in automatic subjectivity analysis deals with various kinds of subjective statements involving human attitudes and emotions. While all of them are related to *subjectivity*, these statements usually touch on multiple dimensions such as non-objectivity[1], uncertainty, vagueness, non-objective measurability, imprecision, and ambiguity, which are inherently different. This paper discusses the differences and relations of six dimensions of *subjectivity*. Conceptual and linguistic characteristics of each dimension will be demonstrated under different contexts.

1 Introduction

Natural language involves statements that do not contain complete, exact, and unbiased information. Many of these are subjective, which share the common property described in narrative theory (Banfield, 1982) as "(subjective statements) must all be referred to the speaking subject for interpretation". Wiebe (1990) further adapted this definition of *subjectivity* to be "the linguistic expression of private states (Quirk et al., 1985)". So far, linguistic cues have played an important role in research of subjectivity recognition (e.g. (Wilson et al., 2006)), sentiment analysis (e.g. (Wilson et al., 2005; Pang and Lee, 2004)), and emotion studies (e.g. (Pennebaker et al., 2001)). While most linguistic cues are grouped under the general rubric of *subjectivity*, they are usually originated from different dimensions, including:

- non-objectivity
- uncertainty
- vagueness
- non-objective measurability
- imprecision
- ambiguity

These dimensions all mingle in various applications that deal with subjective statements. For example, opinion extraction processes statements involving non-objectivity and uncertainty. Evaluation and sentiment analysis deal with vague words, which often covers the issue of non-objective measurability and imprecision. Ambiguity sometimes involves implicit subjectivity that is hard to recognize from linguistic patterns, which leads to great challenge of identifying and understanding subjective statements.

Since multiple dimensions are involved in *subjectivity*, discriminating them may be helpful in understanding *subjectivity* and related concepts. The following sections discuss characteristics and relations of the six dimensions of *subjectivity*.

2 Dimensions of Subjective Statements

2.1 Non-objectivity

In this paper, we define non-objectivity as the property of creating a bias according to personal beliefs, judgments and emotions. This does not include the kind of subjectivity originated from particular properties of linguistic units that lead to personal interpretations. Non-objectivity exists in subjective

[1] We use the term "non-objectivity" to refer to the property of creating a bias from a speaker's point of view that is not supported by sufficient objective evidence. It is not identical to the *subjectivity* that involves all the dimensions we discuss in this paper.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 13–16,
Columbus, Ohio, USA, June 2008.

statements such as opinions, evaluations, and persuasive statements. Non-objectivity can be recognized from linguistic patterns including words explicitly expressing thoughts, beliefs, speculations, and postulations such as "think", "believe", "hope" and "guess". Although linguistic cues are found to be reliable, there are cases of non-objectivity that cannot be identified merely from lexical, syntactical or morphological cues. For example, sentence (1) and sentence (2) are very similar in linguistic structures, but only sentence (2) is non-objective.

(1) Living things cannot survive without water.

(2) He cannot survive without music.

Apart from linguistic patterns and conceptual characteristics of non-objectivity, there are two main issues in non-objectivity recognition. First, non-objectivity cannot be clearly identified without knowledge about its source (Wiebe et al., 2005). For example, "Bob says the red team is about to win" is objective with respect to the position of the speaker of the sentence, who objectively stated a speech event. But the fragment "the red team is about to win" is an opinion of Bob. Hence, whether a statement is an opinion depends on both the scope of the statement and the source of that statement. Second, non-objectivity always lies in a context, which cannot be ignored (Wiebe, 1990). For example, "Pinocchio's nose" is likely to be objective when used within the context of the famous fairy tale. But the same phrase can be used subjectively as a metaphor in other contexts, where it may indicate non-objectivity.

2.2 Uncertainty

Uncertainty can indicate either subjectivity or objectivity. Flagged by words such as "probably" and "maybe", statements expressing uncertainty are usually considered subjective because "being uncertain" itself can be a subjective mental activity. However, uncertainty is not a subtype of *subjectivity*. Consider the following sentences:

(3) Bob has probably already finished his homework.

(4) A poll of recent public opinions shows that Bob is likely to win the nomination.

Sentence (3) is a subjective statement, where the speaker expresses his/her postulation of "Bob finished his homework" through the uncertainty indicated by "probably". On the contrary, sentence (4) is an objective statement, although uncertainty about a future event exists. This sentence reports a conclusion drawn from sufficient evident that Bob takes the majority vote based on the survey, which does not rely on a particular speaking subject for interpretation. In this case, uncertainty does not necessarily imply *subjectivity*.

On the other hand, people sometimes explicitly indicate uncertainty to avoid being subjective.

(5) It is *possible* that the red team will win.

(6) It is *likely* that the red team will win.

(7) The red team will win.

We could easily imagine a scenario where sentence (5) is more objective than sentence (6) and (7). For example, the speaker may believe that the red team will lose, but in order to avoid personal bias, he/she may instead say: "It is possible that the red team will win (but the blue team has a better chance)." In general, explicitly showing uncertainty can imply postulation, but it can also convey the intention of being objective by not excluding other possibilities.

Uncertainty sometimes exists in statements where no linguistic cues are present. For example, the linguistic pattern of sentence (7) is similar to that of "I will have an exam tomorrow", but the later one is usually used to describe an objective future event while sentence (7) can be semantically identical to sentence (6)[2], although the indicator of uncertainty in sentence (7) is not shown explicitly.

2.3 Vagueness, Non-objective Measurability, and Imprecision

Vagueness refers to a property of the concepts that have no precise definitions. For example, gradable words such as "small" and "popular" are sometimes treated as linguistic cues of vagueness, and they are found to be good indicators of *subjectivity* (Hatzivassiloglou and Wiebe, 2000).

Especially, gradable words are vague if there is no well-defined frame of reference. This in some cases

[2]These two are identical as long as the game is not fixed.

leads to two issues: comparison class and boundary. In the sentence "Elephants are big", the comparison class of "elephants" is unclear: we could compare the size of elephants with either land animals or all the animals including both land and aquatic creatures[3]. Also, there is no clear boundary between "being small" and "not being small". Different individuals usually have their own fuzzy boundaries for vague concepts. As such, vague words are usually treated as important cues for *subjectivity*. However, learning which words are vague is non-trivial, because vagueness cannot be hard-coded into lexicons. For example, the gradable word "cold" is vague in sentence (8) but not in sentence (9). The difference between these two is the one in sentence (9) has a known boundary which is the temperature for liquid water to exist, and the one in sentence (8) simply reflects personal perception.

(8) It is *cold* outside.

(9) It is too *cold* during the night on the moon for liquid water to exist.

Vagueness is often a strong indicator of *subjectivity* because it involves personal explanation of a concept. But there are exceptions. For example, the definition of "traditional education" can be vague, but talking about "traditional education" may not necessarily imply *subjectivity*.

When speaking of qualities, there are two major dimensions related to vagueness: non-objective measurability and imprecision. Attributes like height, length, weight, temperature, and time are objectively measurable, whereas things like beauty and wisdom are usually not objectively measurable. Vagueness exists at different levels for non-objectively and objectively measurable qualities. For non-objectively measurable qualities, vagueness exists at the conceptual level, where it intersects with non-objectivity. In the sentence "He is not as charming as his brother", the word "charming" refers to a quality whose interpretation may vary among different cultures and different individuals. For objectively measurable qualities, vagueness exists at the boundary-setting level, where either subjectivity or common sense comes into play. Sentence (10) shows an example of the objectively measurable quality "long time" indicating an opinion that the speaker is unsatisfied with someone's work. On the contrary, an objective meaning of "long time" in sentence (11) can be resolved by common sense.

(10) You finally finished the work, but it took you a *long time*.

(11) Intelligent life took a *long time* to develop on Earth.[4]

Statements involving objectively measurable quantities often have an imprecision problem, where vagueness is usually resolved from common agreements on small variations of values. For example, "Bob is six feet tall" usually implies that the height is "around" six feet[5], with a commonly acceptable precision of about an inch. Generally, specific precisions are determined by variations tied to measurement technologies for specific quantities: the precision for the size of a cell may be around a micron, and the error tolerance for the distance between stars can be on the order of light years. Imprecision can also indicate subjectivity when used for subjective estimation. For instance, "Bob needs two days to finish his homework" is usually not telling an exact period of time, but a personal estimation.

2.4 Ambiguity

While vagueness exists at the conceptual level, ambiguity lies at the level of linguistic expressions. In other words, an ambiguous statement contains linguistic expressions that can refer to multiple explanations, whereas a vague statement carries a concept with unclear or soft definition.

Previous studies have explored the relationship between ambiguity and *subjectivity*. They have shown that subjectivity annotations can be helpful for word sense disambiguation when a word has distinct subjective senses and objective senses (Wiebe and Mihalcea, 2006).

Lexical and syntactical ambiguity usually can be resolved from contextual information and/or common consensus. But when ambiguity is used intentionality, identifying and understanding the ambiguity become creative and interactive procedures,

[3]Other comparison classes are also possible.

[4]Sentence fragment adapted from *Astrobiology Magazine (Dec 02, 2002)*.

[5]It could also mean "at least six feet tall" in some cases.

which usually indicate *subjectivity*. The sentence "I'd like to see more of you" is an example of this kind, which could be used to indicate multiple meanings under the same context [6].

3 Mixtures of Multiple Dimensions

In many cases, subjective statements involve multiple of the dimensions discussed in previous sections. For example, the subjectivity of the sentence "It's a nice car" comes from three dimensions: non-objectivity, vagueness and ambiguity. First, "a car being nice" is usually a personal opinion which may not be commonly acceptable. Second, the gradable word "nice" indicates vagueness, since there is no clear boundary for "being nice". Third, the sentence is also ambiguous because "nice" could refer to appearance, acceleration, angle rate, and many other metrics that might affect personal evaluations.

For information retrieval systems, processing natural queries such as "find me the popular movies of 2007" requires proper understanding of the vague word "popular". Besides, non-objectivity and ambiguity also take part in the query: on the non-objectivity side, the definition of "popular" may differ according to different individuals; on the ambiguity side, the word "popular" may refer to different metrics related to the popularity of a movie such as movie ratings and box office performance.

In applications requiring certain level of language-understanding, things can get even more complicated while different dimensions weave together. As in sentence (5), the speaker may bias towards the blue team while he/she shows uncertainty towards the red team. Correctly understanding this kind of subjective statements would probably need some investigation in different dimensions of *subjectivity*.

4 Conclusion

In this paper, we demonstrated that subjectivity in natural language is a complex phenomenon that contains multiple dimensions including non-objectivity, uncertainty, vagueness, non-objective measurability, imprecision and ambiguity. These dimensions pattern together in various kinds of subjective statements such as opinions, evaluations and natural queries. Since these dimensions have different behaviors in subjective statements, discriminating them in both linguistic and psychological aspects would be necessary in subjectivity analysis.

[6] Kent Bach, Ambiguity. *Routledge Encyclopedia of Philosophy*, http://online.sfsu.edu/ kbach/ambguity.html

Acknowledgments

The author would like to thank Scott Fahlman for the original motivation of the idea and helpful discussions.

References

Ann Banfield. 1982. *Unspeakable Sentences: Narration and Representation in the Language of Fiction*. Routledge and Kegan Paul, Boston.

Vasileios Hatzivassiloglou and Janyce Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th conference on Computational linguistics*, pages 299–305.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

James Pennebaker, Martha Francis, and Roger Booth. 2001. *Linguistic Inquiry and Word Count: LIWC*. Lawrence Erlbaum Associates, Mahwah.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, New York.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of the ACL*, pages 1065–1072.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. In *Language Resources and Evaluation*, volume 39, pages 165–210.

Janyce Wiebe. 1990. *Recognizing Subjective Sentences: A Computational Investigation of Narrative Text*. Ph.D. thesis, SUNY Buffalo Dept. of Computer Science.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2006. Recognizing strong and weak opinion clauses. *Computational Intelligence*, 22(2):73–99.

Extractive Summaries for Educational Science Content

Sebastian de la Chica, Faisal Ahmad, James H. Martin, Tamara Sumner
Institute of Cognitive Science
Department of Computer Science
University of Colorado at Boulder
sebastian.delachica, faisal.ahmad, james.martin,
tamara.sumner@colorado.edu

Abstract

This paper describes an extractive summarizer for educational science content called COGENT. COGENT extends MEAD based on strategies elicited from an empirical study with domain and instructional experts. COGENT implements a hybrid approach integrating both domain independent sentence scoring features and domain-aware features. Initial evaluation results indicate that COGENT outperforms existing summarizers and generates summaries that closely resemble those generated by human experts.

1 Introduction

Knowledge maps consist of nodes containing rich concept descriptions interconnected using a limited set of relationship types (Holley and Dansereau, 1984). Learning research indicates that knowledge maps may be useful for learners to understand the macro-level structure of an information space (O'Donnell et al., 2002). Knowledge maps have also emerged as an effective computational infrastructure to support the automated generation of conceptual browsers. Such conceptual browsers appear to allow students to focus on the science content of large educational digital libraries (Sumner et al., 2003), such as the Digital Library for Earth System Education (DLESE.org). Knowledge maps have also shown promise as domain and student knowledge representations to support personalized learning interactions (de la Chica et al., 2008).

In this paper we describe our progress towards the generation of science concept inventories as summaries of digital library collections. Such inventories provide the basis for the construction of knowledge maps useful both as computational knowledge representations and as learning resources for presentation to the student.

2 Related Work

Our work is informed by efforts to automate the acquisition of ontology concepts from text. OntoLearn extracts candidate domain terms from texts using a syntactic parse and updates an existing ontology with the identified concepts and relationships (Navigli and Velardi, 2004). Knowledge Puzzle focuses on n-gram identification to produce a list of candidate terms pruned using information extraction techniques to derive the ontology (Zouaq et al., 2007). Lin and Pantel (2002) discover concepts using clustering by committee to group terms into conceptually related clusters. These approaches produce ontologies of very fine granularity and therefore graphs that may not be suitable for presentation to a student.

Multi-document summarization (MDS) research also informs our work. XDoX analyzes large document sets to extract important themes using n-gram scoring and clustering (Hardy et al., 2002). Topic representation and topic themes have also served as the basis for the exploration of promising MDS techniques (Harabagiu and Lacatusu, 2005). Finally, MEAD is a widely used MDS and evaluation platform (Radev et al., 2000). While all these systems have produced promising results in automated evaluations, none have directly targeted educational content collections.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 17–20,
Columbus, Ohio, USA, June 2008.

3 Empirical Study

We have conducted a study to capture how human experts processed digital library resources to create a domain knowledge map. Four geology and instructional design experts selected 20 resources from DLESE to construct a knowledge map on earthquakes and plates tectonics for high school age learners. The resulting knowledge map consists of 564 concepts and 578 relationships.

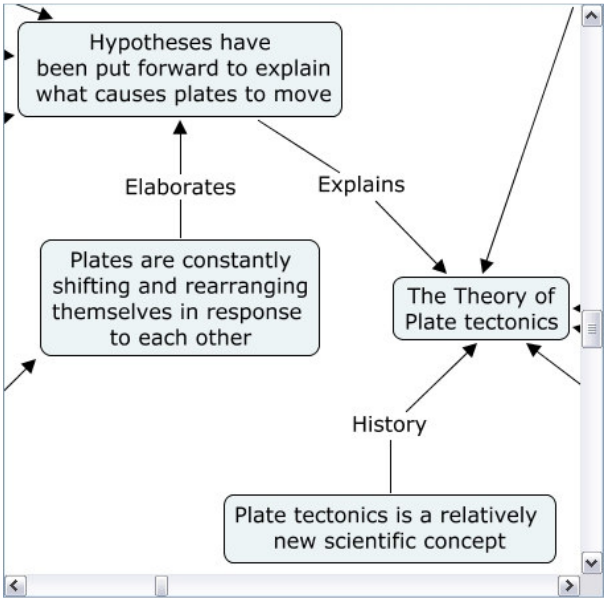


Figure 1. Expert knowledge map excerpt

The concepts include 7,846 words, or 5% of the resources. Our experts relied on copying-and-pasting (58%) and paraphrasing (37%) to create most concepts. Only 5% of the concepts could not be traced directly to the original resources. Relationship types were used in a Zipf-like distribution with the top 2 relationship types each accounting for more than 10% of all relationships: elaborations (19%) and examples (14%).

Analysis by an independent instructional expert indicates that this knowledge map provides adequate coverage of nationally-recognized educational goals on earthquakes and plate tectonics for high school learners using the American Association for the Advancement of Science (AAAS) Benchmarks (Project 2061, 1993).

Verbal protocol analysis shows that all experts used external sources to create the knowledge map, including their own expertise, other digital library resources, and the National Science Education Standards (NSES), a comprehensive collection of nationally-recognized science learning goals for K-12 students (National Research Council, 1996).

We have examined sentence extraction agreement between experts using the prevalence-adjusted bias-adjusted (PABA) kappa to account for prevalence of judgments and conflicting biases amongst experts (Byrt et al., 1993). The average PABA-kappa value of 0.62 indicates that experts substantially agree on sentence extraction from digital library resources. This level of agreement suggests that these concepts may serve as the reference summary to evaluate our system.

4 Summarizer for Science Education

We have implemented an extractive summarizer for educational science content, COGENT, based on MEAD version 3.11 (Radev et al., 2000). COGENT complements the default MEAD sentence scoring features with features based on findings from the empirical study. COGENT represents a hybrid approach integrating bottom-up (hypertext and content word density) and top-down (educational standards and gazetteer) features.

We model how human experts used external information sources with the educational standards feature. This feature leverages the text of the relevant AAAS Benchmarks and associated NSES. Each sentence receives a score based on its TFIDF similarity to the textual contents of these learning goals and educational standards.

We have developed a feature that reflects the large number of examples extracted by the experts. Earth science examples often refer to geographical locations and geological formations. The gazetteer feature checks named entities from each sentence against the Alexandria Digital Library (ADL) Gazetteer (Hill, 2000). A gazetteer is a geo-referencing resource containing location and type information about place-names. Each sentence receives a TFIDF score based on place-name term frequency and overall uniqueness in the gazetteer. Our assumption is that geographical locations with more unique names may be more pedagogically relevant.

Based on the intuition that the HTML structure of a resource reflects relevancy, we have developed the hypertext feature. This feature computes a sentence score directly proportional to the HTML heading level and inversely proportional to the relative paragraph number within a heading and to the relative sentence position within a paragraph.

To promote the extraction of sentences containing science concepts, we have developed the content word density feature. This feature computes the ratio of content to function words in a sentence. Function words are identified using a stopword list, and the feature only keeps sentences featuring more content words than function words.

We compute the final sentence score by adding the MEAD default feature scores (centroid and position) to the COGENT feature scores (educational standards, gazetteer, and hypertext). COGENT keeps sentences that pass the cut-off constraints, including the MEAD sentence length of 9 and COGENT content word density of 50%. The default MEAD cosine re-ranker eliminates redundant sentences. Since the experts used 5% of the total word count in the resources, we produce summaries of that same length.

5 Evaluation

We have evaluated COGENT by processing the 20 digital library resources used in the empirical study and comparing the output against the concepts identified by the experts. Three configurations are considered: Random, Default, and COGENT. The Random summary uses MEAD to extract random sentences. The Default summary uses the MEAD centroid, position and length default features. Finally, the COGENT summary extends MEAD with the COGENT features.

We use ROUGE (Lin, 2004) to assess summary quality using common n-gram counts and longest common subsequence (LCS) measures. We report on ROUGE-1 (unigrams), ROUGE-2 (bigrams), ROUGE W-1.2 (weighted LCS), and ROUGE-S* (skip bigrams) as they have been shown to correlate well with human judgments for longer multi-document summaries (Lin, 2004). Table 1 shows the results for recall (R), precision (P), and balanced f-measure (F).

		Random	Default	COGENT
R-1	**R**	0.4855	0.4976	0.6073
	P	0.5026	0.5688	0.6034
	F	0.4939	0.5308	0.6054
R-2	**R**	0.0972	0.1321	0.1907
	P	0.1006	0.1510	0.1895
	F	0.0989	0.1409	0.1901
R-W-1.2	**R**	0.0929	0.0951	0.1185
	P	0.1533	0.1733	0.1877
	F	0.1157	0.1228	0.1453
R-S*	**R**	0.2481	0.2620	0.3820
	P	0.2657	0.3424	0.3772
	F	0.2566	0.2969	0.3796

Table 1. Quality evaluation results

Table 1 indicates that COGENT consistently outperforms the Random and Default summaries. These results indicate the promise of our approach to generate extractive summaries of educational science content. Given our interest in generating a pedagogically effective domain knowledge map, we have also conducted a content-centric evaluation.

To characterize the COGENT summary contents, one of the authors manually constructed a summary corresponding to the best case output for an extractive summarizer. This Best Case summary comprises all the sentences from the resources that align to all the concepts selected by the experts. This summary comprises 621 sentences consisting of 13,116 words, or about a 9% word compression.

We use ROUGE-L to examine the union LCS between the reference and candidate summaries, thus capturing their linguistic surface structure similarity. We also use MEAD to report on cosine similarity. Table 2 shows the results for recall (R), precision (P), and balanced f-measure (F).

		Random (5%)	Default (5%)	COGENT (5%)	Best Case (9%)
R-L	**R**	0.4814	0.4919	0.6021	0.9669
	P	0.4982	0.5623	0.5982	0.6256
	F	0.4897	0.5248	0.6001	0.7597
Cosine		0.5382	0.6748	0.8325	0.9323

Table 2. Content evaluation results (word compression)

The ROUGE-L scores consistently indicate that the COGENT summary may be closer to the reference in linguistic surface structure than either the Random or Default summaries. Since the COGENT ROUGE-L recall score (R=0. 6021) is lower than the Best Case (R=0.9669), it is likely that COGENT may be extracting different sentences than those selected by the experts. Based on the high cosine similarity with the reference (0.8325), we hypothesize that COGENT may be selecting sentences that cover very similar concepts to those selected by the experts, but expressed differently.

Given the difference in word compression for the Best Case summary, we have performed an

incremental analysis using the ROUGE-L measure shown in Figure 2.

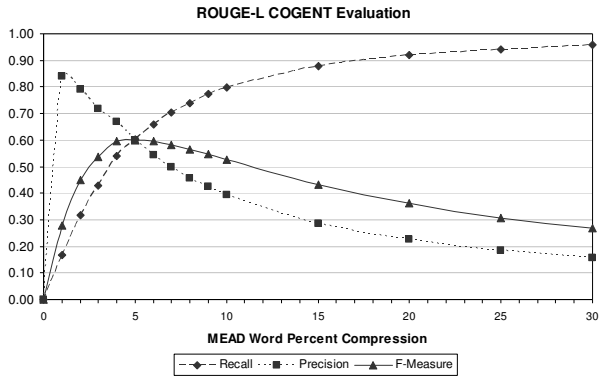


Figure 2. Incremental COGENT ROUGE-L analysis

Figure 2 indicates that COGENT can match the Best Case recall (R=0.9669) by generating a longer summary. For educational applications, lengthier summaries may be better suited for computational purposes, such as diagnosing student understanding, while shorter summaries may be more appropriate for display to the student.

6 Conclusions

COGENT extends MEAD based on strategies elicited from an empirical study with domain and instructional experts. Initial evaluation results indicate that COGENT holds promise for identifying important domain pedagogical concepts. We are exploring portability to other science education domains and machine learning techniques to connect concepts into a knowledge map. Automating the creation of inventories of pedagogically important concepts may represent an important step towards scalable intelligent tutoring systems.

Acknowledgements

This research is funded in part by the National Science Foundation under NSF IIS/ALT Award 0537194. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

T. Byrt, J. Bishop and J. B. Carlin. Bias, prevalence, and kappa. *Journal of Clinical Epidemiology*, *46*, *5* (1993), 423-429.

S. de la Chica, F. Ahmad, T. Sumner, J. H. Martin and K. Butcher. Computational foundations for personalizing instruction with digital libraries. *International Journal of Digital Libraries*, to appear in the Special Issue on Digital Libraries and Education.

S. Harabagiu and F. Lacatusu. Topic themes for multi-document summarization. In *Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Salvador, Brazil, 2005), 202-209.

H. Hardy, N. Shimizu, T. Strzalkowski, L. Ting, G. B. Wise and X. Zhang. Summarizing large document sets using concept-based clustering. In *Proc. of the Human Language Technology Conference 2002*, (San Diego, California, United States, 2002), 222-227.

L. L. Hill. Core elements of digital gazetteers: placenames, categories, and footprints. In *Proc. of the 4th European Conference on Digital Libraries*, (Lisbon, Portugal, 2000), 280-290.

C. D. Holley and D. F. Dansereau. *Spatial learning strategies: Techniques, applications, and related issues*. Academic Press, Orlando, Florida, 1984.

C. Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proc. of the Workshop on Text Summarization Branches Out*, (Barcelona, Spain, 2004).

D. Lin and P. Pantel. Concept discovery from text. In *Proc. of the 19th International Conference on Computational Linguistics*, (Taipei, Taiwan, 2002), 1-7.

National Research Council. *National Science Education Standards*. National Academy Press, Washington, DC, 1996.

R. Navigli and P. Velardi. Learning domain ontologies from document warehouses and dedicated websites. *Computational Linguistics*, *30*, *2* (2004), 151-179.

A. M. O'Donnell, D. F. Dansereau and R. H. Hall. Knowledge maps as scaffolds for cognitive processing. *Educational Psychology Review*, *14*, *1* (2002), 71-86.

Project 2061. *Benchmarks for science literacy*. Oxford University Press, New York, New York, United States, 1993.

D. R. Radev, H. Jing and M. Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proc. of the ANLP/NAACL 2000 Workshop on Summarization*, (2000), 21-30.

T. Sumner, S. Bhushan, F. Ahmad and Q. Gu. Designing a language for creating conceptual browsing interfaces for digital libraries. In *Proc. of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, (Houston, Texas, 2003), 258-260.

A. Zouaq, R. Nkambou and C. Frasson. Learning a domain ontology in the Knowledge Puzzle project. In *Proc. of the Fifth International Workshop on Ontologies and Semantic Web for E-Learning*, (Marina del Rey, California, 2007).

Dialect Classification for online podcasts fusing Acoustic and Language based Structural and Semantic Information

Rahul Chitturi, John. H.L. Hansen[1]
Center for Robust Speech Systems(CRSS)
Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
Richardson, Texas 75080, U.S.A
{rahul.ch@student, john.hansen@}utdallas.edu

Abstract

The variation in speech due to dialect is a factor which significantly impacts speech system performance. In this study, we investigate effective methods of combining acoustic and language information to take advantage of (i) speaker based *acoustic* traits as well as (ii) content based word selection across the *text* sequence. For acoustics, a GMM based system is employed and for text based dialect classification, we proposed n-gram language models combined with Latent Semantic Analysis (LSA) based dialect classifiers. The performance of the individual classifiers is established for the three dialect family case (DC rates vary from 69.1%-72.4%). The final combined system achieved a DC accuracy of 79.5% and significantly outperforms the baseline acoustic classifier with a relative improvement of 30%, confirming that an integrated dialect classification system is effective for American, British and Australian dialects.

1 Introduction

Automatic Dialect Classification has recently gained substantial interest in the speech processing community (Gray and Hansen, 2005; Hansen et al., 2004; NIST LRE 2005). Dialect classification systems have been employed to improve the performance for Automatic Speech Recognition (ASR) by employing dialect dependent acoustic and language models (Diakoloukas et al., 1997) and for Rich Indexing of Spoken Document Retrieval Systems(Gray and Hansen 2005). (Huang and Hansen, 2005; 2006) focused on identifying pronunciation differences for dialect classification. In this study, unsupervised MFCC based GMM classifiers are employed for pronunciation modeling. However, English dialects differ in many ways other than pronunciation like Word Selection and Grammar, which cannot be modeled using frame based GMM acoustic information. For example, word selection differences between UK and US dialects such as - "lorry" vs. "truck", "lift", vs. "elevator", etc. Australian English has its own lexical terms such as tucker (food), outback (wilderness), etc (John Laver, 1994). N-gram language models are employed to address these problems. One additional factor in which dialects differ is in Semantics. For example, *momentarily* which means for a *moments duration* (UK) vs. *in a minute or any minute now* (US). The sentence "*This flight will be leaving momentarily"* could represent different time duration in US vs. UK dialects (John Laver, 1994). Latent Semantic Analysis is a technique that can distinguish these differences (Landauer et al.,1998). LSA has been shown to be effective for NLP based problems but has yet to be applied for dialect classification. Therefore, we develop an approach that uses a combination with n-gram language modeling and LSA processing to achieve effective language based dialect classification accuracy. Sec 4 explains the baseline acoustic classifier. Language classifiers are described in Sec 5 and the results which are presented in Sec 6 affirm that combining various sources of information significantly outperforms the traditional (or individual) techniques used for dialect classification.

[1]This project was funded by AFRL under a subcontract to RADC Inc. under FA8750-05-C-0029

2 Online Podcast Database

The speech community has no formal corpus of audio and text across dialects of common languages that could address the problems discussed in Sec.1. It was suggested in (Huang and Hansen, 2007) that it is more probable to observe semantic differences in the spontaneous text and speech rather than formal newspapers or prepared speeches since they must transcend dialects of a language (Hasegawa-Johnson and Levinson, 2006; Antoine 1996). Therefore, we collected a database from web based online podcasts of interviews where people talk spontaneously. All these are already been transcribed in order to separate text and audio structure and to temporarily set aside automatic speech recognition (ASR) error. These podcasts are not transcribed with an exact word to

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 21–24,
Columbus, Ohio, USA, June 2008.

word match but they match the audio to an extent that include what the speakers intended to say. The language and Acoustic statistics of this database are described in Sec 2.1, and 2.2.

2.1 Language Statistics

Huang and Hansen observed that the best dialect classification accuracy for N-gram classification requires at least 300 text words to obtain reasonable performance (Huang and Hansen, 2007). So, these interviews are segmented into blocks of text with an average text of 300 words. Table 1 summarizes the text material for three family-tree branches of English, containing 474k words and 1325 documents.

Dialect	No.of words	No. of Documents	
		Train	Test
US English	200k	383	158
UK English	154k	288	122
AU English	120k	233	141

Table 1: Language Statistics

2.2 Acoustic Statistics

We note that the data collected from online podcasts is not well structured. The audio data is segmented into smaller audio segment files since we are interested in 300 word blocks. Since the collection of dialect podcasts are collected from a wide range of online sources, we assume that channel effects and recording conditions are normalized across these three dialects. We also note that there is no speaker overlap between the test and train data. Therefore, there are no additional acoustic clues other than dialect. Table 2 summarizes the acoustic content of the corpus with 231 speakers and 13.5 hrs of audio.

Dialect	Males	Females	No. of Hours	
			Train	Test
US English	48	37	3.2	1.7
UK English	40	32	2.3	1
AU English	36	38	3.3	2

Table 2: Acoustic Statistics

3 System Architecture

The system architecture is shown in Fig 1, which consists of two main system phases for acoustic and language classifiers. MFCC based classifiers are used for acoustic modeling, while for language modeling, we use a combination of n-gram language modes and LSA classifiers. In the final phase, we combine the acoustic and language classifiers into our final dialect classifier. To construct the overall system, we first train the individual classifiers, and then set the weights of the hybrid classifiers using a greedy strategy to form the overall decision.

4 Baseline Acoustic Dialect Classification

GMM based acoustic classification is a popular method for text-independent dialect classification (Huang and Hansen, 2006) and therefore it is used as a baseline for our system. Fig. 2 shows the block diagram of the baseline gender-independent MFCC based GMM training system with 600 mixtures for each dialect. While testing, the incoming audio is classified as a particular dialect based on the maximum posterior probability measure over all the Gaussian Mixture Models. Mixture and frame selection based techniques as well as SVM-GMM hybrid techniques have been considered for dialect classification (Chitturi and Hansen, 2007). In order to assess the improvement by leveraging audio and text, we did not include these audio classification improvements in this study.

5 Dialect Classification using Language

As shown in Fig 1, the language based dialect classification module has two distinct classifiers. We describe in detail the n-gram and LSA based classifiers in the sections 5.1 and 5.2

5.1 N-gram based dialect classification

It is assumed that the text document is composed of many sentences. Each sentence can be regarded as a sequence of words **W**. The probability of generating **W** is given by $P(W|D) = P(w_1, w_2, \dots, w_m|D)$. Assuming the probability depends on the previous n words is $P(W|D) = \prod_{i=1}^{m} P(w_i|w_{i-n+1}, \dots, w_{i-1}, D)$ where m is the number of words in **W,** w_i is the word and D™ {UK, US, AU) is the dialect specific language model. The n-gram probabilities are calculated from occurrence counting. The final classification decision is given by C=$argmax_D \prod_{W\epsilon\varphi} P(W|D)$, where φ is a set of sentences in a document and D ™ {UK, US, AU}. In this study, we use the derivative measure of the cross entropy known as the test set perplexity for dialect classification. If the word sequence is sufficiently long, the cross entropy of the word sequence **W** is approximated as $H(W|D) = -\frac{1}{m} log_2 P(W|D)$. The perplexity of the test word sequence **W** as it relates to the language model D is $PP(W|D) = 2^{H(W|D)} = P(W|D)^{-1/m}$.The perplexity of the test word sequence is the generalization capability of the language model. The smaller the perplexity, the better

the language model generalizes to the test word sequence. The final classification decision is, C= $argmax_D \prod_{W \epsilon \varphi} P(W|D)^{-1/m}$, where is the set of sentences in a document, D ™ {UK, US, AU}.

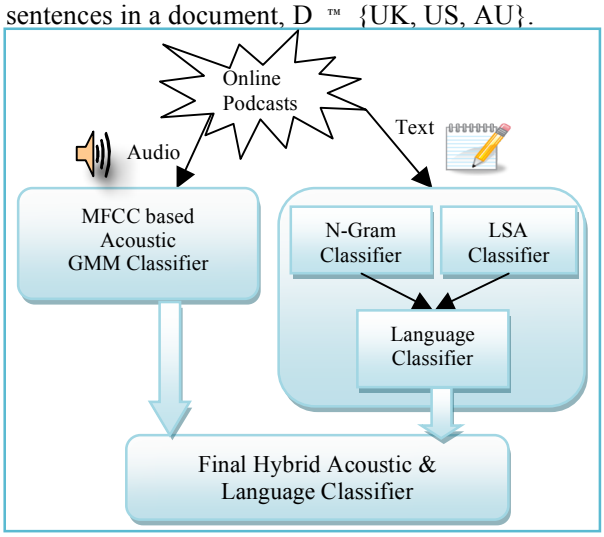


Figure 1: Proposed architecture

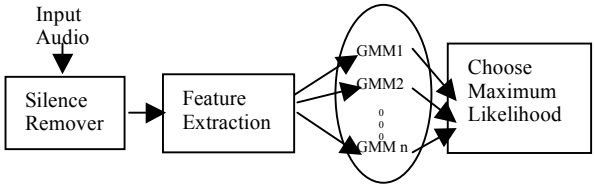


Figure 2: Baseline GMM based dialect classification

5.2 Latent Semantic Analysis for Dialect ID

One approach used to address topic classification problems has been latent semantic analysis (LSA), which was first explored for document indexing in (Deerwester et al., 1990). This addresses the issues of synonymy - many ways to refer to the same idea and polysemy – words having more than one distinct meaning. These two issues present problems for dialect classification as two conversations about a topic need not contain the same words and conversely two conversations about different topics may contain the same words but with different intended meanings. In order to find a different feature space which avoids these problems, singular value decomposition (SVD) is performed to derive orthogonal vector representations of the documents. SVD uses eigen-analysis to derive linearly independent directions of the original term by document matrix **A** whose columns correspond to the number of dialects, while the rows correspond to the words/terms in the entire text database. SVD decomposes this original term document matrix **A**, into three other matrices: $\mathbf{A=U*S*V^T}$, where the columns of **U** are the eigenvectors of $\mathbf{AA^T}$ (left eigenvectors), **S** is a diagonal matrix, whose diagonal elements are the singular values of **A,** and the columns of **V** are the eigenvectors of $\mathbf{A^TA}$(called right eigenvectors). The new dialect vector coordinates in this reduced 3 dimensional space are the rows of **V.** The coordinates of the test utterance is given by $\mathbf{q1=q^T*U*S^{-1}}$**.** The test utterance is then classified as a particular dialect based on the scores, given by the cosine similarity measure as $d_{best} = argmax_{d_i} \frac{(q_1.d_i)}{|q_1||d_i|}$, where d_i is one of the three dialects.

6 Results and Discussion

All evaluations presented in this section were conducted on the online podcast database described in the section 2. The first row of Table 3 shows the performance of the N-gram LM based dialect classification (69.1% avg. performance). From this we observe that this approach is good for US and UK, but not as effective for AU family dialect classification, with AU being confused with UK. The performance of the LSA based dialect classification is shown in the second row of Table 3. This classifier is consistent over all the dialects with better performance than the N-gram LM approach. There is more semantic similarity of US with AU than UK (24% vs 5% - false positives), while UK has a balanced semantic error with US and AU. This implies that there is more semantic information in these dialects than text sequence structure.

Next, the N-gram and the LSA classifiers are combined using optimal weights based on a greedy approach. Fig. 3 shows the performance of this hybrid classifier with respect to the weights of the individual classifiers (N-gram vs LSA: 0→all N-gram, 50→0.5 N-gram and 0.5 LSA, 100→ all LSA). After setting the optimal weights 0.18 to LSA and 0.82 to N-gram classifier, the hybrid classifier is seen to be consistent and better than the individual classifiers (Table 3: row 3 vs row2/row1). Performance of the hybrid classifier is not as good as the LSA classifier for AU classification, but significantly better for classification of US and UK. The hybrid classifier is better in all cases when compared to the N-gram classifier, with an overall average improvement of 7.3% absolute. The fourth row in Table 3 shows the performance of acoustic based dialect classification which is as good as the language based dialect classification, but it is noted that performance is poor for UK classification. It is expected that the type of errors made by text (word selection), semantics and acoustic space

will have differences and therefore we combine these acoustical and language classifiers as shown in Fig1. The overall performance of the proposed approach, combining the acoustic and language information, is better than the individual classifiers (Row 3 and Row 4 vs. Row 5 of Table 3). Even though the performance for US is reduced from 87.2% to 86.38%, the classification of UK is improved significantly from 54% to 74%. This shows that this approach is more consistent with accuracy that outperforms traditional acoustic classifiers with a relative improvement of 30%. With respect to a language only classifier, this hybrid classifier is better in all the cases.

7 Conclusions

In this study, we have developed a dialect classification (DC) algorithm that addresses family branch DC for English (US, UK, AU), by combining GMM based acoustic, and text based N-gram LM and LSA language information. In this paper, we employed LSA in combination with N-gram language models and GMM acoustic models to improve DC accuracy. The performance of the individual classifiers were shown to vary from 69.1%-72.4%. The final combined system achieves a DC accuracy of 79.5% and significantly outperformed the baseline acoustic classifier with a relative improvement of 30%, confirming that an integrated dialect classification system employing GMM based acoustic and N-gram LM, LSA based language information is effective for dialect classification.

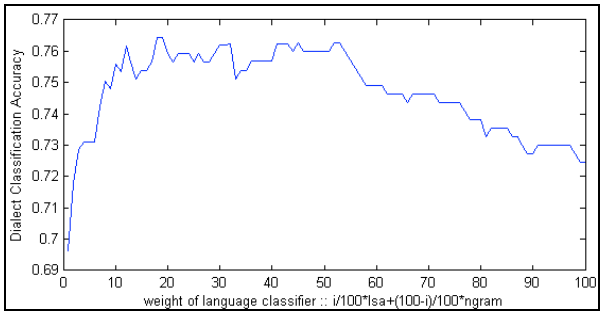


Figure 3: Language classifier

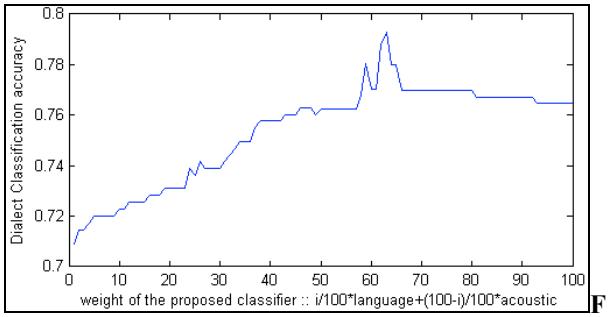


Figure 4: Acoustic + Language classifier

Accuracy→ Methods↓	US	UK	AU	Overall
N-Gram LM Classifier	75.2%	71.2%	60.7%	69.1%
Latent Semantic (LSA) Classifier	70.2%	68.5%	78.7%	72.47%
N-Gram+ LSA (Based on Text)	79.3%	74.6%	75.4%	76.4%
Acoustic GMM Classifier	87.2%	54.0%	73.3%	71.6%
Acoustic GMM + N-gram+ LSA	86.4%	74.6%	77.0%	79.5%

Table 3: Performance of classifiers on Dialect-ID

References

Diakoloukas, V.; Neumeyer, L.; Kaja, J.; 1997. "*Development of dialect-specific speech recognizers using adaptation methods"* IEEE- ICASSP

John Laver; 1994. "*Principles of Phonetics"*. Cambridge University Press, Cambridge, UK.

Gray, S.; Hansen, J.H.L; 2005. "*An integrated approach to the detection and classification of /dialects for a spoken document retrieval system"* IEEE- ASRU

Huang R; Hansen J.H.L.; 2005. *"Dialect/Accent Classification via Boosted Word Modeling,"* IEEE-ICASSP

Landauer,T.K., Foltz,P.W., & Laham,D; 1998. *" Introduction to Latent Semantic Analysis "* Discourse Processes, 25, 259-284.

Huang R; Hansen J.H.L.2007 *"Dialect Classification on Printed Text using Perplexity Measure and Conditional Random Fields,"* IEEE- ICASSP

Hasegawa-Johnson M, Levinson S.E, 2006 *"Extraction of pragmatic and semantic salience from spontaneous spoken English "* Speech Comm. Vol. 48(3-4)

Antoine, J.-Y 1996 *" Spontaneous speech and natural language processing. ALPES: a robust semantic-led parser "* ICSLP

Deerwester.S. et al.1990.*" Indexing by latent semantic anlysis "* Journal of American Society of Information Science, 391–407.

Chitturi. R, Hansen J.H.L, 2007.*" Multi stream based Dialect classification using SVM-GMM hybrids"* IEEE-ASRU

Huang. R, Hansen J.L.H.; 2006. *"Gaussian Mixture Selection and Data Selection for Unsupervised Spanish Dialect Classification"* ICSLP

Hansen J.H.L., Yapanel.U, Huang.R., Ikeno.A..; 2004. *"Dialect Anal'ysis and Modeling for Automatic Classification"* ICSLP

NIST- LRE 2005,*" Language Recognition Evaluation"*

The Complexity of Phrase Alignment Problems

John DeNero and Dan Klein
Computer Science Division, EECS Department
University of California at Berkeley
{denero, klein}@cs.berkeley.edu

Abstract

Many phrase alignment models operate over the combinatorial space of *bijective phrase alignments*. We prove that finding an optimal alignment in this space is NP-hard, while computing alignment expectations is #P-hard. On the other hand, we show that the problem of finding an optimal alignment can be cast as an integer linear program, which provides a simple, declarative approach to Viterbi inference for phrase alignment models that is empirically quite efficient.

1 Introduction

Learning in phrase alignment models generally requires computing either Viterbi phrase alignments or expectations of alignment links. For some restricted combinatorial spaces of alignments—those that arise in ITG-based phrase models (Cherry and Lin, 2007) or local distortion models (Zens et al., 2004)—inference can be accomplished using polynomial time dynamic programs. However, for more permissive models such as Marcu and Wong (2002) and DeNero et al. (2006), which operate over the full space of *bijective phrase alignments* (see below), no polynomial time algorithms for exact inference have been exhibited. Indeed, Marcu and Wong (2002) conjectures that none exist. In this paper, we show that Viterbi inference in this full space is NP-hard, while computing expectations is #P-hard.

On the other hand, we give a compact formulation of Viterbi inference as an integer linear program (ILP). Using this formulation, exact solutions to the Viterbi search problem can be found by highly optimized, general purpose ILP solvers. While ILP is of course also NP-hard, we show that, empirically, exact solutions are found very quickly for most problem instances. In an experiment intended to illustrate the practicality of the ILP approach, we show speed and search accuracy results for aligning phrases under a standard phrase translation model.

2 Phrase Alignment Problems

Rather than focus on a particular model, we describe four problems that arise in training phrase alignment models.

2.1 Weighted Sentence Pairs

A *sentence pair* consists of two word sequences, $\mathbf{e}$ and $\mathbf{f}$. A set of phrases $\{e_{ij}\}$ contains all spans e_{ij} from between-word positions i to j of $\mathbf{e}$. A *link* is an aligned pair of phrases, denoted (e_{ij}, f_{kl}).[1]

Let a *weighted sentence pair* additionally include a real-valued function $\phi : \{e_{ij}\} \times \{f_{kl}\} \rightarrow \mathbb{R}$, which scores links. $\phi(e_{ij}, f_{kl})$ can be sentence-specific, for example encoding the product of a translation model and a distortion model for (e_{ij}, f_{kl}). We impose no additional restrictions on ϕ for our analysis.

2.2 Bijective Phrase Alignments

An alignment is a set of links. Given a weighted sentence pair, we will consider the space of bijective phrase alignments $\mathcal{A}$: those $\mathbf{a} \subset \{e_{ij}\} \times \{f_{kl}\}$ that use each word token in exactly one link. We first define the notion of a partition: $\sqcup_i S_i = T$ means S_i are *pairwise disjoint* and cover T. Then, we can formally define the set of bijective phrase alignments:

$$\mathcal{A} = \left\{ \mathbf{a} : \bigsqcup_{(e_{ij}, f_{kl}) \in \mathbf{a}} e_{ij} = \mathbf{e} \, ; \bigsqcup_{(e_{ij}, f_{kl}) \in \mathbf{a}} f_{kl} = \mathbf{f} \right\}$$

[1] As in parsing, the position between each word is assigned an index, where 0 is to the left of the first word. In this paper, we assume all phrases have length at least one: $j > i$ and $l > k$.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 25–28,
Columbus, Ohio, USA, June 2008.

Both the conditional model of DeNero et al. (2006) and the joint model of Marcu and Wong (2002) operate in $\mathcal{A}$, as does the phrase-based decoding framework of Koehn et al. (2003).

2.3 Problem Definitions

For a weighted sentence pair $(\mathbf{e}, \mathbf{f}, \phi)$, let the score of an alignment be the product of its link scores:

$$\phi(\mathbf{a}) = \prod_{(e_{ij}, f_{kl}) \in \mathbf{a}} \phi(e_{ij}, f_{kl}).$$

Four related problems involving scored alignments arise when training phrase alignment models.

OPTIMIZATION, $\mathcal{O}$: Given $(\mathbf{e}, \mathbf{f}, \phi)$, find the highest scoring alignment $\mathbf{a}$.

DECISION, $\mathcal{D}$: Given $(\mathbf{e}, \mathbf{f}, \phi)$, decide if there is an alignment $\mathbf{a}$ with $\phi(\mathbf{a}) \geq 1$.

$\mathcal{O}$ arises in the popular Viterbi approximation to EM (Hard EM) that assumes probability mass is concentrated at the mode of the posterior distribution over alignments. $\mathcal{D}$ is the corresponding decision problem for $\mathcal{O}$, useful in analysis.

EXPECTATION, $\mathcal{E}$: Given a weighted sentence pair $(\mathbf{e}, \mathbf{f}, \phi)$ and indices i, j, k, l, compute $\sum_{\mathbf{a}} \phi(\mathbf{a})$ over all $\mathbf{a} \in \mathcal{A}$ such that $(e_{ij}, f_{kl}) \in \mathbf{a}$.

SUM, $\mathcal{S}$: Given $(\mathbf{e}, \mathbf{f}, \phi)$, compute $\sum_{\mathbf{a} \in \mathcal{A}} \phi(\mathbf{a})$.

$\mathcal{E}$ arises in computing sufficient statistics for re-estimating phrase translation probabilities (E-step) when training models. The existence of a polynomial time algorithm for $\mathcal{E}$ implies a polynomial time algorithm for $\mathcal{S}$, because $\mathcal{A} = \bigcup_{j=1}^{|\mathbf{e}|} \bigcup_{k=0}^{|\mathbf{f}|-1} \bigcup_{l=k+1}^{|\mathbf{f}|} \{\mathbf{a} : (e_{0j}, f_{kl}) \in \mathbf{a}, \mathbf{a} \in \mathcal{A}\}$.

3 Complexity of Inference in $\mathcal{A}$

For the space $\mathcal{A}$ of bijective alignments, problems $\mathcal{E}$ and $\mathcal{O}$ have long been suspected of being NP-hard, first asserted but not proven in Marcu and Wong (2002). We give a novel proof that $\mathcal{O}$ is NP-hard, showing that $\mathcal{D}$ is NP-complete by reduction from SAT, the boolean satisfiability problem. This result holds despite the fact that the related problem of finding an optimal matching in a weighted bipartite graph (the ASSIGNMENT problem) is polynomial-time solvable using the Hungarian algorithm.

3.1 Reducing Satisfiability to $\mathcal{D}$

A reduction proof of NP-completeness gives a construction by which a known NP-complete problem can be solved via a newly proposed problem. From a SAT instance, we construct a weighted sentence pair for which alignments with positive score correspond exactly to the SAT solutions. Since SAT is NP-complete and our construction requires only polynomial time, we conclude that $\mathcal{D}$ is NP-complete.[2]

SAT: Given vectors of boolean variables $\mathbf{v} = (v)$ and propositional clauses[3] $\mathbf{C} = (C)$, decide whether there exists an assignment to $\mathbf{v}$ that simultaneously satisfies each clause in $\mathbf{C}$.

For a SAT instance $(\mathbf{v}, \mathbf{C})$, we construct $\mathbf{f}$ to contain one word for each clause, and $\mathbf{e}$ to contain several copies of the literals that appear in those clauses. ϕ scores only alignments from clauses to literals that satisfy the clauses. The crux of the construction lies in ensuring that no variable is assigned both *true* and *false*. The details of constructing such a weighted sentence pair $\text{wsp}(\mathbf{v}, \mathbf{C}) = (\mathbf{e}, \mathbf{f}, \phi)$, described below, are also depicted in figure 1.

1. $\mathbf{f}$ contains a word for each C, followed by an assignment word for each variable, $\text{assign}(v)$.
2. $\mathbf{e}$ contains $c(\ell)$ consecutive words for each literal ℓ, where $c(\ell)$ is the number of times that ℓ appears in the clauses.

Then, we set $\phi(\cdot, \cdot) = 0$ everywhere except:

3. For all clauses C and each satisfying literal ℓ, and each one-word phrase e in $\mathbf{e}$ containing ℓ, $\phi(e, f_C) = 1$. f_C is the one-word phrase containing C in $\mathbf{f}$.
4. The $\text{assign}(v)$ words in $\mathbf{f}$ align to longer phrases of literals and serve to consistently assign each variable by using up inconsistent literals. They also align to unused literals to yield a bijection. Let $e^k_{[\ell]}$ be the phrase in $\mathbf{e}$ containing all literals ℓ and k negations of ℓ. $f_{\text{assign}(v)}$ is the one-word phrase for $\text{assign}(v)$. Then, $\phi(e^k_{[\ell]}, f_{\text{assign}(v)}) = 1$ for $\ell \in \{v, \bar{v}\}$ and all applicable k.

[2] Note that D is trivially in NP: given an alignment $\mathbf{a}$, it is easy to determine whether or not $\phi(\mathbf{a}) \geq 1$.

[3] A clause is a disjunction of literals. A literal is a bare variable v_n or its negation $\bar{v}_n$. For instance, $v_2 \vee \bar{v}_7 \vee \bar{v}_9$ is a clause.

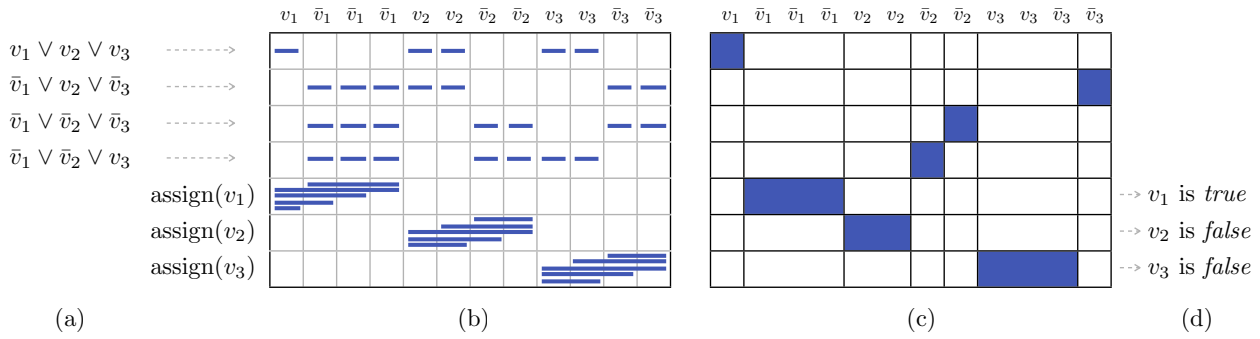


Figure 1: (a) The clauses of an example SAT instance with $\mathbf{v} = (v_1, v_2, v_3)$. (b) The weighted sentence pair wsp$(\mathbf{v}, \mathbf{C})$ constructed from the SAT instance. All links that have $\phi = 1$ are marked with a blue horizontal stripe. Stripes in the last three rows demarcate the alignment options for each assign(v_n), which consume all words for some literal. (c) A bijective alignment with score 1. (d) The corresponding satisfying assignment for the original SAT instance.

Claim 1. *If wsp*$(\mathbf{v}, \mathbf{C})$ *has an alignment* $\mathbf{a}$ *with* $\phi(\mathbf{a}) \geq 1$*, then* $(\mathbf{v}, \mathbf{C})$ *is satisfiable.*

Proof. The score implies that $\mathbf{f}$ aligns using all one-word phrases and $\forall a_i \in \mathbf{a}, \phi(a_i) = 1$. By condition 4, each $f_{\text{assign}(v)}$ aligns to all $\bar{v}$ or all v in $\mathbf{e}$. Then, assign each v to *true* if $f_{\text{assign}(v)}$ aligns to all $\bar{v}$, and *false* otherwise. By condition 3, each C must align to a satisfying literal, while condition 4 assures that all available literals are consistent with this assignment to $\mathbf{v}$, which therefore satisfies $\mathbf{C}$. □

Claim 2. *If* $(\mathbf{v}, \mathbf{C})$ *is satisfiable, then wsp*$(\mathbf{v}, \mathbf{C})$ *has an alignment* $\mathbf{a}$ *with* $\phi(\mathbf{a}) = 1$.

Proof. We construct such an alignment $\mathbf{a}$ from the satisfying assignment $\mathbf{v}$. For each C, we choose a satisfying literal ℓ consistent with the assignment. Align f_C to the first available ℓ token in $\mathbf{e}$ if the corresponding v is *true*, or the last if v is *false*. Align each $f_{\text{assign}(v)}$ to all remaining literals for v. □

Claims 1 and 2 together show that $\mathcal{D}$ is NP-complete, and therefore that $\mathcal{O}$ is NP-hard.

3.2 Reducing Perfect Matching to $\mathcal{S}$

With another construction, we can show that $\mathcal{S}$ is #P-hard, meaning that it is at least as hard as any #P-complete problem. #P is a class of counting problems related to NP, and #P-hard problems are NP-hard as well.

COUNTING PERFECT MATCHINGS, CPM
Given a bipartite graph G with $2n$ vertices, count the number of matchings of size n.

For a bipartite graph G with edge set $E = \{(v_j, v_l)\}$, we construct $\mathbf{e}$ and $\mathbf{f}$ with n words each, and set $\phi(e_{j-1\ j}, f_{l-1\ l}) = 1$ and 0 otherwise. The number of perfect matchings in G is the sum $\mathcal{S}$ for this weighted sentence pair. CPM is #P-complete (Valiant, 1979), so $\mathcal{S}$ (and hence $\mathcal{E}$) is #P-hard.

4 Solving the Optimization Problem

Although $\mathcal{O}$ is NP-hard, we present an approach to solving it using integer linear programming (ILP).

4.1 Previous Inference Approaches

Marcu and Wong (2002) describes an approximation to $\mathcal{O}$. Given a weighted sentence pair, high scoring phrases are linked together greedily to reach an initial alignment. Then, local operators are applied to hill-climb $\mathcal{A}$ in search of the maximum $\mathbf{a}$. This procedure also approximates $\mathcal{E}$ by collecting weighted counts as the space is traversed.

DeNero et al. (2006) instead proposes an exponential-time dynamic program to systematically explore $\mathcal{A}$, which can in principle solve either $\mathcal{O}$ or $\mathcal{E}$. In practice, however, the space of alignments has to be pruned severely using word alignments to control the running time of EM.

Notably, neither of these inference approaches offers any test to know if the optimal alignment is ever found. Furthermore, they both require small data sets due to computational expense.

4.2 Alignment via an Integer Program

We cast $\mathcal{O}$ as an ILP problem, for which many optimization techniques are well known. First, we in-

troduce binary indicator variables $a_{i,j,k,l}$ denoting whether $(e_{ij}, f_{kl}) \in \mathbf{a}$. Furthermore, we introduce binary indicators $e_{i,j}$ and $f_{k,l}$ that denote whether some $(e_{ij}, \cdot)$ or $(\cdot, f_{kl})$ appears in $\mathbf{a}$, respectively. Finally, we represent the weight function ϕ as a weight vector in the program: $w_{i,j,k,l} = \log \phi(e_{ij}, f_{kl})$.

Now, we can express an integer program that, when optimized, will yield the optimal alignment of our weighted sentence pair.

$$\max \quad \sum_{i,j,k,l} w_{i,j,k,l} \cdot a_{i,j,k,l}$$

$$\text{s.t.} \quad \sum_{i,j:i<x\leq j} e_{i,j} = 1 \qquad \forall x : 1 \leq x \leq |e| \quad (1)$$

$$\sum_{k,l:k<y\leq l} f_{k,l} = 1 \qquad \forall y : 1 \leq y \leq |f| \quad (2)$$

$$e_{i,j} = \sum_{k,l} a_{i,j,k,l} \qquad \forall i, j \quad (3)$$

$$f_{k,l} = \sum_{i,j} a_{i,j,k,l} \qquad \forall k, l \quad (4)$$

with the following constraints on index variables:

$$0 \leq i < |\mathbf{e}|, \quad 0 < j \leq |\mathbf{e}|, \quad i < j$$
$$0 \leq k < |\mathbf{f}|, \quad 0 < l \leq |\mathbf{f}|, \quad k < l \quad .$$

The objective function is $\log \phi(\mathbf{a})$ for $\mathbf{a}$ implied by $\{a_{i,j,k,l} = 1\}$. Constraint equation 1 ensures that the English phrases form a partition of **e** – each word in **e** appears in exactly one phrase – as does equation 2 for **f**. Constraint equation 3 ensures that each phrase in the chosen partition of **e** appears in exactly one link, and that phrases not in the partition are not aligned (and likewise constraint 4 for **f**).

5 Applications

The need to find an optimal phrase alignment for a weighted sentence pair arises in at least two applications. First, a generative phrase alignment model can be trained with Viterbi EM by finding optimal phrase alignments of a training corpus (approximate E-step), then re-estimating phrase translation parameters from those alignments (M-step).

Second, this is an algorithm for *forced decoding*: finding the optimal phrase-based derivation of a particular target sentence. Forced decoding arises in online discriminative training, where model updates are made toward the most likely derivation of a gold translation (Liang et al., 2006).

Sentences per hour on a four-core server	20,000
Frequency of optimal solutions found	93.4%
Frequency of ϵ-optimal solutions found	99.2%

Table 1: The solver, tuned for speed, regularly reports solutions that are within 10^{-5} of optimal.

Using an off-the-shelf ILP solver,[4] we were able to quickly and reliably find the globally optimal phrase alignment under $\phi(e_{ij}, f_{kl})$ derived from the Moses pipeline (Koehn et al., 2007).[5] Table 1 shows that finding the optimal phrase alignment is accurate and efficient.[6] Hence, this simple search technique effectively addresses the intractability challenges inherent in evaluating new phrase alignment ideas.

References

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *NAACL Workshop on Statistical Machine Translation*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL*.

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

Leslie G. Valiant. 1979. The complexity of computing the permanent. In *Theoretical Computer Science 8*.

Richard Zens, Hermann Ney, Taro Watanabeand, and E. Sumita. 2004. Reordering constraints for phrase based statistical machine translation. In *Coling*.

[4] We used Mosek: `www.mosek.com`.

[5] $\phi(e_{ij}, f_{kl})$ was estimated using the relative frequency of phrases extracted by the default Moses training script. We evaluated on English-Spanish Europarl, sentences up to length 25.

[6] ILP solvers include many parameters that trade off speed for accuracy. Substantial speed gains also follow from explicitly pruning the values of ILP variables based on prior information.

Novel Semantic Features for Verb Sense Disambiguation

Dmitriy Dligach
The Center for Computational Language and Education Research
1777 Exposition Drive
Boulder, Colorado 80301
Dmitriy.Dligach@colorado.edu

Martha Palmer
Department of Linguistics
University of Colorado at Boulder
295 UCB
Boulder, Colorado 80309
Martha.Palmer@colorado.edu

Abstract

We propose a novel method for extracting semantic information about a verb's arguments and apply it to Verb Sense Disambiguation (VSD). We contrast this method with two popular approaches to retrieving this information and show that it improves the performance of our VSD system and outperforms the other two approaches

1 Introduction

The task of Verb Sense Disambiguation (VSD) consists in automatically assigning a sense to a verb (target verb) given its context. In a supervised setting, a VSD system is usually trained on a set of pre-labeled examples; the goal of this system is to tag unseen examples with a sense from some sense inventory.

An automatic VSD system usually has at its disposal a diverse set of features among which the semantic features play an important role: verb sense distinctions often depend on the distinctions in the semantics of the target verb's arguments (Hanks, 1996). Therefore, some method of capturing the semantic knowledge about the verb's arguments is crucial to the success of a VSD system.

The approaches to obtaining this kind of knowledge can be based on extracting it from electronic dictionaries such as WordNet (Fellbaum, 1998), using Named Entity (NE) tags, or a combination of both (Chen, 2005). In this paper, we propose a novel method for obtaining semantic knowledge about words and show how it can be applied to VSD. We contrast this method with the other two approaches and compare their performances in a series of experiments.

2 Lexical and Syntactic Features

We view VSD as a supervised learning problem, solving which requires three groups of features: lexical, syntactic, and semantic. Lexical features include all open class words; we extract them from the target sentence and the two surrounding sentences. We also use as features two words on the right and on the left of the target verb as well as their POS tags. We extract syntactic features from constituency parses; they indicate whether the target verb has a subject/object and what their head words and POS tags are, whether the target verb is in a passive or active form, whether the target verb has a subordinate clause, and whether the target verb has a PP adjunct. Additionally, we implement several new syntactic features, which have not been used in VSD before: the path through the parse tree from the target verb to the verb's arguments and the subcategorization frame, as used in semantic role labeling.

3 Semantic Features

Consider the verb *prepare* for which our sense inventory defines two senses: (1) to put together, assemble (e.g. *He is going to prepare breakfast for the whole crowd*; *I haven't prepared my lecture*

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 29–32,
Columbus, Ohio, USA, June 2008.

yet); (2) to make ready (e.g. *She prepared the children for school every morning*). Knowing the semantic class of the objects *breakfast, lecture* and *children* is the decisive factor in distinguishing the two senses and facilitates better generalization from the training data. One way to obtain this knowledge is from WordNet (WN) or from the output of a NE-tagger. However, both approaches suffer from the same limitation: they collapse multiple semantic properties of nouns into a finite number of predefined static classes. E.g., the most immediate hypernym of *breakfast* in WN is *meal*, while the most immediate hypernym of *lecture* is *address*, which makes these two nouns unrelated. Yet, *breakfast* and *lecture* are both social events which share some semantic properties: they both can be *attended, hosted, delivered, given, held, organized* etc. To discover these class-like descriptions of nouns, one can observe which verbs take these nouns as objects. E.g. *breakfast* can serve as the object of *serve, host, attend, and cook* which are all indicative of *breakfast's* semantic properties.

Given a noun, we can dynamically retrieve other verbs that take that noun as an object from a dependency-parsed corpus; we call this kind of data **Dynamic Dependency Neighbors** (DDNs) because it is obtained dynamically and based on the dependency relations in the neighborhood of the noun of interest. The top 50[1] DDNs can be viewed as a reliable inventory of semantic properties of the noun. To collect this data, we utilized two resources: (1) MaltParser (Nivre, 2007) – a high-efficiency dependency parser; (2) English Gigaword – a large corpus of 5.7M news articles. We preprocessed Gigaword with MaltParser, extracted all pairs of nouns and verbs that were parsed as participants of the object-verb relation, and counted the frequency of occurrence of all the unique pairs. Finally, we indexed the resulting records of the form <frequency, verb, object> using the Lucene[2] indexing engine.

As an example, consider four nouns: *dinner, breakfast, lecture, child*. When used as the objects of *prepare*, the first three of them correspond to the instances of the sense 1 of *prepare*; the fourth one corresponds to an instance of the sense 2. With the help of our index, we can retrieve their DDNs. There is a considerable overlap among the DDNs of the first three nouns and a much smaller overlap between *child* and the first three nouns. E.g., *dinner* and *breakfast* have 34 DDNs in common, while *dinner* and *child* only share 14.

Once we have set up the framework for the extraction of DDNs, the algorithm for applying them to VSD is straightforward: (1) find the noun object of the ambiguous verb (2) extract the DDNs for that noun (3) sort the DDNs by frequency and keep the top 50 (4) include these DDNs in the feature vector so that each of the extracted verbs becomes a separate feature.

4 Relevant Work

At the core of our work lies the notion of distributional similarity (Harris, 1968), which states that similar words occur in similar contexts. In various sources, the notion of context ranges from bag-of-words-like approaches to more structured ones in which syntax plays a role. Schutze (1998) used bag-of-words contexts for sense discrimination. Hindle (1990) grouped nouns into thesaurus-like lists based on the similarity of their syntactic contexts. Our approach is similar with the difference that we do not group noun arguments into finite categories, but instead leave the category boundaries blurry and allow overlaps.

The DDNs are essentially a form of world knowledge which we extract automatically and apply to VSD. Other researches attacked the problem of unsupervised extraction of world knowledge: Schubert (2003) reports a method for extracting general facts about the world from tree-banked Brown corpus. Lin and Pantel in (2001) describe their DIRT system for extraction of paraphrase-like inference rules.

5 Evaluation

We selected a subset of the verbs annotated in the OntoNotes project (Chen, 2007) that had at least 50 instances. The resulting data set consisted of 46,577 instances of 217 verbs. The predominant sense baseline for this data is 68%. We used

[1] In future, we will try to optimize this parameter
[2] Available at http://lucene.apache.org/

libsvm[3] for classification. We computed the accuracy and error rate using 5-fold cross-validation.

5.1 Experiments with a limited set of features

The main objective of this experiment was to isolate the effect of the novel semantic features we proposed in this paper, i.e. the DDN features. Toward that goal, we stripped our system of all the features but the most essential ones to investigate whether the DDN features would have a clearly positive or negative impact on the system performance. Lexical features are the most essential to our system: a model that includes only the lexical features achieves an accuracy of 80.22, while the accuracy of our full-blown VSD system is 82.88%[4]. Since the DDN features have no effect when the object is not present, we identified 18,930 instances where the target verb had an object (about 41% of all instances) and used only them in the experiment.

We built three models that included (1) the lexical features only (2) the lexical and the DDN features (3) the lexical and the object features. The object features consist of the head word of the NP object and the head word's POS tag. The object is included since extracting the DDN features requires knowledge of the object; therefore the performance of a model that only includes lexical features cannot be considered a fair baseline for studying the effect of the DDN features. Results are in Table 4.

Features Included in Model	Accuracy, %	Error Rate, %
Lexical	78.95	21.05
Lexical + Object	79.34	20.66
Lexical + DDN	82.40	17.60

Table 4. Experiments with object instances

As we see, the model that includes the DDN features performs more than 3 percentage points better than the model that only includes the object features (approximately 15% reduction in error rate). Also, based on the comparison of the performance of the "lexical features only" and the "lexical + DDN" models, we can claim that the knowledge of the DDNs provides richer semantic knowledge than just the knowledge of the object's head word.

5.2 Integrating the DDN features into a full-fledged VSD system

The objective of this experiment was to investigate whether the DDN features improve the performance of a full-fledged VSD system. We built two models which consisted of (1) the entire set of features (2) all the features of the first model excluding the DDN features. The entire data set (46K instances) participated in the experiment. Results are in Table 5.

Features Included in Model	Accuracy, %	Error Rate, %
All Features – DDN	82.38	17.62
All Features	82.88	17.12

Table 5. Performance of the full-fledged VSD system

The DDN features improved performance by 0.5% (3% drop in error rate). The difference between the accuracies is statistically significant (p=0.05).

5.3 Relative Contribution of Various Semantic Features

The goal of this experiment was to study the relative contribution of various semantic features to the performance of our VSD system. We built five models each of which, in addition to the lexical and syntactic features, included only certain type(s) of semantic feature: (1) WN (2) NE (3) WN and NE (4) DDN (5) no semantic features (baseline). All 46K instances participated in the experiment. The results are shown in Table 6.

Features Included in Model	**Accuracy, %**	**Error Rate, %**
Lexical + Syntactic	81.82	18.18
Lexical + Syntactic + WN	82.34	17.60
Lexical + Syntactic + NE	82.01	17.99
Lexical + Syntactic + WN + NE	82.38	17.62
Lexical + Syntactic + DDN	82.97	17.03

Table 6. Relative Contribution of Semantic Features

The DDN features outperform the other two types of semantic features used separately and in conjunction. The difference in performance is statistically significant (p=0.05).

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4] Given this high baseline, we include error rate when reporting the results of the experiments as it is more informative

6 Discussion and Conclusion

As we saw, the novel semantic features we proposed are beneficial to the task of VSD: they resulted in a decrease in error rate from 3% to 15%, depending on the particular experiment. We also discovered that the DDN features contributed twice as much as the other two types of semantic features combined: adding the WN and NE features to the baseline resulted in about a 3% decrease in error rate, while adding the DDN features caused a more than 6% drop.

Our results suggest that DDNs duplicate the effect of WN and NE: our system achieved the same performance when all three types of semantic features were used and when we discarded WN and NE features and kept only the DDNs. This finding is important because such resources as WN and NE-taggers are domain and language specific while the DDNs have the advantage of being obtainable from a large collection of texts in the domain or language of interest. Thus, the DDNs can become a crucial part of building a robust VSD system for a resource-poor domain or language, given a high-accuracy parser.

7 Future Work

In this paper we only experimented with verbs' objects, however the concept of DDNs can be easily extended to other arguments of the target verb. Also, we only utilized the object-verb relation in the dependency parses, but the range of potentially useful relations does not have to be limited only to it. Finally, we used as features the 50 most frequent verbs that took the noun argument as an object. However, the raw frequency is certainly not the only way to rank the verbs; we plan on exploring other metrics such as Mutual Information.

Acknowledgements

We gratefully acknowledge the support of the National Science Foundation Grant NSF-0715078, Consistent Criteria for Word Sense Disambiguation, and the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022, a subcontract from the BBN-AGILE Team. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We also thank our colleagues Rodney Nielsen and Philipp Wetzler for parsing English Gigaword with MaltParser.

References

Jinying Chen, Dmitriy Dligach and Martha Palmer. 2007. Towards Large-scale High-Performance English Verb Sense Disambiguation by Using Linguistically Motivated Features. In *International Conference on Semantic Computing*. Issue , 17-19.

Jinying Chen and Martha Palmer. 2005. Towards Robust High Performance Word Sense Disambiguation of English Verbs Using Rich Linguistic Features. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, Korea.

Christiane Fellbaum. 1998. WordNet - an Electronic Lexical Database. The MIT Press, Cambridge, Massachusetts, London, UK.

Patrick Hanks, 1996. Contextual Dependencies and Lexical Sets. In *The Int. Journal of Corpus Linguistics*, 1:1

Zelig S. Harris. 1968. Mathematical Structures of Language. New York. Wiley.

Donald Hindle. 1990. Noun Classification from Predicate-Argument Structures. In *Proceedings of the 28th Annual Meeting of Association for Computational Linguistics*. Pages 268-275

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining*. pp. 323-328. San Francisco, CA.

Joakim Nivre, Johan Hall, Jens Nilsson, et. al. MaltParser: A language-independent system for data-driven dependency parsing. 2007. In *Natural Language Engineering*, 13(2), 95-135.

Lenhart Schubert and Matthew Tong, Extracting and evaluating general world knowledge from the Brown corpus. 2003. In *Proc. of the HLT/NAACL Workshop on Text Meaning*, May 31, Edmonton, Alberta, Canada.

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

Icelandic Data Driven Part of Speech Tagging

Mark Dredze
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
mdredze@cis.upenn.edu

Joel Wallenberg
Department of Linguistics
University of Pennsylvania
Philadelphia, PA 19104
joelcw@babel.ling.upenn.edu

Abstract

Data driven POS tagging has achieved good performance for English, but can still lag behind linguistic rule based taggers for morphologically complex languages, such as Icelandic. We extend a statistical tagger to handle fine grained tagsets and improve over the best Icelandic POS tagger. Additionally, we develop a case tagger for non-local case and gender decisions. An error analysis of our system suggests future directions.

1 Introduction

While part of speech (POS) tagging for English is very accurate, languages with richer morphology demand complex tagsets that pose problems for data driven taggers. In this work we consider Icelandic, a language for which a linguistic rule-based method is the current state of the art, indicating the difficulty this language poses to learning systems. Like Arabic and Czech, other morphologically complex languages with large tagsets, Icelandic can overwhelm a statistical tagger with ambiguity and data sparsity.

Shen et al. (2007) presented a new framework for bidirectional sequence classification that achieved the best POS score for English. In this work, we evaluate their tagger on Icelandic and improve results with extensions for fine grained annotations. Additionally, we show that good performance can be achieved using a strictly data-driven learning approach without external linguistic resources (morphological analyzer, lexicons, etc.). Our system achieves the best performance to date on Icelandic, with insights that may help improve other morphologically rich languages.

After some related work, we describe Icelandic morphology followed by a review of previous approaches. We then apply a bidirectional tagger and extend it for fine grained languages. A tagger for case further improves results. We conclude with an analysis of remaining errors and challenges.

2 Related Work

Previous approaches to tagging morphologically complex languages with fine grained tagsets have considered Czech and Arabic. Khoja (2001) first introduced a tagger for Arabic, which has 131 tags, but subsequent work has collapsed the tagset to simplify tagging (Diab et al., 2004). Like previous Icelandic work (Loftsson, 2007), morphological analyzers disambiguate words before statistical tagging in Arabic (Habash and Rambow, 2005) and Czech (Hajič and Hladká, 1998). This general approach has led to the serial combination of rule based and statistical taggers for efficiency and accuracy (Hajič et al., 2001). While our tagger could be combined with these linguistic resources as well, as in Loftsson (2007), we show state of the art performance without these resources. Another approach to fine-grained tagging captures grammatical structures with tree-based tags, such as "supertags" in the tree-adjoining grammar of Bangalore and Joshi (1999).

3 Icelandic Morphology

Icelandic is notable for its morphological richness. Verbs potentially show as many as 54 different forms depending on tense, mood, voice, person and

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 33–36,
Columbus, Ohio, USA, June 2008.

number. A highly productive class of verbs also show stem vowel alternations reminiscent of Semitic verb morphology (Arabic). Noun morphology exhibits a robust case system; nouns may appear in as many as 16 different forms. The four-case system of Icelandic is similar to that of the Slavic languages (Czech), with case morphology also appearing on elements which agree in case with nouns. However, unlike Czech, case frequently does not convey distinct meaning in Icelandic as it is often determined by elements such as the governing verb in a clause (non-local information). Therefore, while Icelandic case looks formally like Slavic and presents similar challenges for POS tagging, it also may be syntactically-determined, as in Standard Arabic. Icelandic word-order allows a very limited form of scrambling, but does not produce the variety of permutations allowed in Slavic languages. This combination of morphological complexity and syntactic constraint makes Icelandic a good case study for statistical POS tagging techniques.

The morphology necessitates the large extended tagset developed for the Icelandic Frequency Dictionary (Íslensk orðtíðnibók/IFD), a corpus of roughly 590,000 tokens (Pind et al., 1991). We use the 10 IFD data splits produced by Helgadóttir (2004), where the first nine splits are used for evaluation and the tenth for model development. Tags are comprised of up to six elements, such as word class, gender, number, and case, yielding a total of 639 tags, not all of which occur in the training data.

4 Previous Approaches

Helgadóttir (2004) evaluated several data-driven models for Icelandic, including MXPost, a maximum entropy tagger, and TnT, a trigram HMM; both did considerably worse than on English. Icelandic poses significant challenges: data sparseness, non-local tag dependencies, and 136,264 observed trigram sequences make discriminative sequence models, such as CRFs, prohibitively expensive. Given these challenges, the most successful tagger is IceTagger (Loftsson, 2007), a linguistic rule based system with several linguistic resources: a morphological analyzer, a series of local rules and heuristics for handling PPs, verbs, and forcing agreement. Loftsson also improves TnT by integrating a morphological analyzer (TnT*).

Despite these challenges, data driven taggers have several advantages. Learning systems can be easily applied to new corpora, tagsets, or languages and can accommodate integration of other systems (including rule based) or new linguistic resources, such as those used by Loftsson. Therefore, we seek a learning system that can handle these challenges.

5 Bidirectional Sequence Classification

Bidirectional POS tagging (Shen et al., 2007), the current state of the art for English, has some properties that make it appropriate for Icelandic. For example, it can be trained quickly with online learning and does not use tag trigrams, which reduces data sparsity and the cost of learning. It can also allow long range dependencies, which we consider below.

Bidirectional classification uses a perceptron style classifier to assign potential POS tags (hypotheses) to each word using standard POS features and some additional local context features. On each round, the algorithm selects the highest scoring hypothesis and assigns the guessed tag. Unassigned words in the context are reevaluated with this new information. If an incorrect hypothesis is selected during training, the algorithm promotes the score of the correct hypothesis and demotes the selected one. See Shen *et al.* for a detailed explanation.

We begin with a direct application of the bidirectional tagger to Icelandic using a beam of one and the same parameters and features as Shen *et al.* On the development split the tagger achieved an accuracy of 91.61%, which is competitive with the best Icelandic systems. However, test evaluation is not possible due to the prohibitive cost of training the tagger on nine splits; training took almost 4 days on an AMD Opteron 2.8 GHz machine.

Tagset size poses a problem since the tagger must evaluate over 600 options to select the top tag for a word. The tagger rescores the local context after a tag is committed or all untagged words if the classifier is updated. This also highlights a problem with the learning model itself. The tagger uses a one vs. all multi-class strategy, requiring a correct tag to have higher score than every other tag to be selected. While this is plausible for a small number of labels, it overly constrains an Icelandic tagger.

	Accuracy			Train
Tagger	*All*	*Known*	*Unkn.*	*Time*
Bidir	91.61	93.21	69.76	90:27
Bidir+WC	91.98	93.58	70.10	12:20
Bidir+WC+CT	92.36	93.93	70.95	14:02

Table 1: Results on development data. Accuracy is measured by exact match with the gold tag. About 7% of tokens are unknown at test time.

As with most languages, it is relatively simple to assign word class (noun, verb, etc.) and we use this property to divide the tagset into separate learning problems. First, the tagger classifies a word according to one of the eleven word classes. Next, it selects and evaluates all tags consistent with that class. When an incorrect selection is updated, the word class classifier is updated only if it was mistaken as well. The result is a dramatic reduction in the number of tags considered at each step. For some languages, it may make sense to consider further reductions, but not for Icelandic since case, gender, and number decisions are interdependent. Additionally, by learning word class and tag separately, a correct tag need only score higher than other tags of the same word class, not all 639. Furthermore, collapsing tags into word class groups increases training data, allowing the model to generalize features over all tags in a class instead of learning each tag separately (a form of parameter tying).

Training time dropped to 12 hours with the bidirectional word class (WC) tagger and learning performance increased to 91.98% (table 1). Word class accuracy, already quite high at 97.98%, increased to 98.34%, indicating that the tagger can quickly filter out most inappropriate tags. The reduced training cost allowed for test data evaluation, yielding 91.68%, which is a 12.97% relative reduction in error over the best pure data driven model (TnT) and a 1.65% reduction over the best model (IceTagger).

6 Case Tagger

Examining tagger error reveals that most mistakes are caused by case confusion on nouns (84.61% accuracy), adjectives (76.03%), and pronouns (90.67%); these account for 40% of the corpus. While there are 16 case-number-definiteness combinations in the noun morphology, a noun might realize several combinations with a single phonological/orthographic form (case-syncretism). Mistakes in noun case lead to further mistakes for categories which agree with nouns, e.g. adjectives. Assigning appropriate case for nouns is important for a number of other tagging decisions, but often the noun's case provides little or no information about the identity of other tags. It is in this situation that the tagger makes most case-assignment errors. Therefore, while accuracy depends on correct case assignment for these nouns, other tags are mostly unaffected.

One approach to correcting these errors is to introduce long range dependencies, such as those used by IceTagger. While normally hard to add to a learning system, bidirectional learning provides a natural framework since non-local features can be added once a tag has been committed. To allow dependencies on all other tag assignments, and because correcting the remaining case assignments is unlikely to improve other tags, we constructed a separate bidirectional case tagger (CT) that retags case on nouns, adjectives and pronouns. [1] Since gender is important as it relates to case, it is retagged as well. The CT takes a fully tagged sentence from the POS tagger and retags case and gender to nouns, adjectives and pronouns. The CT uses the same features as the POS tagger, but it now has access to all predicted tags. Additionally, we develop several non-local features.

Many case decisions are entirely idiosyncratic, even from the point of view of human language-learners. Some simple transitive verbs in Icelandic arbitrarily require their objects to appear in dative or genitive case, rather than the usual accusative. This arbitrary case-assignment adds no additional meaning, and this set of idiosyncratic verbs is memorized by speakers. A statistical tagger likewise must memorize these verbs based on examples in the training data. To aid generalization, verb-forms were augmented by verb-stems features as described in Dredze and Wallenberg (2008): e.g., the verb forms `dveldi, dvaldi, dvelst, dvelur` all mapped to the stem `dv*l` (*dvelja* "dwell"). The tagger used non-local features, such as the preceding verb's (predicted) tag, gender, case, stem, and nouns within the clause boundary as indicated by

[1] We considered adding case tagging features to and removing case decisions from the tagger; both hurt performance.

Tagger	*All*	*Known*	*Unknown*
MXPost	89.08	91.04	62.50
TnT	90.44	91.82	71.68
TnT*	91.18	92.53	72.75
IceTagger	91.54	92.74	75.09
Bidir+WC	91.68	93.32	69.25
Bidir+WC+CT	**92.06**	93.70	69.74

Table 2: Results on test data.

the tags `cn` (complementizer) or `ct` (relativizer) (Dredze and Wallenberg, 2008).

The CT was used to correct the output of the tagger after training on the corresponding train split. The CT improved results yielding a new best accuracy of 92.06%, a 16.95% and 12.53% reduction over the best data driven and rule systems.

7 Remaining Challenges

We have shown that a data driven approach can achieve state of the art performance on highly inflected languages by extending bidirectional learning to fine grained tagsets and designing a bidirectional non-local case tagger. We conclude with an error analysis to provide future direction.

The tagger is particularly weak on unknown words, a problem caused by case-syncretism and idiosyncratic case-assignment. Data driven taggers can only learn which verbs assign special object cases by observation in the training data. Some verbs and prepositions also assign case based on the meaning of the whole phrase. These are both serious challenges for data-driven methods and could be addressed with the integration of linguistic resources.

However, there is more work to be done on data driven methods. Mistakes in case-assignment due to case syncretism, especially in conjunction with idiosyncratic-case-assigning verbs, account for a large proportion of remaining errors. Verbs that take dative rather than accusative objects are a particular problem, such as mistaking accusative for dative feminine objects (10.6% of occurrences) or dative for accusative feminine objects (11.9%). A possible learning solution lies in combining POS tagging with syntactic parsing, allowing for the identification of clause boundaries, which may help disambiguate noun cases by deducing their grammatical function from that of other clausal constituents.

Additionally, idiosyncratic case-assignment could be learned from *unlabeled* data by finding unambiguous dative objects to identify idiosyncratic verbs. Furthermore, our tagger learns which prepositions idiosyncratically assign a single odd case (e.g. genitive) since prepositions are a smaller class and appear frequently in the corpus. This indicates that further work on data driven methods may still improve the state of the art.

8 Acknowledgments

We thank Hrafn Loftsson for sharing IceTagger and the datasplits, Libin Shen for his tagger, and the Árni Magnússon Institute for Icelandic Studies for access to the corpus.

References

Srinivas Bangalore and Arivand K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In *NAACL/HLT*.

Mark Dredze and Joel Wallenberg. 2008. Further results and analysis of icelandic part of speech tagging. Technical Report MS-CIS-08-13, CIS Dept, University of Pennsylvania.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL*.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: prediction of morphological categories for a rich, structured tagset. In *COLING*.

Jan Hajič, Pavel Krbec, Pavel Květoň, Karel Oliva, and Vladimír Petkevič. 2001. Serial combination of rules and statistics: a case study in czech tagging. In *ACL*.

Sigrun Helgadóttir. 2004. Testing data-driven learning algorithms for pos tagging of icelandic.

Shereen Khoja. 2001. Apt: Arabic part-of-speech tagger. In *NAACL Student Workshop*.

Hrafn Loftsson. 2007. Tagging icelandic text using a linguistic and a statistical tagger. In *NAACL/HLT*.

J Pind, F Magnússon, and S Briem. 1991. The icelandic frequency dictionary. Technical report, The Institute of Lexicography, University of Iceland.

Libin Shen, Giorgio Satta, and Aravind K. Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL*.

Beyond Log-Linear Models: Boosted Minimum Error Rate Training for N-best Re-ranking

Kevin Duh*
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195
kevinduh@u.washington.edu

Katrin Kirchhoff
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195
katrin@ee.washington.edu

Abstract

Current re-ranking algorithms for machine translation rely on log-linear models, which have the potential problem of underfitting the training data. We present **BoostedMERT**, a novel boosting algorithm that uses Minimum Error Rate Training (MERT) as a weak learner and builds a re-ranker far more expressive than log-linear models. BoostedMERT is easy to implement, inherits the efficient optimization properties of MERT, and can quickly boost the BLEU score on N-best re-ranking tasks. In this paper, we describe the general algorithm and present preliminary results on the IWSLT 2007 Arabic-English task.

1 Introduction

N-best list re-ranking is an important component in many complex natural language processing applications (e.g. machine translation, speech recognition, parsing). Re-ranking the N-best lists generated from a 1st-pass decoder can be an effective approach because (a) additional knowledge (features) can be incorporated, and (b) the search space is smaller (i.e. choose 1 out of N hypotheses).

Despite these theoretical advantages, we have often observed little gains in re-ranking machine translation (MT) N-best lists in practice. It has often been observed that N-best list rescoring only yields a moderate improvement over the first-pass output although the potential improvement as measured by the oracle-best hypothesis for each sentence is much higher. This shows that hypothesis features are either not discriminative enough, or that the reranking model is too weak

This performance gap can be mainly attributed to two problems: optimization error and modeling error (see Figure 1).[1] Much work has focused on developing better algorithms to tackle the optimization problem (e.g. MERT (Och, 2003)), since MT evaluation metrics such as BLEU and PER are riddled with local minima and are difficult to differentiate with respect to re-ranker parameters. These optimization algorithms are based on the popular log-linear model, which chooses the English translation e of a foreign sentence f by the rule:

$$\arg\max_e p(e|f) \equiv \arg\max_e \sum_{k=1}^{K} \lambda_k \phi_k(e,f)$$

where $\phi_k(e,f)$ and λ_k are the K features and weights, respectively, and the argmax is over all hypotheses in the N-best list.

We believe that standard algorithms such as MERT already achieve low optimization error (this is based on experience where many random re-starts of MERT give little gains); instead the score gap is mainly due to modeling errors. Standard MT systems use a small set of features (i.e. $K \approx 10$) based on language/translation models.[2] Log-linear models on such few features are simply not expressive enough to achieve the oracle score, regardless of how well the weights $\{\lambda_k\}$ are optimized.

*Work supported by an NSF Graduate Research Fellowship.

[1] Note that we are focusing on closing the gap to the oracle score on the training set (or the development set); if we were focusing on the test set, there would be an additional term, the generalization error.

[2] In this work, we do not consider systems which utilize a large smorgasbord of features, e.g. (Och and others, 2004).

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 37–40,
Columbus, Ohio, USA, June 2008.

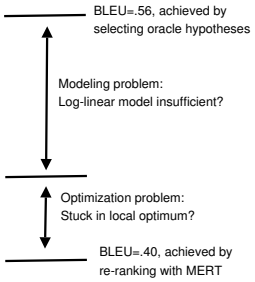


Figure 1: Both modeling and optimization problems increase the (training set) BLEU score gap between MERT re-ranking and oracle hypotheses. We believe that the modeling problem is more serious for log-linear models of around 10 features and focus on it in this work.

To truly achieve the benefits of re-ranking in MT, one must go beyond the log-linear model. The re-ranker should not be a mere dot product operation, but a more dynamic and complex decision maker that exploits the structure of the N-best re-ranking problem.

We present **BoostedMERT**, a general framework for learning such complex re-rankers using standard MERT as a building block. BoostedMERT is easy to implement, inherits MERT's efficient optimization procedure, and more effectively boosts the training score. We describe the algorithm in Section 2, report experiment results in Section 3, and end with related work and future directions (Sections 4, 5).

2 BoostedMERT

The idea for BoostedMERT follows the boosting philosophy of combining several weak classifiers to create a strong overall classifier (Schapire and Singer, 1999). In the classification case, boosting maintains a distribution over each training sample: the distribution is increased for samples that are incorrectly classified and decreased otherwise. In each boosting iteration, a weak learner is trained to optimize on the weighted sample distribution, attempting to correct the mistakes made in the previous iteration. The final classifier is a weighted combination of weak learners. This simple procedure is very effective in reducing training and generalization error.

In BoostedMERT, we maintain a sample distribution $d_i, i = 1 \ldots M$ over the M N-best lists.[3] In each boosting iteration t, MERT is called as as subprocedure to find the best feature weights λ^t on d_i.[4] The sample weight for an N-best list is increased if the currently selected hypothesis is far from the oracle score, and decreased otherwise. Here, the oracle hypothesis for each N-best list is defined as the hypothesis with the best sentence-level BLEU. The final ranker is a combination of (weak) MERT ranker outputs.

Algorithm 1 presents more detailed pseudocode. We use the following notation: Let $\{\mathbf{x_i}\}$ represent the set of M training N-best lists, $i = 1 \ldots M$. Each N-best list $\mathbf{x_i}$ contains N feature vectors (for N hypotheses). Each feature vector is of dimension K, which is the same dimension as the number of feature weights λ obtained by MERT. Let $\{\mathbf{b_i}\}$ be the set of BLEU statistics for each hypothesis in $\{\mathbf{x_i}\}$, which is used to train MERT or to compute BLEU scores for each hypothesis or oracle.

Algorithm 1 BoostedMERT

Input: N-best lists $\{\mathbf{x_i}\}$, BLEU scores $\{\mathbf{b_i}\}$
Input: Initialize sample distribution d_i uniformly
Input: Initialize $y^0 = [0]$, a constant zero vector
Output: Overall Ranker: f^T

for $t = 1$ to T **do**
  Weak ranker: λ^t = MERT($\{\mathbf{x_i}\}$,$\{\mathbf{b_i}\}$,d_i)

  if ($t \geq 2$): $\{y^{t-1}\}$ = PRED($f^{t-1}, \{\mathbf{x_i}\}$)
  $\{y^t\}$ = PRED($\lambda^t, \{\mathbf{x_i}\}$)
  α^t = MERT($[y^{t-1}; y^t]$,$\{\mathbf{b_i}\}$)
  Overall ranker: $f^t = y^{t-1} + \alpha^t y^t$

  for $i = 1$ to M **do**
    a_i = [BLEU of hypothesis selected by f^t] divided by [BLEU of oracle hypothesis]
    $d_i = \exp(-a_i)$/normalizer
  end for
end for

[3] As such, it differs from RankBoost, a boosting-based ranking algorithm in information retrieval (Freund et al., 2003). If applied on MT, RankBoost would maintain a weight for each pair of hypotheses and would optimize a pairwise ranking metric, which is quite dissimilar to BLEU.

[4] This is done by scaling each BLEU statistic, e.g. n-gram precision, reference length, by the appropriate sample weights before computing corpus-level BLEU. Alternatively, one could sample (with replacement) the N-best lists using the distribution and use the resulting stochastic sample as input to an unmodified MERT procedure.

The pseudocode can be divided into 3 sections:

1. Line 2 finds the best log-linear feature weights on distribution d_i. MERT is invoked as a weak learner, so this step is computationally efficient for optimizing MT-specific metrics.

2. Lines 4-7 create an overall ranker by combining the outputs of the previous overall ranker f^{t-1} and current weak ranker λ^t. PRED is a general function that takes a ranker and a M N-best lists and generates a set of M N-dim output vector y representing the predicted reciprocal rank. Specifically, suppose a 3-best list and a ranker predicts ranks (1,3,2) for the 1st, 2nd, and 3rd hypotheses, respectively. Then y = (1/1,1/3,1/2) = (1,0.3,0.5).[5]

   Finally, using a 1-dimensional MERT, the scalar parameter α^t is optimized by maximizing the BLEU of the hypothesis chosen by $y^{t-1}+\alpha^t y^t$. This is analogous to the line search step in boosting for classification (Mason et al., 2000).

3. Lines 9-11 update the sample distribution d_i such that N-best lists with low accuracies a_i are given higher emphasis in the next iteration. The per-list accuracy a_i is defined as the ratio of selected vs. oracle BLEU, but other measures are possible: e.g. ratio of ranks, difference of BLEU.

The final classifier f^T can be seen as a voting procedure among multiple log-linear models generated by MERT. The weighted vote for hypotheses in an N-best list $\mathbf{x_i}$ is represented by the N-dimensional vector: $\hat{y} = \sum_{t=1}^{T} \alpha^t y^t = \sum_{t=1}^{T} \alpha^t$ PRED$(\lambda^t, \mathbf{x_i})$. We choose the hypothesis with the maximum value in $\hat{y}$

Finally, we stress that the above algorithm is an novel extension of boosting to re-ranking problems. There are many open questions and one can not always find a direct analog between boosting for classification and boosting for ranking. For instance, the distribution update scheme of Lines 9-11 is recursive in the classification case (i.e. $d_i = d_i * \exp$(LossOfWeakLearner)), but due to the non-decompositional properties of $\arg\max$ in re-ranking, we have a non-recursive equation based on the overall learner ($d_i = \exp$(LossOfOverallLearner)). This has deep implications on the dynamics of boosting, e.g. the distribution may stay constant in the non-recursive equation, if the new weak ranker gets a small α.

[5]There are other ways to define a ranking output that are worth exploring. For example, a hard argmax definition would be (1,0,0); a probabilistic definition derived from the dot product values can also be used. It is the definition of PRED that introduces non-linearities in BoostedMERT.

3 Experiments

The experiments are done on the IWSLT 2007 Arabic-to-English task (clean text condition). We used a standard phrase-based statistical MT system (Kirchhoff and Yang, 2007) to generated N-best lists (N=2000) on `Development4`, `Development5`, and `Evaluation` sub-sets. `Development4` is used as the Train set; N-best lists that have the same sentence-level BLEU statistics for all hypotheses are filtered since they are not important in impacting training. `Development5` is used as Dev set (in particular, for selecting the number of iterations in boosting), and `Evaluation` (Eval) is the blind dataset for final ranker comparison. Nine features are used in re-ranking.

We compare MERT vs. BoostedMERT. MERT is randomly re-started 30 times, and BoostedMERT is run for 30 iterations, which makes for a relatively fair comparison. MERT usually does not improve its Train BLEU score, even with many random restarts (again, this suggests that optimization error is low). Table 1 shows the results, with BoostedMERT outperforming MERT 42.0 vs. 41.2 BLEU on Eval. BoostedMERT has the potential to achieve 43.7 BLEU, if a better method for selecting optimal iterations can be devised.

It should be noted that the Train scores achieved by both MERT and BoostedMERT is still far from the oracle (around 56). We found empirically that BoostedMERT is somewhat sensitive to the size (M) of the Train set. For small Train sets, BoostedMERT can improve the training score quite drastically; for the current Train set as well as other larger ones, the improvement per iteration is much slower. We plan to investigate this in future work.

	MERT	BOOST	Δ
Train, Best BLEU	40.3	41.0	0.7
Dev, Best BLEU	24.0	25.0	1.0
Eval, Best BLEU	41.2	43.7	2.5
Eval, Selected BLEU	41.2	42.0	0.8

Table 1: The first three rows show the BLEU score for Train, Dev, and Eval from 30 iterations of BoostedMERT or 30 random re-restarts of MERT. The last row shows the actual BLEU on Eval when selecting the number of boosting iterations based on Dev. Last column indicates absolute improvements. BoostedMERT outperforms MERT by 0.8 points on Eval.

4 Related Work

Various methods are used to optimize log-linear models in re-ranking (Shen et al., 2004; Venugopal et al., 2005; Smith and Eisner, 2006). Although this line of work is worthwhile, we believe more gain is possible if we go beyond log-linear models. For example, Shen's method (2004) produces large-margins but observed little gains in performance.

Our BoostedMERT should not be confused with other boosting algorithms such as (Collins and Koo, 2005; Kudo et al., 2005). These algorithms are called boosting because they iteratively choose features (weak learners) and optimize the weights for the boost/exponential loss. They do not, however, maintain a distribution over N-best lists.

The idea of maintaining a distribution over N-best lists is novel. To the best of our knowledge, the most similar algorithm is AdaRank (Xu and Li, 2007), developed for document ranking in information retrieval. Our main difference lies in Lines 4-7 in Algorithm 1: AdaRank proposes a simple closed form solution for α and combines only weak features, not full learners (as in MERT). We have also implemented AdaRank but it gave inferior results.

It should be noted that the theoretical training bounds derived in the AdaRank paper is relevant to BoostedMERT. Similar to standard boosting, this bound shows that the training score can be improved exponentially in the number of iterations. However, we found that the conditions for which this bound is applicable is rarely satisfied in our experiments.[6]

[6]The explanation for this is beyond the scope of this paper; the basic reason is that our weak rankers (MERT) are not weak in practice, so that successive iterations get diminishing returns.

5 Conclusions

We argue that log-linear models often underfit the training data in MT re-ranking, and that this is the reason we observe a large gap between re-ranker and oracle scores. Our solution, BoostedMERT, creates a highly-expressive ranker by voting among multiple MERT rankers.

Although BoostedMERT improves over MERT, more work at both the theoretical and algorithmic levels is needed to demonstrate even larger gains. For example, while standard boosting for classification can exponentially reduce training error in the number of iterations under mild assumptions, these assumptions are frequently not satisfied in the algorithm we described. We intend to further explore the idea of boosting on N-best lists, drawing inspirations from the large body of work on boosting for classification whenever possible.

References

M. Collins and T. Koo. 2005. Discriminative reranking for natural langauge parsing. *Computational Linguistics*, 31(1).

Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4.

K. Kirchhoff and M. Yang. 2007. The UW machine translation system for IWSLT 2007. In *IWSLT*.

T. Kudo, J. Suzuki, and H. Isozaki. 2005. Boosting-based parse reranking with subtree features. In *ACL*.

L. Mason, J. Baxter, P. Bartless, and M. Frean. 2000. Boosting as gradient descent. In *NIPS*.

F.J. Och et al. 2004. A smorgasbord of features for statistical machine translation. In *HLT/NAACL*.

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

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3).

L. Shen, A. Sarkar, and F.J. Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*.

D. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. of COLING/ACL Companion Volume*.

A. Venugopal, A. Zollmann, and A. Waibel. 2005. Training and evaluating error minimization rules for SMT. In *ACL Workshop on Building/Using Parallel Texts*.

J. Xu and H. Li. 2007. AdaRank: A boosting algorithm for information retrieval. In *SIGIR*.

Coreference-inspired Coherence Modeling

Micha Elsner and Eugene Charniak
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{melsner,ec}@cs.brown.edu

Abstract

Research on coreference resolution and summarization has modeled the way entities are realized as concrete phrases in discourse. In particular there exist models of the noun phrase syntax used for discourse-new versus discourse-old referents, and models describing the likely distance between a pronoun and its antecedent. However, models of discourse coherence, as applied to information ordering tasks, have ignored these kinds of information. We apply a discourse-new classifier and pronoun coreference algorithm to the information ordering task, and show significant improvements in performance over the entity grid, a popular model of local coherence.

1 Introduction

Models of discourse coherence describe the relationships between nearby sentences, in which previous sentences help make their successors easier to understand. Models of coherence have been used to impose an order on sentences for multidocument summarization (Barzilay et al., 2002), to evaluate the quality of human-authored essays (Miltsakaki and Kukich, 2004), and to insert new information into existing documents (Chen et al., 2007).

These models typically view a sentence either as a bag of words (Foltz et al., 1998) or as a bag of entities associated with various syntactic roles (Lapata and Barzilay, 2005). However, a mention of an entity contains more information than just its head and syntactic role. The referring expression itself contains discourse-motivated information distinguishing familiar entities from unfamiliar and salient from non-salient. These patterns have been studied extensively, by linguists (Prince, 1981; Fraurud, 1990) and in the field of coreference resolution. We draw on the coreference work, taking two standard models from the literature and applying them to coherence modeling.

Our first model distinguishes discourse-new from discourse-old noun phrases, using features based on Uryupina (2003). Discourse-new NPs are those whose referents have not been previously mentioned in the discourse. As noted by studies since Hawkins (1978), there are marked syntactic differences between the two classes.

Our second model describes pronoun coreference. To be intelligible, pronouns must be placed close to appropriate referents with the correct number and gender. Centering theory (Grosz et al., 1995) describes additional constraints about which entities in a discourse can be pronominalized: if there are pronouns in a segment, they must include the backward-looking center. We use a model which probabilistically attempts to describe these preferences (Ge et al., 1998).

These two models can be combined with the entity grid described by Lapata and Barzilay (2005) for significant improvement. The magnitude of the improvement is particularly interesting given that Barzilay and Lapata (2005) do use a coreference system but are unable to derive much advantage from it.

2 Discourse-new Model

In the task of discourse-new classification, the model is given a referring expression (as in previous work, we consider only NPs) from a document and must

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 41–44,
Columbus, Ohio, USA, June 2008.

determine whether it is a first mention (*discourse-new*) or a subsequent mention (*discourse-old*). Features such as full names, appositives, and restrictive relative clauses are associated with the introduction of unfamiliar entities into discourse (Hawkins, 1978; Fraurud, 1990; Vieira and Poesio, 2000). Classifiers in the literature include (Poesio et al., 2005; Uryupina, 2003; Ng and Cardie, 2002). The system of Nenkova and McKeown (2003) works in the opposite direction. It is designed to rewrite the references in multi-document summaries, so that they conform to the common discourse patterns.

We construct a maximum-entropy classifier using syntactic and lexical features derived from Uryupina (2003), and a publicly available learning tool (Daumé III, 2004). Our system scores 87.4% (F-score of the *disc-new* class on the MUC-7 formal test set); this is comparable to the state-of-the-art system of Uryupina (2003), which scores 86.9[1].

To model coreference with this system, we assign each NP in a document a label $L_{np} \in \{new, old\}$. Since the correct labeling depends on the coreference relationships between the NPs, we need some way to guess at this; we take all NPs with the same head to be coreferent, as in the non-coreference version of (Barzilay and Lapata, 2005)[2]. We then take the probability of a document as $\prod_{np:NPs} P(L_{np}|np)$.

We must make several small changes to the model to adapt it to this setting. For the discourse-new classification task, the model's most important feature is whether the head word of the NP to be classified has occurred previously (as in Ng and Cardie (2002) and Vieira and Poesio (2000)). For coherence modeling, we must remove this feature, since it depends on document order, which is precisely what we are trying to predict. The coreference heuristic will also fail to resolve any pronouns, so we discard them.

Another issue is that NPs whose referents are familiar tend to resemble discourse-old NPs, even though they have not been previously mentioned (Fraurud, 1990). These include unique objects like *the FBI* or generic ones like *danger* or *percent*. To avoid using these deceptive phrases as examples of discourse-newness, we attempt to heuristically remove them from the training set by discarding any NP whose head occurs only once in the document[3].

The labels we apply to NPs in our test data are systematically biased by the "same head" heuristic we use for coreference. This is a disadvantage for our system, but it has a corresponding advantage– we can use training data labeled using the same heuristic, without any loss in performance on the coherence task. NPs we fail to learn about during training are likely to be mislabeled at test time anyway, so performance does not degrade by much. To counter this slight degradation, we can use a much larger training corpus, since we no longer require gold-standard coreference annotations.

3 Pronoun Coreference Model

Pronoun coreference is another important aspect of coherence– if a pronoun is used too far away from any natural referent, it becomes hard to interpret, creating confusion. Too many referents, however, create ambiguity. To describe this type of restriction, we must model the probability of the text containing pronouns (denoted r_i), jointly with their referents a_i. (This takes more work than simply resolving the pronouns conditioned on the text.) The model of Ge et al. (1998) provides the requisite probabilities:

$$P(a_i, r_i|a_i^{i-1}) = P(a_i|h(a_i), m(a_i)) \\ P_{gen}(a_i, r_i)P_{num}(a_i, r_i)$$

Here $h(a)$ is the Hobbs distance (Hobbs, 1976), which measures distance between a pronoun and prospective antecedent, taking into account various factors, such as syntactic constraints on pronouns. $m(a)$ is the number of times the antecedent has been mentioned previously in the document (again using "same head" coreference for full NPs, but also counting the previous antecedents a_i^{i-1}). P_{gen} and P_{num} are distributions over gender and number given words. The model is trained using a small hand-annotated corpus first used in Ge et al. (1998).

[1] Poesio et al. (2005) score 90.2%, but on a different corpus.

[2] Unfortunately, this represents a substantial sacrifice; as Poesio and Vieira (1998) show, only about 2/3 of definite descriptions which are anaphoric have the same head as their antecedent.

[3] Bean and Riloff (1999) and Uryupina (2003) construct quite accurate classifiers to detect unique NPs. However, some preliminary experiments convinced us that our heuristic method worked well enough for the purpose.

	Disc. Acc	Disc. F	Ins.
Random	50.00	50.00	12.58
Entity Grid	76.17	77.55	19.57
Disc-New	70.35	73.47	16.27
Pronoun	55.77	62.27	13.95
EGrid+Disc-New	78.88	80.31	21.93
Combined	79.60	81.02	22.98

Table 1: Results on 1004 WSJ documents.

Finding the probability of a document using this model requires us to sum out the antecedents a. Unfortunately, because each a_i is conditioned on the previous ones, this cannot be done efficiently. Instead, we use a greedy search, assigning each pronoun left to right. Finally we report the probability of the resulting sequence of pronoun assignments.

4 Baseline Model

As a baseline, we adopt the entity grid (Lapata and Barzilay, 2005). This model outperforms a variety of word overlap and semantic similarity models, and is used as a component in the state-of-the-art system of Soricut and Marcu (2006). The entity grid represents each entity by tracking the syntactic roles in which it appears throughout the document. The internal syntax of the various referring expressions is ignored. Since it also uses the "same head" coreference heuristic, it also disregards pronouns.

Since the three models use very different feature sets, we combine them by assuming independence and multiplying the probabilities.

5 Experiments

We evaluate our models using two tasks, both based on the assumption that a human-authored document is coherent, and uses the best possible ordering of its sentences (see Lapata (2006)). In the discrimination task (Barzilay and Lapata, 2005), a document is compared with a random permutation of its sentences, and we score the system correct if it indicates the original as more coherent[4].

Discrimination becomes easier for longer documents, since a random permutation is likely to be much less similar to the original. Therefore we also test our systems on the task of insertion (Chen et al., 2007), in which we remove a sentence from a document, then find the point of insertion which yields the highest coherence score. The reported score is the average fraction of sentences per document reinserted in their original position (averaged over documents, not sentences, so that longer documents do not disproportionally influence the results)[5].

We test on sections 14-24 of the Penn Treebank (1004 documents total). Previous work has focused on the AIRPLANE corpus (Barzilay and Lee, 2004), which contains short announcements of airplane crashes written by and for domain experts. These texts use a very constrained style, with few discourse-new markers or pronouns, and so our system is ineffective; the WSJ corpus is much more typical of normal informative writing. Also unlike previous work, we do not test the task of completely reconstructing a document's order, since this is computationally intractable and results on WSJ documents[6] would likely be dominated by search errors.

Our results are shown in table 5. When run alone, the entity grid outperforms either of our models. However, all three models are significantly better than random. Combining all three models raises discrimination performance by 3.5% over the baseline and insertion by 3.4%. Even the weakest component, pronouns, contributes to the joint model; when it is left out, the resulting *EGrid + Disc-New* model is significantly worse than the full combination. We test significance using Wilcoxon's signed-rank test; all results are significant with $p < .001$.

6 Conclusions

The use of these coreference-inspired models leads to significant improvements in the baseline. Of the two, the discourse-new detector is by far more effective. The pronoun model's main problem is that, although a pronoun may have been displaced from its original position, it can often find another seemingly acceptable referent nearby. Despite this issue

[4]Since the model might refuse to make a decision by scoring a permutation the same as the original, we also report F-score, where precision is $correct/decisions$ and recall is $correct/total$.

[5]Although we designed a metric that distinguishes near misses from random performance, it is very well correlated with exact precision, so, for simplicity's sake, we omit it.

[6]Average 22 sentences, as opposed to 11.5 for AIRPLANE.

it performs significantly better than chance and is capable of slightly improving the combined model. Both of these models are very different from the lexical and entity-based models currently used for this task (Soricut and Marcu, 2006), and are probably capable of improving the state of the art.

As mentioned, Barzilay and Lapata (2005) uses a coreference system to attempt to improve the entity grid, but with mixed results. Their method of combination is quite different from ours; they use the system's judgements to define the "entities" whose repetitions the system measures[7]. In contrast, we do not attempt to use any proposed coreference links; as Barzilay and Lapata (2005) point out, these links are often erroneous because the disorded input text is so dissimilar to the training data. Instead we exploit our models' ability to measure the probability of various aspects of the text.

Acknowledgements

Chen and Barzilay, reviewers, DARPA, et al.

References

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: an entity-based approach. In *ACL 2005*.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004*, pages 113–120.

Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Results (JAIR)*, 17:35–55.

David L. Bean and Ellen Riloff. 1999. Corpus-based identification of non-anaphoric noun phrases. In *ACL'99*, pages 373–380.

Erdong Chen, Benjamin Snyder, and Regina Barzilay. 2007. Incremental text structuring with online hierarchical ranking. In *Proceedings of EMNLP*.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regressio n. Paper available at http://pub.hal3.name#daume04cg-bfgs, implementation available at http://hal3.name/megam/, August.

Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–307.

Kari Fraurud. 1990. Definiteness and the processing of noun phrases in natural discourse. *Journal of Semantics*, 7(4):395–433.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–171.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

John A. Hawkins. 1978. *Definiteness and indefiniteness: a study in reference and grammaticality prediction.* Croom Helm Ltd.

Jerry R. Hobbs. 1976. Pronoun resolution. Technical Report 76-1, City College New York.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*, pages 1085–1090.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):1–14.

E. Miltsakaki and K. Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Nat. Lang. Eng.*, 10(1):25–55.

Ani Nenkova and Kathleen McKeown. 2003. References to named entities: a corpus study. In *NAACL '03*, pages 70–72.

Vincent Ng and Claire Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *COLING*.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Massimo Poesio, Mijail Alexandrov-Kabadjov, Renata Vieira, Rodrigo Goulart, and Olga Uryupina. 2005. Does discourse-new detection help definite description resolution? In *Proceedings of the Sixth International Workshop on Computational Semantics*, Tillburg.

Ellen Prince. 1981. Toward a taxonomy of given-new information. In Peter Cole, editor, *Radical Pragmatics*, pages 223–255. Academic Press, New York.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *ACL-2006*.

Olga Uryupina. 2003. High-precision identification of discourse new and unique noun phrases. In *Proceedings of the ACL Student Workshop*, Sapporo.

Renata Vieira and Massimo Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

[7] We attempted this method for pronouns using our model, but found it ineffective.

Enforcing Transitivity in Coreference Resolution

Jenny Rose Finkel and Christopher D. Manning
Department of Computer Science
Stanford University
Stanford, CA 94305
{jrfinkel|manning}@cs.stanford.edu

Abstract

A desirable quality of a coreference resolution system is the ability to handle transitivity constraints, such that even if it places high likelihood on a particular mention being coreferent with each of two other mentions, it will also consider the likelihood of those two mentions being coreferent when making a final assignment. This is exactly the kind of constraint that integer linear programming (ILP) is ideal for, but, surprisingly, previous work applying ILP to coreference resolution has not encoded this type of constraint. We train a coreference classifier over pairs of mentions, and show how to encode this type of constraint on top of the probabilities output from our pairwise classifier to extract the most probable legal entity assignments. We present results on two commonly used datasets which show that enforcement of transitive closure consistently improves performance, including improvements of up to 3.6% using the b^3 scorer, and up to 16.5% using cluster f-measure.

1 Introduction

Much recent work on coreference resolution, which is the task of deciding which noun phrases, or *mentions*, in a document refer to the same real world entity, builds on Soon et al. (2001). They built a decision tree classifier to label pairs of mentions as coreferent or not. Using their classifier, they would build up *coreference chains*, where each mention was linked up with the most recent previous mention that the classifier labeled as coreferent, if such a mention existed. Transitive closure in this model was done implicitly. If *John Smith* was labeled coreferent with *Smith*, and *Smith* with *Jane Smith*, then *John Smith* and *Jane Smith* were also coreferent regardless of the classifier's evaluation of that pair. Much work that followed improved upon this strategy, by improving the features (Ng and Cardie, 2002b), the type of classifier (Denis and Baldridge, 2007), and changing mention links to be to the most likely antecedent rather than the most recent positively labeled antecedent (Ng and Cardie, 2002b). This line of work has largely ignored the implicit transitivity of the decisions made, and can result in unintuitive chains such as the *Smith* chain just described, where each pairwise decision is sensible, but the final result is not.

Ng and Cardie (2002a) and Ng (2004) highlight the problem of determining whether or not common noun phrases are anaphoric. They use two classifiers, an anaphoricity classifier, which decides if a mention should have an antecedent and a pairwise classifier similar those just discussed, which are combined in a cascaded manner. More recently, Denis and Baldridge (2007) utilized an integer linear programming (ILP) solver to better combine the decisions made by these two complementary classifiers, by finding the globally optimal solution according to both classifiers. However, when encoding constraints into their ILP solver, they did not enforce transitivity.

The goal of the present work is simply to show that transitivity constraints are a useful source of information, which can and should be incorporated into an ILP-based coreference system. For this goal, we put aside the anaphoricity classifier and focus on the pairwise classifier and transitivity constraints. We build a pairwise logistic classifier, trained on all pairs of mentions, and then at test time we use an ILP solver equipped with transitivity constraints to find the most likely legal assignment to the variables which represent the pairwise decisions.[1] Our results show a significant improvement compared to the naïve use of the pairwise classifier.

Other work on global models of coreference (as

[1] A *legal assignment* is one which respects transitive closure.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 45–48,
Columbus, Ohio, USA, June 2008.

opposed to pairwise models) has included: Luo et al. (2004) who used a Bell tree whose leaves represent possible partitionings of the mentions into entities and then trained a model for searching the tree; McCallum and Wellner (2004) who defined several conditional random field-based models; Ng (2005) who took a reranking approach; and Culotta et al. (2006) who use a probabilistic first-order logic model.

2 Coreference Resolution

For this task we are given a document which is annotated with a set of mentions, and the goal is to cluster the mentions which refer to the same entity. When describing our model, we build upon the notation used by Denis and Baldridge (2007).

2.1 Pairwise Classification

Our baseline systems are based on a logistic classifier over pairs of mentions. The probability of a pair of mentions takes the standard logistic form:

$$P(x_{\langle i,j\rangle}|m_i, m_j; \theta) = \left(1 + e^{-f(m_i,m_j)\cdot\theta}\right)^{-1} \quad (1)$$

where m_i and m_j correspond to mentions i and j respectively; $f(m_i, m_j)$ is a feature function over a pair of mentions; θ are the feature weights we wish to learn; and $x_{\langle i,j\rangle}$ is a boolean variable which takes value 1 if m_i and m_j are coreferent, and 0 if they are not. The log likelihood of a document is the sum of the log likelihoods of all pairs of mentions:

$$\mathcal{L}(\mathbf{x}|\mathbf{m};\theta) = \sum_{m_i,m_j\in\mathbf{m}^2} \log P(x_{\langle i,j\rangle}|m_i, m_j; \theta) \quad (2)$$

where $\mathbf{m}$ is the set of mentions in the document, and $\mathbf{x}$ is the set of variables representing each pairwise coreference decision $x_{\langle i,j\rangle}$. Note that this model is degenerate, because it assigns probability mass to nonsensical clusterings. Specifically, it will allow $x_{\langle i,j\rangle} = x_{\langle j,k\rangle} = 1$ while $x_{\langle i,k\rangle} = 0$.

Prior work (Soon et al., 2001; Denis and Baldridge, 2007) has generated training data for pairwise classifiers in the following manner. For each mention, work backwards through the preceding mentions in the document until you come to a true coreferent mention. Create negative examples for all intermediate mentions, and a positive example for the mention and its correct antecedent. This approach made sense for Soon et al. (2001) because testing proceeded in a similar manner: for each mention, work backwards until you find a previous mention which the classifier thinks is coreferent, add a link, and terminate the search. The COREF-ILP model of Denis and Baldridge (2007) took a different approach at test time: for each mention they would work backwards and add a link for *all* previous mentions which the classifier deemed coreferent. This is equivalent to finding the most likely assignment to each $x_{\langle i,j\rangle}$ in Equation 2. As noted, these assignments may not be a legal clustering because there is no guarantee of transitivity. The transitive closure happens in an ad-hoc manner after this assignment is found: any two mentions linked through other mentions are determined to be coreferent. Our SOON-STYLE baseline used the same training and testing regimen as Soon et al. (2001). Our D&B-STYLE baseline used the same test time method as Denis and Baldridge (2007), however at training time we created data for all mention pairs.

2.2 Integer Linear Programming to Enforce Transitivity

Because of the ad-hoc manner in which transitivity is enforced in our baseline systems, we do not necessarily find the most probable legal clustering. This is exactly the kind of task at which integer linear programming excels. We need to first formulate the objective function which we wish the ILP solver to maximize at test time.[2] Let $p_{\langle i,j\rangle} = \log P(x_{\langle i,j\rangle}|m_i, m_j; \theta)$, which is the log probability that m_i and m_j are coreferent according to the pairwise logistic classifier discussed in the previous section, and let $\bar{p}_{\langle i,j\rangle} = \log(1 - p_{\langle i,j\rangle})$, be the log probability that they are not coreferent. Our objective function is then the log probability of a particular (possibly illegal) variable assignment:

$$\max \sum_{m_i,m_j\in\mathbf{m}^2} p_{\langle i,j\rangle} \cdot x_{\langle i,j\rangle} - \bar{p}_{\langle i,j\rangle} \cdot (1 - x_{\langle i,j\rangle}) \quad (3)$$

We add binary constraints on each of the variables: $x_{\langle i,j\rangle} \in \{0, 1\}$. We also add constraints, over each triple of mentions, to enforce transitivity:

$$(1 - x_{\langle i,j\rangle}) + (1 - x_{\langle j,k\rangle}) \geq (1 - x_{\langle i,k\rangle}) \quad (4)$$

[2] Note that there are no changes from the D&B-STYLE baseline system at training time.

This constraint ensures that whenever $x_{\langle i,j \rangle} = x_{\langle j,k \rangle} = 1$ it must also be the case that $x_{\langle i,k \rangle} = 1$.

3 Experiments

We used *lp_solve*[3] to solve our ILP optimization problems. We ran experiments on two datasets. We used the MUC-6 formal training and test data, as well as the NWIRE and BNEWS portions of the ACE (Phase 2) corpus. This corpus had a third portion, NPAPER, but we found that several documents where too long for *lp_solve* to find a solution.[4]

We added named entity (NE) tags to the data using the tagger of Finkel et al. (2005). The ACE data is already annotated with NE tags, so when they conflicted they overrode the tags output by the tagger. We also added part of speech (POS) tags to the data using the tagger of Toutanova et al. (2003), and used the tags to decide if mentions were plural or singular. The ACE data is labeled with mention type (*pronominal*, *nominal*, and *name*), but the MUC-6 data is not, so the POS and NE tags were used to infer this information. Our feature set was simple, and included many features from (Soon et al., 2001), including the pronoun, string match, definite and demonstrative NP, number and gender agreement, proper name and appositive features. We had additional features for NE tags, head matching and head substring matching.

3.1 Evaluation Metrics

The MUC scorer (Vilain et al., 1995) is a popular coreference evaluation metric, but we found it to be fatally flawed. As observed by Luo et al. (2004), if all mentions in each document are placed into a single entity, the results on the MUC-6 formal test set are 100% recall, 78.9% precision, and 88.2% F1 score – significantly higher than any published system. The b^3 scorer (Amit and Baldwin, 1998) was proposed to overcome several shortcomings of the MUC scorer. However, coreference resolution is a clustering task, and many cluster scorers already exist. In addition to the MUC and b^3 scorers, we also evaluate using cluster f-measure (Ghosh, 2003), which is the standard f-measure computed over true/false coreference decisions for pairs of mentions; the Rand index (Rand, 1971), which is pairwise accuracy of the clustering; and variation of information (Meila, 2003), which utilizes the entropy of the clusterings and their mutual information (and for which lower values are better).

3.2 Results

Our results are summarized in Table 1. We show performance for both baseline classifiers, as well as our ILP-based classifier, which finds the most probable legal assignment to the variables representing coreference decisions over pairs of mentions. For comparison, we also give the results of the COREF-ILP system of Denis and Baldridge (2007), which was also based on a naïve pairwise classifier. They used an ILP solver to find an assignment for the variables, but as they note at the end of Section 5.1, it is equivalent to taking all links for which the classifier returns a probability ≥ 0.5, and so the ILP solver is not really necessary. We also include their JOINT-ILP numbers, however that system makes use of an additional anaphoricity classifier.

For all three corpora, the ILP model beat both baselines for the cluster f-score, Rand index, and variation of information metrics. Using the b^3 metric, the ILP system and the D&B-STYLE baseline performed about the same on the MUC-6 corpus, though for both ACE corpora, the ILP system was the clear winner. When using the MUC scorer, the ILP system always did worse than the D&B-STYLE baseline. However, this is precisely because the transitivity constraints tend to yield smaller clusters (which increase precision while decreasing recall). Remember that going in the opposite direction and simply putting *all* mentions in one cluster produces a MUC score which is higher than any in the table, even though this clustering is clearly not useful in applications. Hence, we are skeptical of this measure's utility and provide it primarily for comparison with previous work. The improvements from the ILP system are most clearly shown on the ACE NWIRE corpus, where the b^3 f-score improved 3.6%, and the cluster f-score improved 16.5%.

4 Conclusion

We showed how to use integer linear programming to encode transitivity constraints in a corefer-

[3] From `http://lpsolve.sourceforge.net/`

[4] Integer linear programming is, after all, NP-hard.

	MUC Scorer			b^3 Scorer			Cluster				
Model	P	R	F1	P	R	F1	P	R	F1	RAND	VOI
MUC-6											
D&B-style baseline	84.8	59.4	69.9	79.7	54.4	64.6	43.8	44.4	44.1	89.9	1.78
Soon-style baseline	91.5	51.5	65.9	94.4	46.7	62.5	88.2	31.9	46.9	93.5	1.65
ILP	89.7	55.1	68.3	90.9	49.7	64.3	74.1	37.1	49.5	93.2	1.65
ACE − NWIRE											
D&B coref-ILP	74.8	60.1	66.8		–			–		–	–
D&B joint-ILP	75.8	60.8	67.5		–			–		–	–
D&B-style baseline	73.3	67.6	70.4	70.1	71.4	70.8	31.1	54.0	39.4	91.7	1.42
Soon-style baseline	85.3	37.8	52.4	94.1	56.9	70.9	67.7	19.8	30.6	95.5	1.38
ILP	78.7	58.5	67.1	86.8	65.2	74.5	76.1	44.2	55.9	96.5	1.09
ACE − BNEWS											
D&B coref-ILP	75.5	62.2	68.2		–			–		–	–
D&B joint-ILP	78.0	62.1	69.2		–			–		–	–
D&B-style baseline	77.9	51.1	61.7	80.3	64.2	71.4	35.5	33.8	34.6	0.89	1.32
Soon-style baseline	90.0	43.2	58.3	95.6	58.4	72.5	83.3	21.5	34.1	0.93	1.09
ILP	87.8	46.8	61.1	93.5	59.9	73.1	77.5	26.1	39.1	0.93	1.06

Table 1: Results on all three datasets with all five scoring metrics. For VOI a lower number is better.

ence classifier which models pairwise decisions over mentions. We also demonstrated that enforcing such constraints at test time can significantly improve performance, using a variety of evaluation metrics.

Acknowledgments

Thanks to the following members of the Stanford NLP reading group for helpful discussion: Sharon Goldwater, Michel Galley, Anna Rafferty.

This paper is based on work funded by the Disruptive Technology Office (DTO) Phase III Program for Advanced Question Answering for Intelligence (AQUAINT).

References

B. Amit and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *MUC7*.

A. Culotta, M. Wick, and A. McCallum. 2006. First-order probabilistic models for coreference resolution. In *NAACL*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT-NAACL*, Rochester, New York.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*.

J. Ghosh. 2003. Scalable clustering methods for data mining. In N. Ye, editor, *Handbook of Data Mining*, chapter 10, pages 247–277. Lawrence Erlbaum Assoc.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *ACL*.

A. McCallum and B. Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*.

M. Meila. 2003. Comparing clusterings by the variation of information. In *COLT*.

V. Ng and C. Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *COLING*.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *ACL*.

V. Ng. 2004. Learning noun phrase anaphoricity to improve coreference resolution: issues in representation and optimization. In *ACL*.

V. Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *ACL*.

W. M. Rand. 1971. Objective criteria for the evaluation of clustering methods. In *Journal of the American Statistical Association, 66*, pages 846–850.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. In *Computational Linguistics, 27(4)*.

K. Toutanova, D. Klein, and C. Manning. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL 2003*.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC6*.

Simulating the Behaviour of Older versus Younger Users when Interacting with Spoken Dialogue Systems

Kallirroi Georgila, Maria Wolters and Johanna D. Moore
Human Communication Research Centre
University of Edinburgh
kgeorgil|mwolters|jmoore@inf.ed.ac.uk

Abstract

In this paper we build user simulations of older and younger adults using a corpus of interactions with a Wizard-of-Oz appointment scheduling system. We measure the quality of these models with standard metrics proposed in the literature. Our results agree with predictions based on statistical analysis of the corpus and previous findings about the diversity of older people's behaviour. Furthermore, our results show that these metrics can be a good predictor of the behaviour of different types of users, which provides evidence for the validity of current user simulation evaluation metrics.

1 Introduction

Using machine learning to induce dialogue management policies requires large amounts of training data, and thus it is typically not feasible to build such models solely with data from real users. Instead, data from real users is used to build simulated users (SUs), who then interact with the system as often as needed. In order to learn good policies, the behaviour of the SUs needs to cover the range of variation seen in real users (Schatzmann et al., 2005; Georgila et al., 2006). Furthermore, SUs are critical for evaluating candidate dialogue policies.

To date, several techniques for building SUs have been investigated and metrics for evaluating their quality have been proposed (Schatzmann et al., 2005; Georgila et al., 2006). However, to our knowledge, no one has tried to build user simulations for different populations of real users and measure whether results from evaluating the quality of those simulations agree with what is known about those particular types of real users, extracted from other studies of those populations. This is presumably due to the lack of corpora for different types of users.

In this paper we focus on the behaviour of older vs. younger adults. Most of the work to date on dialogue systems focuses on young users. However, as average life expectancy increases, it becomes increasingly important to design dialogue systems in such a way that they can accommodate older people's behaviour. Older people are a user group with distinct needs and abilities (Czaja and Lee, 2007) that present challenges for user modelling. To our knowledge no one so far has built statistical user simulation models for older people. The only statistical spoken dialogue system for older people we are aware of is Nursebot, an early application of statistical methods (POMDPs) within the context of a medication reminder system (Roy et al., 2000).

In this study, we build SUs for both younger and older adults using n-grams. Our data comes from a fully annotated corpus of 447 interactions of older and younger users with a Wizard-of-Oz (WoZ) appointment scheduling system (Georgila et al., 2008). We then evaluate these models using standard metrics (Schatzmann et al., 2005; Georgila et al., 2006) and compare our findings with the results of statistical corpus analysis.

The novelty of our work lies in two areas. First, to the best of our knowledge this is the first time that statistical SUs have been built for the increasingly important population of older users.

Secondly, a general (but as yet untested) assumption in this field is that current SUs are "enough like" real users for training good policies, and that testing system performance in simulated dialogues is an accurate indication of how a system will perform with human users. The validity of these assumptions is

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 49–52,
Columbus, Ohio, USA, June 2008.

a critically important open research question. Currently one of the standard methods for evaluating the quality of a SU is to run a user simulation on a real corpus and measure how often the action generated by the SU agrees with the action observed in the corpus (Schatzmann et al., 2005; Georgila et al., 2006). This method can certainly give us some insight into how strongly a SU resembles a real user, but the validity of the metrics used remains an open research problem. In this paper, we take this a step further. We measure the quality of user simulation models for both older and younger users, and show that these metrics are a good predictor of the behaviour of those two user types.

The structure of the paper is as follows: In section 2 we describe our data set. In section 3 we discuss the differences between older and younger users as measured in our corpus using standard statistical techniques. Then in section 4 we present our user simulations. Finally in section 5 we present our conclusions and propose future work.

2 The Corpus

The dialogue corpus which our simulations are based on was collected during a controlled experiment where we systematically varied: (1) the number of options that users were presented with (one option, two options, four options); (2) the confirmation strategy employed (explicit confirmation, implicit confirmation, no confirmation). The combination of these 3×3 design choices yielded 9 different dialogue systems.

Participants were asked to schedule a health care appointment with each of the 9 systems, yielding a total of 9 dialogues per participant. System utterances were generated using a simple template-based algorithm and synthesised using the speech synthesis system Cerevoice (Aylett et al., 2006), which has been shown to be intelligible to older users (Wolters et al., 2007). The human wizard took over the function of the speech recognition, language understanding, and dialogue management components.

Each dialogue corresponded to a fixed schema: First, users arranged to see a specific health care professional, then they arranged a specific half-day, and finally, a specific half-hour time slot on that half-day was agreed. In a final step, the wizard confirmed the appointment.

The full corpus consists of 447 dialogues; 3 dialogues were not recorded. A total of 50 participants were recruited, of which 26 were older (50–85) and 24 were younger (20–30). The older users contributed 232 dialogues, the younger ones 215. Older and younger users were matched for level of education and gender.

All dialogues were transcribed orthographically and annotated with dialogue acts and dialogue context information. Using a unique mapping, we associate each dialogue act with a $\langle$speech act, task$\rangle$ pair, where the speech act is task independent and the task corresponds to the slot in focus (health professional, half-day or time slot). For each dialogue, five measures of dialogue quality were recorded: objective task completion, perceived task completion, appointment recall, length (in turns), and detailed user satisfaction ratings. A detailed description of the corpus design, statistics, and annotation scheme is provided in (Georgila et al., 2008).

Our analysis of the corpus shows that there are clear differences in the way users interact with the systems. Since it is these differences that good user simulations need to capture, the most relevant findings for the present study are summarised in the next section.

3 Older vs. Younger Users

Since the user simulations (see section 4) are based mainly on dialogue act annotations, we will use speech act statistics to illustrate some key differences in behaviour between older and younger users. User speech acts were grouped into four categories that are relevant to dialogue management: speech acts that result in grounding (`ground`), speech acts that result in confirmations (`confirm`) (note, this category overlaps with `ground` and occurs after the system has explicitly or implicitly attempted to confirm the user's response), speech acts that indicate user initiative (`init`), and speech acts that indicate social interaction with the system (`social`). We also computed the average number of different speech act types used, the average number of speech act tokens, and the average token/type ratio per user. Results are given in Table 1.

There are 28 distinct user speech acts (Georgila et al., 2008). Older users not only produce more individual speech acts, they also use a far richer variety of speech acts, on average 14 out of 28 as opposed to 9 out of 28. The token/type ratio remains the same, however. Although the absolute frequency of confirmation and grounding speech acts is approximately

Variable	Older	Younger	Sig.
# speech act types	14	9	***
# speech act tokens	126	73	***
Sp. act tokens/types	8.7	8.5	n.s.
# Confirm	31	30	n.s.
% Confirm	28.3	41.5	***
# Ground	33	30	n.s.
% Ground	29.4	41.7	***
# Social	26	5	***
% Social	17.9	5.3	***
# Init	15	3	***
% Init	9.0	3.4	**

Table 1: Behaviour of older vs. younger users. Numbers are summed over all dialogues and divided by the number of users. *: p<0.01, **: p<0.005, ***: p<0.001 or better.

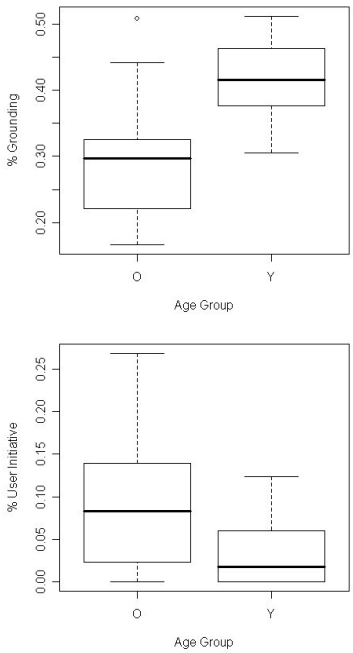


Figure 1: Relative frequency of (a) grounding and (b) user initiative.

the same for younger and older users, the relative frequency of these types of speech acts is far lower for older than for younger users, because older users are far more likely to take initiative by providing additional information to the system and speech acts indicating social interaction. Based on this analysis alone, we would predict that user simulations trained on younger users only will not fare well when tested on older users, because the behaviour of older users is richer and more complex.

But do older and younger users constitute two separate groups, or are there older users that behave like younger ones? In the first case, we cannot use data from older people to create simulations of younger users' behaviour. In the second case, data from older users might be sufficient to approximately cover the full range of behaviour we see in the data. The boxplots given in Fig. 1 indicate that the latter is in fact true. Even though the means differ considerably between the two groups, older users' behaviour shows much greater variation than that of younger users. For example, for user initiative, the main range of values seen for older users includes the majority of values observed for younger users.

4 User Simulations

We performed 5-fold cross validation ensuring that there was no overlap in speakers between different folds. Each user utterance corresponds to a user action annotated as a list of ⟨speech act, task⟩ pairs. For example, the utterance "I'd like to see the diabetes nurse on Thursday morning" could be annotated as [(accept_info, hp), (provide_info, half-day)] or similarly, depending on the previous system prompt. There are 389 distinct actions for older people and 125 for younger people. The actions of the younger people are a subset of the actions of the older people.

We built n-grams of system and user actions with n varying from 2 to 5. Given a history of system and user actions (n-1 actions) the SU generates an action based on a probability distribution learned from the training data (Georgila et al., 2006). We tested four values of n, 2, 3, 4, and 5. For reasons of space, we only report results from 3-grams because they suffer less from data sparsity than 4- and 5-grams and take into account larger contexts than 2-grams. However, results are similar for all values of n.

The actions generated by our SUs were compared to the actions observed in the corpus using five metrics proposed in the literature (Schatzmann et al., 2005; Georgila et al., 2006): perplexity (PP), precision, recall, expected precision and expected recall. While precision and recall are calculated based on the most likely action at a given state, expected precision and expected recall take into account all possible user actions at a given state. Details are given in (Georgila et al., 2006). In our cross-validation experiments, we used three different sources for the training and test sets: data from older users (O), data

	PP	Prec	Rec	ExpPrec	ExpRec
O-O	18.1	**42.8**	**39.8**	**56.0**	**49.4**
Y-O	19.6	**34.2**	**25.1**	**53.4**	**40.7**
A-O	18.7	**41.1**	**35.9**	**58.9**	**49.0**
O-Y	5.7	44.8	60.6	66.3	73.4
Y-Y	3.7	50.5	54.1	73.1	70.4
A-Y	3.8	45.8	58.5	70.5	73.0
O-A	10.3	**43.7**	**47.2**	60.3	**58.0**
Y-A	9.3	**40.3**	**33.3**	62.0	**51.5**
A-A	9.3	**43.2**	**43.4**	63.9	**57.9**

Table 2: Results for 3-grams and different combinations of training and test data. O: older users, Y: younger users, A: all users.

from younger users (Y), and data from all users (A). Our results are summarised in Table 2.

We find that models trained on younger users, but tested on older users (Y-O) perform worse than models trained on older users / all users and tested on older users (O-O, A-O). Thus, models of the behaviour of younger users cannot be used to simulate older users. In addition, models which are trained on older users tend to generalise better to the whole data set (O-A) than models trained only on younger users (Y-A). These results are in line with our statistical analysis, which showed that the behaviour of younger users appears to be a subset of the behaviour of older users. All results are statistically significant at $p<0.05$ or better.

5 Conclusions

In this paper we built user simulations for older and younger adults and evaluated them using standard metrics. Our results suggest that SUs trained on older people may also cover the behaviour of younger users, but not vice versa. This finding supports the principle of "inclusive design" (Keates and Clarkson, 2004): designers should consider a wide range of users when developing a product for general use. Furthermore, our results agree with predictions based on statistical analysis of our corpus. They are also in line with findings of tests of deployed Interactive Voice Response systems with younger and older users (Dulude, 2002), which show the diversity of older people's behaviour. Therefore, we have shown that standard metrics for evaluating SUs are a good predictor of the behaviour of our two user types. Overall, the metrics we used yielded a clear and consistent picture. Although our result needs to be verified on similar corpora, it has an important implication for corpus design. In order to yield realistic models of user behaviour, we need to gather less data from students, and more data from older and middle-aged users.

In our future work, we will perform more detailed statistical analyses of user behaviour. In particular, we will analyse the effect of dialogue strategies on behaviour, experiment with different Bayesian network structures, and use the resulting user simulations to learn dialogue strategies for both older and younger users as another way for testing the accuracy of our user models and validating our results.

Acknowledgements

This research was supported by the Wellcome Trust VIP grant and the Scottish Funding Council grant MATCH (HR04016). We would like to thank Robert Logie and Sarah MacPherson for contributing to the design of the original experiment, Neil Mayo and Joe Eddy for coding the Wizard-of-Oz interface, Vasilis Karaiskos and Matt Watson for collecting the data, and Melissa Kronenthal for transcribing the dialogues.

References

M. Aylett, C. Pidcock, and M.E. Fraser. 2006. The Cerevoice Blizzard Entry 2006: A prototype database unit selection engine. In *Proc. BLIZZARD Challenge*.

S. Czaja and C. Lee. 2007. The impact of aging on access to technology. *Universal Access in the Information Society (UAIS)*, 5:341–349.

L. Dulude. 2002. Automated telephone answering systems and aging. *Behaviour Information Technology*, 21:171–184.

K. Georgila, J. Henderson, and O. Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Proc. Interspeech/ICSLP*.

K. Georgila, M. Wolters, V. Karaiskos, M. Kronenthal, R. Logie, N. Mayo, J. Moore, and M. Watson. 2008. A fully annotated corpus for studying the effect of cognitive ageing on users' interactions with spoken dialogue systems. In *Proc. LREC*.

S. Keates and J. Clarkson. 2004. *Inclusive Design*. Springer, London.

N. Roy, J. Pineau, and S. Thrun. 2000. Spoken dialog management for robots. In *Proc. ACL*.

J. Schatzmann, K. Georgila, and S. Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proc. SIGdial*.

M. Wolters, P. Campbell, C. DePlacido, A. Liddell, and D. Owens. 2007. Making synthetic speech accessible to older people. In *Proc. Sixth ISCA Workshop on Speech Synthesis, Bonn, Germany*.

Active Sample Selection for Named Entity Transliteration

Dan Goldwasser **Dan Roth**
Department of Computer Science
University of Illinois
Urbana, IL 61801
{goldwas1,danr}@uiuc.edu

Abstract

This paper introduces a new method for identifying named-entity (NE) transliterations within bilingual corpora. Current state-of-the-art approaches usually require annotated data and relevant linguistic knowledge which may not be available for all languages. We show how to effectively train an accurate transliteration classifier using very little data, obtained automatically. To perform this task, we introduce a new active sampling paradigm for guiding and adapting the sample selection process. We also investigate how to improve the classifier by identifying repeated patterns in the training data. We evaluated our approach using English, Russian and Hebrew corpora.

1 Introduction

This paper presents a new approach for constructing a discriminative transliteration model.
Our approach is fully automated and requires little knowledge of the source and target languages.
Named entity (NE) transliteration is the process of transcribing a NE from a source language to a target language based on phonetic similarity between the entities. Figure 1 provides examples of NE transliterations in English Russian and Hebrew.

Identifying transliteration pairs is an important component in many linguistic applications such as machine translation and information retrieval, which require identifying out-of-vocabulary words.
In our settings, we have access to source language NE and the ability to label the data upon request. We introduce a new active sampling paradigm that aims to guide the learner toward informative samples, allowing learning from a small number of representative examples. After the data is obtained it is analyzed to identify repeating patterns which can be used to focus the training process of the model.

English NE	Russian NE	Hebrew NE
Saint Petersburg	Санкт Петербург	סנקט פטרבורג

Figure 1: NE in English, Russian and Hebrew.

Previous works usually take a generative approach, (Knight and Graehl, 1997). Other approaches exploit similarities in aligned bilingual corpora; for example, (Tao et al., 2006) combine two unsupervised methods. (Klementiev and Roth, 2006) bootstrap with a classifier used interchangeably with an unsupervised temporal alignment method. Although these approaches alleviate the problem of obtaining annotated data, other resources are still required, such as a large aligned bilingual corpus.
The idea of selectively sampling training samples has been wildly discussed in machine learning theory (Seung et al., 1992) and has been applied successfully to several NLP applications (McCallum and Nigam, 1998). Unlike other approaches,our approach is based on minimizing the distance between the feature distribution of a comprehensive reference set and the sampled set.

2 Training a Transliteration Model

Our framework works in several stages, as summarized in Algorithm 1. First, a training set consisting

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 53–56,
Columbus, Ohio, USA, June 2008.

of NE transliteration pairs (w_s, w_t) is automatically generated using an active sample selection scheme. The sample selection process is guided by the Sufficient Spanning Features criterion (SSF) introduced in section 2.2, to identify informative samples in the source language.An oracle capable of pairing a NE in the source language with its counterpart in the target language is then used. Negative training samples are generated by reshuffling the terms in these pairs. Once the training data has been collected, the data is analyzed to identify repeating patterns in the data which are used to focus the training process by assigning weights to features corresponding to the observed patterns. Finally, a linear model is trained using a variation of the averaged perceptron (Freund and Schapire, 1998) algorithm. The remainder of this section provides details about these stages; the basic formulation of the transliteration model and the feature extraction scheme is described in section 2.1, in section 2.2 the selective sampling process is described and finally section 2.3 explains how learning is focused by using feature weights.

Input: Bilingual, comparable corpus ($\mathcal{S}$, $\mathcal{T}$), set of named entities NE_S from $\mathcal{S}$, Reference Corpus R_S, Transliteration Oracle O, Training Corpora D=D_S,D_T
Output: Transliteration model $\mathcal{M}$

Guiding the Sampling Process
repeat
  select a set C $\subseteq NE_S$ randomly
  $w_s = argmin_{w \in C} distance(R, D_S \cup \{w_s\})$
  $D = D \cup \{W_s, O(W_s)\}$
until *distance(R,*$D_S \cup \{W_s\}$*)* $\geq$ *distance(R,*D_S*)* ;
Determining Features Activation Strength
Define W:$f \rightarrow \Re$ s.t. foreach feature f=$\{f_s, f_t\}$
$W(f) = \frac{\sharp(f_s,f_t)}{\sharp(f_s)} \times \frac{\sharp(f_s,f_t)}{\sharp(f_t)}$
Use $\mathcal{D}$ to train $\mathcal{M}$;

Algorithm 1: Constructing a transliteration model.

2.1 Transliteration Model

Our transliteration model takes a discriminative approach; the classifier is presented with a word pair (w_s, w_t) , where w_s is a named entity and it is asked to determine whether w_t is a transliteration

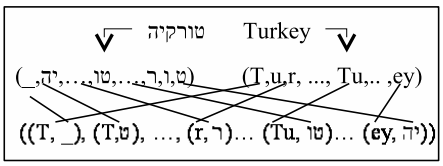


Figure 2: Features extraction process

of the NE in the target language. We use a linear classifier trained with a regularized perceptron update rule (Grove and Roth, 2001) as implemented in SNoW, (Roth, 1998). The classifier's confidence score is used for ranking of positively tagged transliteration candidates. Our initial feature extraction scheme follows the one presented in (Klementiev and Roth, 2006), in which the feature space consists of n-gram pairs from the two languages. Given a sample, each word is decomposed into a set of substrings of up to a given length (including the empty string). Features are generated by pairing substrings from the two sets whose relative positions in the original words differ by one or less places; first each word is decomposed into a set of substrings then substrings from the two sets are coupled to complete the pair representation. Figure 2 depicts this process.

2.2 Guiding the Sampling Process with SSF

The initial step in our framework is to generate a training set of transliteration pairs; this is done by pairing highly informative source language candidate NEs with target language counterparts. We developed a criterion for adding new samples, Sufficiently Spanning Features (SSF), which quantifies the sampled set ability to span the feature space. This is done by evaluating the L-1 distance between the frequency distributions of source language word fragments in the current sampled set and in a comprehensive set of source language NEs, serving as reference. We argue that since the features used for learning are n-gram features, once these two distributions are close enough, our examples space provides a good and concise characterization of all named entities we will ever need to consider. A special care should be given to choosing an appropriate reference; as a general guideline the reference set should be representative of the testing data. We collected a set R, consisting

of 50,000 NE by crawling through Wikipedia's articles and using an English NER system available at - *http://L2R.cs.uiuc.edu/ cogcomp*. The frequency distribution was generated over all character level bi-grams appearing in the text, as bi-grams best correlate with the way features are extracted. Given a reference text *R*, the n-grams distribution of *R* can be defined as follows -$D_R(ng_i) = \frac{\sharp ng_i}{\sum_j \sharp ng_j}$,where *ng* is an n-gram in R. Given a sample set *S*, we measure the L_1 distance between the distributions:
distance (R,S) = $\sum_{ng \in R} \mid D_R(ng) - D_S(ng) \mid$ Samples decreasing the distance between the distributions were added to the training data. Given a set C of candidates for annotation, a sample $w_s \in C$ was added to the training set, if -
$w_s = argmin_{w \in C} distance(R, D_S \cup \{w_s\})$.
A sample set is said to have SSF, if the distance remains constant as more samples are added.

2.2.1 Transliteration Oracle Implementation

The transliteration oracle is essentially a mapping between the named entities, i.e. given an NE in the source language it provides the matching NE in the target language. An automatic oracle was implemented by crawling through Wikipedia topic aligned document pairs. Given a pair of topic aligned documents in the two languages, the topic can be identified either by identifying the top ranking terms or by simply identifying the title of the documents. By choosing documents in Wikipedia's biography category we ensured that the topic of the documents is person NE.

2.3 Training the transliteration model

The feature extraction scheme we use generates features by coupling substrings from the two terms. Ideally, given a positive sample, it is desirable that paired substrings would encode phonetically similar or a distinctive context in which the two scripts correlate. Given enough positive samples, such features will appear with distinctive frequency. Taking this idea further, these features were recognized by measuring the co-occurrence frequency of substrings of up to two characters in both languages. Each feature *f=*(f_s, f_t) composed of two substrings taken from English and Hebrew words was associated with weight. $W(f) = \frac{\sharp(f_s,f_t)}{\sharp(f_s)} \times \frac{\sharp(f_s,f_t)}{\sharp(f_t)}$ where $\sharp(f_s, f_t)$ is the number of occurrences of that feature in the positive sample set, and $\sharp(f_L)$ is the number of occurrences of an individual substring, in any of the features extracted from positive samples in the training set. The result of this process is a weight table, in which, as we empirically tested, the highest ranking weights were assigned to features that preserve the phonetic correlation between the two languages. To improve the classifier's learning rate, the learning process is focused around these features. Given a sample, the learner is presented with a real-valued feature vector instead of a binary vector, in which each value indicates both that the feature is active and its activation strength - i.e. the weight assigned to it.

Data Set	Method	Rus	Heb
1	SSF	0.68	NA
1	KR'06	0.63	NA
2	SSF	0.71	0.52

Table 1: Results summary. The numbers are the proportion of NE recognized in the target language. Lines 1 and 2 compare the results of SSF directed approach with the baseline system on the first dataset. Line 3 summarizes the results on the second dataset.

3 Evaluation

We evaluated our approach in two settings; first, we compared our system to a baseline system described in (Klementiev and Roth, 2006). Given a bilingual corpus with the English NE annotated, the system had to discover the NE in target language text. We used the English-Russian news corpus used in the baseline system. NEs were grouped into equivalence classes, each containing different variations of the same NE. We randomly sampled 500 documents from the corpus. Transliteration pairs were mapped into 97 equivalence classes, identified by an expert. In a second experiment, different learning parameters such as selective sampling efficiency and feature weights were checked. 300 English-Russian and English-Hebrew NE pairs were used; negative samples were generated by coupling every English NE with all other target language NEs. Table 1 presents the key results of these experiments and compared with the baseline system.

Extraction method	Number of samples	Recall Top one	Recall Top two
Directed	200	0.68	0.74
Random	200	0.57	0.65
Random	400	0.63	0.71

Table 2: Comparison of correctly identified English-Russian transliteration pairs in news corpus. The model trained using selective sampling outperforms models trained using random sampling, even when trained with twice the data. The top one and top two results columns describe the proportion of correctly identified pairs ranked in the first and top two places, respectively.

Learning		Russian		Hebrew	
Training	Feature weights	Top one	Top five	Top one	Top five
+	+	0.71	0.89	0.52	0.88
-	+	0.63	0.82	0.33	0.59
+	-	0.64	0.79	0.37	0.68

Table 3: The proportion of correctly identified transliteration pairs with/out using weights and training. The top one and top five results columns describe the proportion of correctly identified pairs ranked in the first place and in any of the top five places, respectively. The results demonstrate that using feature weights improves performance for both target languages.

3.1 Using SSF directed sampling

Table 2 describes the effect of directed sampling in the English-Russian news corpora NE discovery task. Results show that models trained using selective sampling can outperform models trained with more than twice the amount of data.

3.2 Training using feature weights

Table 3 describes the effect training the model with weights.The training set consisted of 150 samples extracted using SSF directed sampling. Three variations were tested - training without feature weights, using the feature weights as the initial network weights without training and training with weights. The results clearly show that using weights for training improve the classifier's performance for both Russian and Hebrew. It can also be observed that in many cases the correct pair was ranked in any of the top five places.

4 Conclusions and future work

In this paper we presented a new approach for constructing a transliteration model automatically and efficiently by selectively extracting transliteration samples covering relevant parts of the feature space and focusing the learning process on these features. We show that our approach can outperform systems requiring supervision, manual intervention and a considerable amount of data. We propose a new measure for selective sample selection which can be used independently. We currently investigate applying it in other domains with potentially larger feature space than used in this work. Another aspect investigated is using our selective sampling for adapting the learning process for data originating from different sources; using the a reference set representative of the testing data, training samples, originating from a different source , can be biased towards the testing data.

5 Acknowledgments

Partly supported by NSF grant ITR IIS-0428472 and DARPA funding under the Bootstrap Learning Program.

References

Y. Freund and R. E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *COLT*.

A. Grove and D. Roth. 2001. Linear concepts and hidden variables. *ML*, 42.

A. Klementiev and D. Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *ACL*.

K. Knight and J. Graehl. 1997. Machine transliteration. In *EACL*.

D. K. McCallum and K. Nigam. 1998. Employing EM in pool-based active learning for text classification. In *ICML*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *AAAI*.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *COLT*.

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

Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation

Nizar Habash
Center for Computational Learning Systems
Columbia University
habash@ccls.columbia.edu

Abstract

We present four techniques for online handling of Out-of-Vocabulary words in Phrase-based Statistical Machine Translation. The techniques use spelling expansion, morphological expansion, dictionary term expansion and proper name transliteration to reuse or extend a phrase table. We compare the performance of these techniques and combine them. Our results show a consistent improvement over a state-of-the-art baseline in terms of BLEU and a manual error analysis.

1 Introduction

We present four techniques for *online* handling of Out-of-Vocabulary (OOV) words in phrase-based Statistical Machine Translation (SMT).[1] The techniques use morphological expansion (MORPHEX), spelling expansion (SPELLEX), dictionary word expansion (DICTEX) and proper name transliteration (TRANSEX) to reuse or extend phrase tables online. We compare the performance of these techniques and combine them. We work with a standard Arabic-English SMT system that has been already optimized for minimizing data sparsity through the use of morphological preprocessing and orthographic normalization. Thus our baseline token OOV rate is rather low (average 2.89%). None of our techniques are specific to Arabic and all can be retargeted to other languages given availability of technique-specific resources. Our results show that we improve over a state-of-the-art baseline by over 2.7% (relative BLEU score) and handle *all* OOV instances. An error analysis shows that, in 60% of the time, our OOV handling successfully produces acceptable output. Additionally, we still improve in BLEU score even as we increase our system's training data by 10-fold.

[1]This work was funded under the DARPA GALE program, contract HR0011-06-C-0023.

2 Related Work

Much work in MT has shown that orthographic and morpho-syntactic preprocessing of the training and test data reduces data sparsity and OOV rates. This is especially true for languages with rich morphology such as Spanish, Catalan, and Serbian (Popović and Ney, 2004) and Arabic (Sadat and Habash, 2006). We are interested in the specific task of *online OOV handling*. We will not consider *solutions* that game precision-based evaluation metrics by deleting OOVs. Some previous approaches anticipate OOV words that are potentially morphologically related to in-vocabulary (INV) words (Yang and Kirchhoff, 2006). Vilar et al. (2007) address spelling-variant OOVs in MT through online re-tokenization into letters and combination with a word-based system. There is much work on name transliteration and its integration in larger MT systems (Hassan and Sorensen, 2005). Okuma et al. (2007) describe a dictionary-based technique for translating OOV words in SMT. We differ from previous work on OOV handling in that we address spelling and name-transliteration OOVs in addition to morphological OOVs. We compare these different techniques and study their combination. Our morphology expansion technique is novel in that we automatically learn which source language morphological features are irrelevant to the target language.

3 Out-of-Vocabulary Words in Arabic-English Machine Translation

Arabic Linguistic Issues Orthographically, we distinguish three major challenges for Arabic processing. First, Arabic script uses *optional* vocalic diacritics. Second, certain letters in Arabic script are often spelled inconsistently, e.g., variants of Hamzated Alif, أ Â[2] or إ Ă, are often written without

[2]Arabic transliteration is provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007).

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 57–60,
Columbus, Ohio, USA, June 2008.

Hamza: أ *A*. Finally, Arabic's alphabet uses *obligatory* dots to distinguish different letters (e.g., ب *b*, ت *t* and ث *θ*). Each letter base is ambiguous two ways on average. Added or missing dots are often seen in spelling errors. Morphologically, Arabic is a rich language with a large set of morphological features such as gender, number, person and voice. Additionally, Arabic has a set of very common clitics that are written attached to the word, e.g., the conjunction و+ *w+* 'and'. We address some of these challenges in our baseline system by removing all diacritics, normalizing Alif and Ya forms, and tokenizing Arabic text in the highly competitive Arabic Treebank scheme (Sadat and Habash, 2006). This reduces our OOV rates by 59% relative to raw text. So our baseline is a real system with 2.89% token OOV rate. The rest of the challenges such as spelling errors and morphological variations are addressed by our OOV handling techniques.

Profile of OOV words in Arabic-English MT In a preliminary study, we manually analyzed a random sample of 400 sentences containing at least one OOV token extracted from the NIST MTEval data sets. There were 686 OOV tokens altogether. 40% of OOV cases involved proper nouns. 60% involved other parts-of-speech such as nouns (26.4%), verbs (19.3%) and adjectives (14.3%). The proper nouns seen come from different origins including Arabic, Hebrew, English, French, and Chinese. In many cases, the OOV words were less common morphological variants of INV words, such as the nominal dual form. The different techniques we discuss in the next section address these different issues in different ways. Proper name transliteration is primarily handled by TRANSEX. However, an OOV with a different spelling of an INV name can be handled by SPELLEX. Morphological variants are handled primarily by MORPHEX and DICTEX, but since some morphological variations involve small changes in lettering, SPELLEX may contribute too.

4 OOV-Handling Techniques

Our approach to handling OOVs is to extend the phrase table with possible translations of these OOVs. In MORPHEX and SPELLEX techniques, we match the OOV word with an INV word that is a possible variant of the OOV word. Phrases associated with the INV token in the phrase table are "recycled" to create new phrases in which the INV word is replaced with the OOV word. The translation weights of the INV phrase are used as is in the new phrase. We limit the added phrases to source-language unigrams and bigrams (determined empirically). In DICTEX and TRANSEX techniques, we add completely new entries to the phrase table. All the techniques could be used with other approaches, such as input-text lattice extension with INV variants of OOVs or their target translations. We briefly describe the techniques next. More details are available in a technical report (Habash, 2008).

MORPHEX We match the OOV word with an INV word that is a possible *morphological* variant of the OOV word. For this to work, we need to be able to morphologically analyze the OOV word (into lexeme and features). OOV words that fail morphological analysis cannot be helped by this technique. The morphological matching assumes the words to be matched agree in their lexeme but have different inflectional features. We collect information on possible inflectional variations from the original phrase table itself: in an off-line process, we cluster all the analyses of single-word Arabic entries in our phrase table that (a) translate into the same English phrase and (b) have the same lexeme analysis. From these clusters we learn which morphological inflectional features in Arabic are irrelevant to English. We create a rule set of morphological inflection maps that we then use to relate analyses of OOV words to analyses of INV words (which we create off-line for speedy use). The most common inflectional variation is the addition or deletion of the Arabic definite article ال+ *Al+*, which is part of the word in our tokenization.

SPELLEX We match the OOV token with an INV token that is a possible correct spelling of the OOV token. In our current implementation, we consider four types of spelling correction involving one letter only: letter deletion, letter insertion, letter inversion (of any two adjacent letters) and letter substitution. The following four misspellings of the word فلسطيني *flsTyny* 'Palestinian' correspond to these four types, respectively: فلسطني *flsTny*, فلسطيننني *flsTynny*, فلطسيني *flTsyny* and قلسطيني *qlsTyny*. We only allow letter substitution from a limited list of around 90 possible substitutions (as opposed to all 1260 possible substitutions). The substitutions we considered include cases we deemed harder than

usual to notice as spelling errors: common letter shape alternations (e.g., ر *r* and ز *z*), phonological alternations (e.g., ص *S* and س *s*) and dialectal variations (e.g., ق *q* and ىء ŷ). We do not handle misspellings involving two words attached to each other or multiple types of single letter errors in the same word.

DICTEX We extend the phrase table with entries from a manually created dictionary – the English glosses of the Buckwalter Arabic morphological analyzer (Buckwalter, 2004). For each analysis of an OOV word, we expand the English lemma gloss to all its possible surface forms. The newly generated pairs are equally assigned very low translation probabilities that do not interfere with the rest of the phrase table.

TRANSEX We produce English transliteration hypotheses that assume the OOV is a proper name. Our transliteration system is rather simple: it uses the transliteration similarity measure described by Freeman et al. (2006) to select a best match from a large list of possible names in English.[3] The list was collected from a large collection of English corpora primarily using capitalization statistics. For each OOV word, we produce a list of possible transliterations that are used to add translation pair entries in the phrase table. The newly generated pairs are assigned very low translation probabilities that do not interfere with the rest of the phrase table. Weights of entries were modulated by the degree of similarity indicated by the metric we used. Given the large number of possible matches, we only pass the top 20 matches to the phrase table. The following are some possible transliterations produced for the name باستور *bAstwr* together with their similarity scores: pasteur and pastor (1.00), pastory and pasturk (0.86) bistrot and bostrom (0.71).

5 Evaluation

Experimental Setup All of our training data is available from the Linguistic Data Consortium (LDC).[4] For our basic system, we use an Arabic-English parallel corpus[5] consisting of 131K sentence pairs, with approximately 4.1M Arabic tokens and 4.4M English tokens. Word alignment is done with GIZA++ (Och and Ney, 2003). All evaluated systems use the same surface trigram language model, trained on approximately 340 million words from the English Gigaword corpus (LDC2003T05) using the SRILM toolkit (Stolcke, 2002). We use the standard NIST MTEval data sets for the years 2003, 2004 and 2005 (henceforth MT03, MT04 and MT05, respectively).[6]

We report results in terms of case-insensitive 4-gram BLEU (Papineni et al., 2002) scores. The first 200 sentences in the 2002 MTEval test set were used for Minimum Error Training (MERT) (Och, 2003). We decode using Pharaoh (Koehn, 2004). We tokenize using the MADA morphological disambiguation system (Habash and Rambow, 2005), and TOKAN, a general Arabic tokenizer (Sadat and Habash, 2006). English preprocessing simply included down-casing, separating punctuation from words and splitting off "'s".

OOV Handling Techniques and their Combination We compare our baseline system (BASELINE) to each of our basic techniques and their full combination (ALL). Combination was done by using the union of all additions. In each setting, the extension phrases are added to the baseline phrase table. Our baseline phrase table has 3.5M entries. In our experiments, on average, MORPHEX handled 60% of OOVs and added 230 phrases per OOV; SPELLEX handled 100% of OOVs and added 343 phrases per OOV; DICTEX handled 73% of OOVs and added 11 phrases per OOV; and TRANSEX handled 93% of OOVs and added 16 phrases per OOV.

Table 1 shows the results of all these settings. The first three rows show the OOV rates for each test set. $\text{OOV}_{sentence}$ indicates the ratio of sentences with at least one OOV. The last two rows show the best absolute and best relative increase in BLEU scores above BASELINE. All conditions improve over BASELINE. Furthermore, the combination improved over BASELINE and its components. There is no clear pattern of technique rank across all test sets. The average increase in the best performing conditions is around 1.2% BLEU (absolute) or 2.7% (relative). These consistent improvements are not statistically significant. However, this is still a nice

[3] Freeman et al. (2006) report 80% F-score at 0.85 threshold.

[4] http://www.ldc.upenn.edu

[5] The parallel text includes Arabic News (LDC2004T17), eTIRR (LDC2004E72), Arabic Treebank with English translation (LDC2005E46), and Ummah (LDC2004T18).

[6] The following are the statistics of these data sets in terms of (sentences/tokens/types): MT03 (663/18,755/4,358), MT04 (1,353/42,774/8,418) and MT05(1,056/32,862/6,313). The data sets are available at http://www.nist.gov/speech/tests/mt/.

Table 1: OOV Rates (%) and BLEU Results of Using Different OOV Handling Techniques

	MT03	MT04	MT05
$\text{OOV}_{sentence}$	40.12	54.47	48.30
OOV_{type}	8.36	13.32	11.38
OOV_{token}	2.46	3.21	2.99
BASELINE	44.20	40.60	42.86
MORPHEX	44.79	41.18	43.37
SPELLEX	45.09	41.11	43.47
DICTEX	44.88	41.24	43.46
TRANSEX	44.83	40.90	43.25
ALL	**45.60**	**41.56**	**43.95**
Best Absolute	1.40	0.96	1.09
Best Relative	3.17	2.36	2.54

result given that we only focused on OOV words.

Scalability Evaluation To see how well our approach scales up, we added over 40M words (1.6M sentences) to our training data using primarily the UN corpus (LDC2004E13). As expected, the token OOV rates dropped from an average of 2.89% in our baseline to 0.98% in the scaled-up system. Our average baseline BLEU score went up from 42.60 to 45.00. However, using the ALL combination, we still increase the scaled-up system's score to an average BLEU of 45.28 (0.61% relative). The increase was seen on all data sets.

Error Analysis We conducted an informal error analysis of 201 random sentences in MT03 from BASELINE and ALL. There were 95 different sentences containing 141 OOV words. We judged words as *acceptable* or *wrong*. We only considered as *acceptable* cases that produce a correct translation or transliteration *in context*. Our OOV handling successfully produces acceptable translations in 60% of the cases. Non-proper-noun OOVs are well handled in 76% of the time as opposed to proper nouns which are only correctly handled in 40% of the time.

6 Conclusion and Future Plans

We have presented four techniques for handling OOV words in SMT. Our results show that we consistently improve over a state-of-the-art baseline in terms of BLEU, yet there is still potential room for improvement. The described system is publicly available. In the future, we plan to improve each of the described techniques; explore better ways of weighing added phrases; and study how these techniques function under different tokenization conditions in Arabic and with other languages.

References

T. Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium (LDC2004L02).

A. Freeman, S. Condon, and C. Ackerman. 2006. Cross Linguistic Name Matching in English and Arabic. In *Proc. of HLT-NAACL*.

N. Habash. 2008. Online Handling of Out-of-Vocabulary Words for Statistical Machine Translation. CCLS Technical Report.

N. Habash, A. Soudi and T. Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors. *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, Springer.

N. Habash and O. Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proc. of ACL'05*.

H. Hassan and J. Sorensen. 2005. An integrated approach for Arabic-English named entity translation. In *Proc. of the ACL Workshop on Computational Approaches to Semitic Languages*.

P. Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-based Statistical Machine Translation Models. In *Proc. of AMTA*.

F. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52.

F. Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. In *Proc. of ACL*.

H. Okuma, H. Yamamoto, and E. Sumita. 2007. Introducing translation dictionary into phrase-based SMT.. In *Proc. of MT Summit.*

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*.

M. Popović and H. Ney. 2004. Towards the Use of Word Stems and Suffixes for Statistical Machine Translation. In *Proc. of LREC.*

F. Sadat and N. Habash. 2006. Combination of Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proc. of ACL*.

A. Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proc. of ICSLP*.

D. Vilar, J. Peter, and H. Ney 2007. Can we translate letters?. In *Proc. of ACL workshop on SMT*.

M. Yang and K. Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proc. of EACL*.

Combined One Sense Disambiguation of Abbreviations

Yaakov HaCohen-Kerner
Department of Computer Science, Jerusalem College of Technology (Machon Lev)
21 Havaad Haleumi St., P.O.B. 16031, 91160 Jerusalem, Israel
kerner@jct.ac.il

Ariel Kass
Department of Computer Science, Jerusalem College of Technology (Machon Lev)
21 Havaad Haleumi St., P.O.B. 16031, 91160 Jerusalem, Israel
ariel.kass@gmail.com

Ariel Peretz
Department of Computer Science, Jerusalem College of Technology (Machon Lev)
21 Havaad Haleumi St., P.O.B. 16031, 91160 Jerusalem, Israel
relperetz@gmail.com

Abstract

A process that attempts to solve abbreviation ambiguity is presented. Various context-related features and statistical features have been explored. Almost all features are domain independent and language independent. The application domain is Jewish Law documents written in Hebrew. Such documents are known to be rich in ambiguous abbreviations. Various implementations of the one sense per discourse hypothesis are used, improving the features with new variants. An accuracy of 96.09% has been achieved by SVM.

1 Introduction

An abbreviation is a letter or sequence of letters, which is a shortened form of a word or a sequence of words, which is called the sense of the abbreviation. Abbreviation disambiguation means to choose the correct sense for a specific context.

Jewish Law documents written in Hebrew are known to be rich in ambiguous abbreviations (HaCohen-Kerner et al., 2004). They can, therefore, serve as an excellent test-bed for the development of models for abbreviation disambiguation.

As opposed to the documents investigated in previous systems, Jewish Law documents usually do not contain the sense of the abbreviations in the same discourse. Therefore, the abbreviations are regarded as more difficult to disambiguate.

This research defines features, as well as experiments with various variants of the one sense per discourse hypothesis. The developed process considers other languages and does not define pre-execution assumptions. The only limitation to this process is the input itself: the languages of the different text documents and the man-made solution database inputted during the learning process limit the datasets of documents that may be solved by the resulting disambiguation system.

The proposed system, preserves its portability between languages and domains because it does not use any natural language processing (NLP) sub-system (e.g.: tokenizer and tagger). In this matter, the system is not limited to any specific language or dataset. The system is only limited by the different inputs used during the system's learning stage and the set of abbreviations defined.

This paper is organized as follows: Section 2 presents previous systems dealing with disambiguation of abbreviations. Section 3 describes the features for disambiguation of Hebrew abbreviations. Section 4 presents the implementation of the one sense per discourse hypothesis. Section 5 describes the experiments that have been carried out. Section 6 concludes and proposes future directions for research.

2 Abbreviation Disambiguation

The one sense per collocation hypothesis was introduced by Yarowsky (1993). This hypothesis states that natural languages tend to use consistent spoken and written styles. Based on this hypothesis, many terms repeat themselves with the same meaning in all their occurrences. Within the context of determining the sense of an abbreviation, it may be assumed that authors tend to use the same words in the vicinity of a specific long form of an abbreviation. The words may be reused as indicators of the proper solution of an additional unknown abbreviation with the same words in its vicinity. This is the basis for all contextual features defined in this research.

The one sense per discourse hypothesis (OS) was introduced by Gale et al. (1992). This

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 61–64,
Columbus, Ohio, USA, June 2008.

hypothesis assumes that in natural languages, there is a tendency for an author to be consistent in the same discourse or article. That is, if in a specific discourse, an ambiguous phrase or term has a specific meaning, any other subsequent instance of this phrase or term will have the same specific meaning. Within the context of determining the sense of an abbreviation, it may be assumed that authors tend to use a specific abbreviation in a specific sense throughout the discourse or article.

Research has been done within this domain, mainly for English medical documents. Systems developed by Pakhomov (2002; 2005), Yu et al. (2003) and Gaudan et al. (2005) achieved 84% to 98% accuracy. These systems used various machine learning (ML) methods, e.g.: Maximum Entropy, SVM and C5.0.

In our previous research (HaCohen-Kerner et al., 2004), we developed a prototype abbreviation disambiguation system for Jewish Law documents written in Hebrew, without using any ML method. The system integrated six basic features: common words, prefixes, suffixes, two statistical features and a Hebrew specific feature. It achieved about 60% accuracy while solving 50 abbreviations with an average of 2.3 different senses in the dataset.

3 Abbreviation Disambiguation Features

Eighteen different features of any abbreviation instance were defined. They are divided into three distinct groups, as follows:

Statistical attributes: Writer/Dataset Common Rule (WC/DS). The most common solution used for the specific abbreviation by the discussed writer/ in the entire dataset.

Hebrew specific attribute: Gimatria Rule (GM). The numerical sum of the numerical values attributed to the Hebrew letters forming the abbreviation (HaCohen-Kerner et al., 2004).

Contextual relationship attributes:

1. **Prefix Counted Rule (PRC)**: The selected sense is the most commonly appended sense by the specific prefix.
2. **Before/After K (1,2,3,4) Words Counted Rule (BKWC/AKWC):** The selected sense is the most commonly preceded/succeeded sense by the K specific words in the sentence of the specific abbreviation instance.
3. **Before/After Sentence Counted Rule (BSC/ASC):** The selected sense is the most commonly preceded/succeeded sense by all the specific words in the sentence of the specific abbreviation instance.
4. **All Sentence/Article Counted Rule (AllSC/AllAC):** The selected sense is the most commonly surrounded sense by all the specific words in the sentence/article of the specific abbreviation instance.
5. **Before/After Article Counted Rule (BAC/AAC):** The selected sense is the most commonly preceded/succeeded sense by all the specific words in the article of the specific abbreviation instance.

4 Implementing the OS Hypothesis

As mentioned above, the basic assumption of the OS hypothesis is that there exists at least one solvable abbreviation in the discourse and that the sense of that abbreviation is the same for all the instances of this abbreviation in the discourse. The correctness of all the features was investigated based on this hypothesis for several variants of "one sense" based on the discussed discourse: none (No OS), a sentence (osS), an article (osA) or all the articles of the writer (osW).

The OS hypothesis was implemented in two forms. The “pure” form (with the suffix S/A/W without C) uses the sense found by the majority voting method for an abbreviation in the discourse and applies it “blindly” to all other instances.

The “combined” form (with the suffix C) tries to find the sense of the abbreviation using the discussed feature only. If the feature is unsuccessful, then we use the relevant one sense variant using the majority voting method. This form is derived from the possibility that more than one sense may be used within a single discourse and only instances with an unknown sense conform to the hypothesis.

The use of the OS hypothesis, in both forms, is only relevant for context based features, since the solutions by other features are static and identical from one instance to another.

Therefore, for each of the 15 context based features, 6 variants of the hypothesis were implemented. This produces 90 variants, which together with the 18 features in their normal form, results in a total of 108 variants. In addition, the ML methods were experimented together with the OS hypothesis. Of the 108 possible variants, for the 18 features, the best variant for each feature

was chosen. In each step, the next best variant is added, starting from the 2 best variants.

5 Experiments

The examined dataset includes Jewish Law Documents written by two Jewish scholars: Rabbi Y. M. HaCohen (1995) and Rabbi O. Yosef (1977; 1986). This dataset includes 564,554 words where 114,814 of them are abbreviations instances, and 42,687 of them are ambiguous. That is, about 7.5% of the words are ambiguous abbreviations. These ambiguous abbreviations are instances of a set of 135 different abbreviations. Each one of the abbreviations has between 2 to 8 relevant possible senses. The average number of senses for an abbreviation in the dataset is 3.27.

To determine the accuracy of the system, all the instances of the ambiguous abbreviations were solved beforehand. Some of them were based on published solutions (HaCohen, 1995) and some of them were solved by experienced readers.

5.1 Results of the variants of OS Hypothesis

The results of the OS hypothesis variants, for all the features, are presented in Table 1. These results are obtained without using any ML methods.

Use of OS / Feature	Accuracy Percentage %						
	No OS	osS	osSC	osA	osAC	osW	osWC
PRC	33.67	34.41	34.52	52.77	54.54	66.66	71.04
B1WC	56.05	56.41	56.61	67.74	71.84	72.93	82.51
B2WC	55.72	56.23	56.35	69	72.34	74.85	82.84
B3WC	60.54	60.89	61.01	72.67	75.48	75.44	82.86
B4WC	64.49	64.72	64.85	74.29	76.5	75.52	82.2
BSC	75.21	75.18	75.24	76.85	78.15	74.92	78.52
BAC	76	76	76	76.01	76	75.39	76
A1WC	**78.79**	79.01	79.21	78.72	83.81	76.32	**87.75**
A2WC	77.57	78.07	78.26	79.15	83.43	78.54	87.62
A3WC	78.64	79.11	79.28	79.61	83	78.19	85.8
A4WC	75.44	79.28	79.5	79.41	82.42	78.01	84.99
ASC	78.59	78.61	78.62	78.25	78.94	77.37	79.04
AAC	75.44	75.44	75.44	75.34	75.44	77.28	75.44
AllSC	77.97	77.97	77.97	77.9	78.02	77.22	78.04
AllAC	74.12	74.12	74.12	74.12	74.12	76.93	74.12
GM	**46.82**	46.82	46.82	46.82	46.82	46.82	**46.82**
WC	**82.84**	82.84	82.84	82.84	82.84	82.84	**82.84**
DC	**78.34**	78.34	78.34	78.34	78.34	78.34	**78.34**

Table 1. Results of the OS Variants for all the Features.

The two best pure features were WC and A1WC with 82.84% and 78.79% of accuracy, respectively. The first finding shows that about 83% of the abbreviations have the same sense in the whole dataset. The second finding shows that about 79% of the abbreviations can be solved by the first word that comes after the abbreviation.

Generally, contextual features based on the context that comes after the abbreviation, achieve considerably better results than all other contextual features. Specifically, the A1WC_osWC feature variant achieves the best result with 87.75% accuracy. These results suggest that each individual abbreviation has stronger relationship to the words after a specific instance, especially to the first word.

Almost every feature has at least one variant that achieves a substantial improvement in results compared the results achieved by the feature in its normal form. The average relative improvement is about 18%.

For all features, except BAC, the best variant uses the OS implementation with the discourse defined as the entire dataset. This may be attributed to the similarity of the different articles in the dataset. This is supported by the fact that the best feature, in its normal form, is the WC feature.

In addition, for all but three features (BAC, AAC, AllAC), the best variant used the combined form of the OS implementation. This is intuitively understandable, since "blindly" overwriting probably erases many successes of the feature in its normal form.

5.2 The Results of the Supervised ML Methods

Several well-known supervised ML methods have been selected: artificial neural networks (ANN), Naïve Bayes (NB), Support Vector Machines (SVM) and J48 (Witten and Frank, 1999) an improved variant of the C4.5 decision tree induction. These methods have been applied with default values and no feature normalization using Weka (Witten and Frank, 1999). Tuning is left for future research. To test the accuracy of the models, 10-fold cross-validation was used.

Table 2 presents the results of these supervised ML methods, by incrementally combining the best variant for each feature (according to Table 1).

Table 2 shows that SVM achieved the best result with 96.09% accuracy. The best improvement is

about 13%, from 82.84% accuracy for the best variant of any feature to 96.02% accuracy. This table also reveals that incremental combining of most of the variants leads to better results for most of the ML methods.

# of Vari-ants	Variants / ML Method	ANN	NB	SVM	J48
2	**A1WC_osWC +A2WC_osWC**	91.56	91.40	94.29	91.94
3	**+ A3WC_osWC**	91.72	91.42	94.43	92.20
4	**+ A4WC_osWC**	91.75	91.51	94.43	92.34
5	**+ B3WC_osWC**	92.68	92.11	95.33	93.33
6	**+ WC**	92.95	92.16	95.71	93.54
7	**+ B2WC_osWC**	92.81	91.79	95.67	93.59
8	**+ B1WC_osWC**	92.91	91.06	95.68	93.56
9	**+ B4WC_osWC**	92.83	91.15	95.62	93.55
10	**+ ASC_osWC**	92.83	91.10	95.60	93.52
11	**+ BSC_osWC**	92.95	91.17	95.65	93.58
12	**+ DC**	92.98	91.17	95.63	93.58
13	**+ AllSC_osWC**	92.82	91.50	95.63	93.58
14	**+ AAC_osW**	92.84	91.42	95.59	93.58
15	**+ AllAC_osW**	93.10	91.43	95.77	93.58
16	**+ BAC_osA**	93.09	91.28	95.79	93.70
17	**+ PRC_osWC**	93.25	91.50	**96.09**	93.71
18	**+ GM**	93.28	91.52	96.02	93.93

Table 2. The Results of the ML Methods.

The comparison of the SVM results to the results of previous (Section 2) shows that our system achieves relatively high accuracy. However, most previous systems researched ambiguous abbreviations in the English language, as well as different abbreviations and texts.

6 Conclusions, Summary and Future Work

This is the first ML system for disambiguation of abbreviations in Hebrew. High accuracy percentages were achieved, with improvement ascribed to the use of OS hypothesis combined with ML methods. These results were achieved without the use of any NLP features. Therefore, the developed system is adjustable to any specific type of texts, simply by changing the database of texts and abbreviations.

This system is the first that applies many versions of the one sense per discourse hypothesis. In addition, we performed a comparison between the achievements of four different standard ML methods, to the goal of achieving the best results, as opposed to the other systems that mainly focused on one ML method, each.

Future research directions are: comparison to abbreviation disambiguation using the standard bag-of-words or collocation feature representations, definition and implementation of other NLP-based features and use of these features interlaced with the already defined features, applying additional ML methods, and augmenting the databases with articles from additional datasets in the Hebrew language and other languages.

References

Y. M. Hacohen (Kagan). 1995. *Mishnah Berurah* (in Hebrew), Hotzaat Leshem, Jerusalem.

Y. HaCohen-Kerner, A. Kass and A. Peretz. 2004. *Baseline Methods for Automatic Disambiguation of Abbreviations in Jewish Law Documents*. Proceedings of the 4th International Conference on Advances in Natural Language LNAI, Springer Berlin/Heidelberg, 3230: 58-69.

W. Gale, K. Church and D. Yarowsky. 1992. *One Sense Per Discourse*. Proceedings of the 4th DARPA speech in Natural Language Workshop, 233-237.

S. Gaudan, H. Kirsch and D. Rebholz-Schuhmann. 2005. *Resolving abbreviations to their senses in Medline*. Bioinformatics, 21 (18): 3658-3664.

S. Pakhomov. 2002. *Semi-Supervised Maximum Entropy Based Approach to Acronym and Abbreviation Normalization in Medical Texts*. Association for Computational Linguistics (ACL), 160-167.

S. Pakhomov, T. Pedersen and C. G. Chute. 2005. *Abbreviation and Acronym Disambiguation in Clinical Discourse*. American Medical Informatics Association Annual Symposium, 589-593.

D. Yarowsky. 1993. *One sense per collocation*. Proceedings of the workshop on Human Language Technology, 266-271.

O. Yosef. 1977. *Yechave Daat* (in Hebrew), Publisher: Chazon Ovadia, Jerusalem.

O. Yosef. 1986. *Yabia Omer* (in Hebrew), Publisher: Chazon Ovadia, Jerusalem.

Z. Yu, Y. Tsuruoka and J. Tsujii. 2003. *Automatic Resolution of Ambiguous Abbreviations in Biomedical Texts using SVM and One Sense Per Discourse Hypothesis*. SIGIR'03 Workshop on Text Analysis and Search for Bioinformatics.

H. Witten and E. Frank. 2007. *Weka 3.4.12: Machine Learning Software in Java:* http://www.cs.waikato.ac.nz/~ml/weka.

Assessing the Costs of Sampling Methods in Active Learning for Annotation

Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, Peter McClanahan
Department of Computer Science
Brigham Young University
Provo, UT 84602, USA
robbie_haertel@byu.edu, ringger@cs.byu.edu, kseppi@cs.byu.edu,
jlcarroll@gmail.com, petermcclanahan@gmail.com

Abstract

Traditional Active Learning (AL) techniques assume that the annotation of each datum costs the same. This is not the case when annotating sequences; some sequences will take longer than others. We show that the AL technique which performs best depends on how cost is measured. Applying an hourly cost model based on the results of an annotation user study, we approximate the amount of time necessary to annotate a given sentence. This model allows us to evaluate the effectiveness of AL sampling methods in terms of time spent in annotation. We acheive a 77% reduction in hours from a random baseline to achieve 96.5% tag accuracy on the Penn Treebank. More significantly, we make the case for measuring cost in assessing AL methods.

1 Introduction

Obtaining human annotations for linguistic data is labor intensive and typically the costliest part of the acquisition of an annotated corpus. Hence, there is strong motivation to reduce annotation costs, but not at the expense of quality. Active learning (AL) can be employed to reduce the costs of corpus annotation (Engelson and Dagan, 1996; Ringger et al., 2007; Tomanek et al., 2007). With the assistance of AL, the role of the human oracle is either to label a datum or simply to correct the label from an automatic labeler. For the present work, we assume that correction is less costly than annotation from scratch; testing this assumption is the subject of future work. In AL, the learner leverages newly provided annotations to select more informative sentences which in turn can be used by the automatic labeler to provide more accurate annotations in future iterations. Ideally, this process yields accurate labels with less human effort.

Annotation cost is project dependent. For instance, annotators may be paid for the number of annotations they produce or by the hour. In the context of parse tree annotation, Hwa (2004) estimates cost using the number of constituents needing labeling and Osborne & Baldridge (2004) use a measure related to the number of possible parses. With few exceptions, previous work on AL has largely ignored the question of actual labeling *time*. One exception is (Ngai and Yarowsky, 2000) (discussed later) which compares the cost of manual rule writing with AL-based annotation for noun phrase chunking. In contrast, we focus on the performance of AL algorithms using different estimates of cost (including time) for part of speech (POS) tagging, although the results are applicable to AL for sequential labeling in general. We make the case for measuring cost in assessing AL methods by showing that the choice of a cost function significantly affects the choice of AL algorithm.

2 Benefit and Cost in Active Learning

Every annotation task begins with a set of unannotated items $\mathcal{U}$. The ordered set $\mathcal{A} \subseteq \mathcal{U}$ consists of all annotated data after annotation is complete or after available financial resources (or time) have been exhausted. We expand the goal of AL to produce the annotated set $\hat{\mathcal{A}}$ such that the benefit gained is maximized and cost is minimized.

In the case of POS tagging, tag accuracy is usu-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 65–68,
Columbus, Ohio, USA, June 2008.

ally used as the measure of benefit. Several heuristic AL methods have been investigated for determining which data will provide the most information and hopefully the best accuracy. Perhaps the best known are Query by Committee (QBC) (Seung et al., 1992) and uncertainty sampling (or Query by Uncertainty, QBU) (Thrun and Moeller, 1992). Unfortunately, AL algorithms such as these ignore the cost term of the maximization problem and thus assume a uniform cost of annotating each item. In this case, the ordering of annotated data $\mathcal{A}$ will depend entirely on the algorithm's estimate of the expected benefit.

However, for AL in POS tagging, the cost term may not be uniform. If annotators are required to change only those automatically generated tags that are incorrect, and depending on how annotators are paid, the cost of tagging one sentence can depend greatly on what is known from sentences already annotated. Thus, in POS tagging both the benefit (increase in accuracy) and cost of annotating a sentence depend not only on properties of the sentence but also on the order in which the items are annotated.

Therefore, when evaluating the performance of an AL technique, cost should be taken into account. To illustrate this, consider some basic AL algorithms evaluated using several simple cost metrics. The results are presented against a random baseline which selects sentences at random; the learning curves represent the average of five runs starting from a random initial sentence. If annotators are paid by the sentence, Figure 1(a) presents a learning curve indicating that the AL policy that selects the longest sentence (LS) performs rather well. Figure 1(a) also shows that given this cost model, QBU and QBC are essentially tied, with QBU enjoying a slight advantage. This indicates that if annotators are paid by the sentence, QBU is the best solution, and LS is a reasonable alternative. Next, Figure 1(b) illustrates that the results differ substantially if annotators are paid by the word. In this case, using LS as an AL policy is worse than random selection. Furthermore, QBC outperforms QBU. Finally, Figure 1(c) shows what happens if annotators are paid by the number of word labels corrected. Notice that in this case, the random selector marginally outperforms the other techniques. This is because QBU, QBC, and LS tend to select data that require many corrections. Considered together, Figures 1(a)-Figure 1(c) show the significant impact of choosing a cost model on the relative performance of AL algorithms. This leads us to conclude that AL techniques should be evaluated and compared with respect to a specific cost function.

While not all of these cost functions are necessarily used in real-life annotation, each can be regarded as an important component of a cost model of payment by the hour. Since each of these functions depends on factors having a significant effect on the perceived performance of the various AL algorithms, it is important to combine them in a way that will accurately reflect the true performance of the selection algorithms.

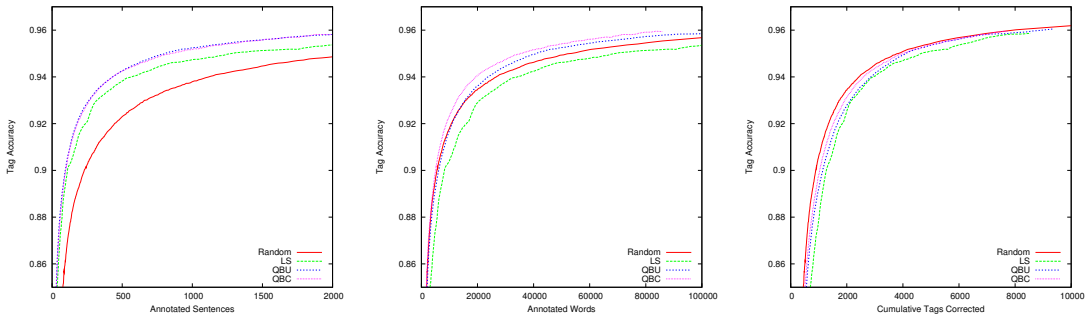
In prior work, we describe such a cost model for POS annotation on the basis of the time required for annotation (Ringger et al., 2008). We refer to this model as the "hourly cost model". This model is computed from data obtained from a user study involving a POS annotation task. In the study, timing information was gathered from many subjects who annotated both sentences and individual words. This study included tests in which words were prelabeled with a candidate labeling obtained from an automatic tagger (with a known error rate) as would occur in the context of AL. Linear regression on the study data yielded a model of POS annotation cost:

$$h = (3.795 \cdot l + 5.387 \cdot c + 12.57)/3600 \qquad (1)$$

where h is the time in hours spent on the sentence, l is the number of tokens in the sentence, and c is the number of words in the sentence needing correction. For this model, the Relative Standard Error (RSE) is 89.5, and the adjusted correlation (R^2) is 0.181. This model reflects the abilities of the annotators in the study and may not be representative of annotators in other projects. However, the purpose of this paper is to create a framework for accounting for cost in AL algorithms. In contrast to the model presented by Ngai and Yarowsky (2000), which predicts monetary cost given time spent, this model estimates time spent from characteristics of a sentence.

3 Evaluation Methodology and Results

Our test data consists of English prose from the POS-tagged Wall Street Journal text in the Penn Treebank (PTB) version 3. We use sections 2-21 as



(a) (b) (c)

Figure 1: QBU, LS, QBC, and the random baseline plotted in terms of accuracy versus various cost functions: (a) number of sentences annotated; (b) number of words annotated; and (c) number of tags corrected.

initially unannotated data. We employ section 24 as the development test set on which tag accuracy is computed at the end of every iteration of AL.

For tagging, we employ an order two Maximum Entropy Markov Model (MEMM). For decoding, we found that a beam of size five sped up the decoder with almost no degradation in accuracy from Viterbi. The features used in this work are typical for modern MEMM POS tagging and are mostly based on work by Toutanova and Manning (2000).

In our implementation, QBU employs a single MEMM tagger. We approximate the entropy of the per-sentence tag sequences by summing over per-word entropy and have found that this approximation provides equivalent performance to the exact sequence entropy. We also consider another selection algorithm introduced in (Ringger et al., 2007) that eliminates the overhead of entropy computations altogether by estimating per-sentence uncertainty with $1 - P(\hat{\underline{t}})$, where $\hat{\underline{t}}$ is the Viterbi (best) tag sequence. We label this scheme QBUOMM (OMM = "One Minus Max").

Our implementation of QBC employs a committee of three MEMM taggers to balance computational cost and diversity, following Tomanek et al. (2007). Each committee member's training set is a random bootstrap sample of the available annotated data, but is otherwise as described above for QBU. We follow Engelson & Dagan (1996) in the implementation of vote entropy for sentence selection using these models.

When comparing the relative performance of AL algorithms, learning curves can be challenging to interpret. As curves proceed to the right, they can approach one another so closely that it may be difficult to see the advantage of one curve over another. For this reason, we introduce the "cost reduction curve". In such a curve, the accuracy is the independent variable. We then compute the percent reduction in cost (e.g., number of words or hours) over the cost of the random baseline for the same accuracy a:

$$redux(a) = (cost_{rnd}(a) - cost(a))/cost_{rnd}(a)$$

Consequently, the random baseline represents the trajectory $redux(a) = 0.0$. Algorithms less costly than the baseline appear above the baseline. For a specific accuracy value on a learning curve, the corresponding value of the cost on the random baseline is estimated by interpolation between neighboring points on the baseline. Using hourly cost, Figure 2 shows the cost reduction curves of several AL algorithms, including those already considered in the learning curves of Figure 1 (except LS). Restricting the discussion to the random baseline, QBC, and QBU: for low accuracies, random selection is the cheapest according to hourly cost; QBU begins to be cost-effective at around 91%; and QBC begins to outperform the baseline and QBU around 80%.

4 Normalized Methods

One approach to convert existing AL algorithms into cost-conscious algorithms is to normalize the results of the original algorithm by the estimated cost. It should be somewhat obvious that many selection algorithms are inherently length-biased for sequence labeling tasks. For instance, since QBU is the sum

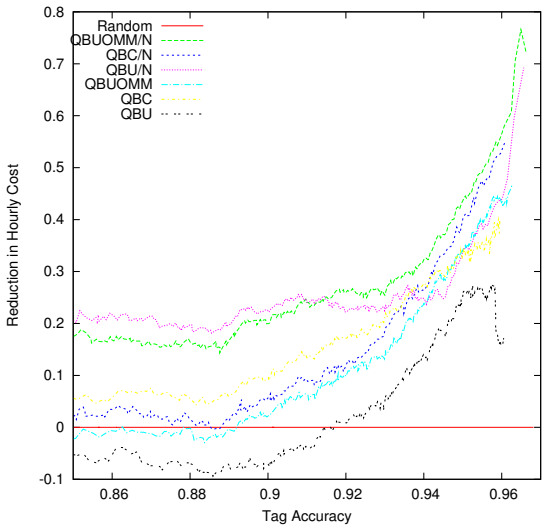


Figure 2: Cost reduction curves for QBU, QBC, QBUOMM, their normalized variants, and the random baseline on the basis of hourly cost

of entropy over all words, longer sentences will tend to have higher uncertainty. The easiest solution is to normalize by sentence length, as has been done previously (Engelson and Dagan, 1996; Tomanek et al., 2007). This of course assumes that annotators are paid by the word, which may or may not be true. Nevertheless, this approach can be justified by the hourly cost model. Replacing the number of words needing correction, c, with the product of l (the sentence length) and the accuracy p of the model, equation 1 can be re-written as the estimate:

$$\hat{h} = ((3.795 + 5.387p) \cdot l + 12.57)/3600$$

Within a single iteration of AL, p is constant, so the cost is approximately proportional to the length of the sentence. Figure 2 shows that normalized AL algorithms (suffixed with "/N") generally outperform the standard algorithms based on hourly cost (in contrast to the cost models used in Figures 1(a) - (c)). All algorithms shown have significant cost savings over the random baseline for accuracy levels above 92%. Furthermore, all algorithms except QBU depict trends of further increasing the advantage after 95%. According to the hourly cost model, QBUOMM/N has an advantage over all other algorithms for accuracies over 91%, achieving a significant 77% reduction in cost at 96.5% accuracy.

5 Conclusions

We have shown that annotation cost affects the assessment of AL algorithms used in POS annotation and advocate the use of a cost estimate that best estimates the true cost. For this reason, we employed an hourly cost model to evaluate AL algorithms for POS annotation. We have also introduced the cost reduction plot in order to assess the cost savings provided by AL. Furthermore, inspired by the notion of cost, we evaluated normalized variants of well-known AL algorithms and showed that these variants out-perform the standard versions with respect to the proposed hourly cost measure. In future work we will build better cost-conscious AL algorithms.

References

S. Engelson and I. Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. In *Proc. of ACL*, pages 319–326.

R. Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30:253–276.

G. Ngai and D. Yarowsky. 2000. Rule writing or annotation: cost-efficient resource usage for base noun phrase chunking. In *Proc. of ACL*, pages 117–125.

M. Osborne and J. Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proc. of HLT-NAACL*, pages 89–96.

E. Ringger, P. McClanahan, R. Haertel, G. Busby, M. Carmen, J. Carroll, K. Seppi, and D. Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proc. of Linguistic Annotation Workshop*, pages 101–108.

E. Ringger, M. Carmen, R. Haertel, K. Seppi, D. Lonsdale, P. McClanahan, J. Carroll, and N. Ellison. 2008. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proc. of LREC*.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proc. of CoLT*, pages 287–294.

S. Thrun and K. Moeller. 1992. Active exploration in dynamic environments. In *NIPS*, volume 4, pages 531–538.

K. Tomanek, J. Wermter, and U. Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. *Proc. of EMNLP-CoNLL*, pages 486–495.

K. Toutanova and C. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proc. of EMNLP*, pages 63–70.

Blog Categorization Exploiting Domain Dictionary and Dynamically Estimated Domains of Unknown Words

Chikara Hashimoto
Graduate School of Science and Engineering
Yamagata University
Yonezawa-shi, Yamagata, 992-8510, Japan
ch@yz.yamagata-u.ac.jp

Sadao Kurohashi
Graduate School of Informatics
Kyoto University
Sakyo-ku, Kyoto, 606-8501, Japan
kuro@i.kyoto-u.ac.jp

Abstract

This paper presents an approach to text categorization that **i)** uses no machine learning and **ii)** reacts on-the-fly to unknown words. These features are important for categorizing Blog articles, which are updated on a daily basis and filled with newly coined words. We categorize 600 Blog articles into 12 domains. As a result, our categorization method achieved an accuracy of 94.0% (564/600).

1 Introduction

This paper presents a simple but high-performance method for text categorization. The method assigns domain tags to words in an article, and categorizes the article as the most dominant domain. In this study, the 12 domains in Table 1 are used following (Hashimoto and Kurohashi, 2007) (H&K hereafter)[1]. Fundamental words are assigned with a domain tag by H&K's domain dictionary, while the domains of non-fundamental words (i.e. unknown words) are dynamically estimated, which makes the method different from previous ones. Another hallmark of the method is that it requires no machine learning. All you need is the domain dictionary and the access to the Web.

Table 1: Domains Assumed in H&K

CULTURE	LIVING	SCIENCE
RECREATION	DIET	BUSINESS
SPORTS	TRANSPORTATION	MEDIA
HEALTH	EDUCATION	GOVERNMENT

[1] In addition, NODOMAIN is prepared for words belonging to no particular domain like *blue* or *people*.

2 The Domain Dictionary

H&K constructed a domain dictionary, where about 30,000 Japanese fundamental content words (JFWs) are associated with appropriate domains. For example, *homer* is associated with SPORTS.

2.1 Construction Process

① Preparing Keywords for each Domain About 20 keywords for each domain were collected manually from words that appear frequently in the Web. They represent the contents of domains.

② Associating JFWs with Domains A JFW is associated with a domain of the highest A_d score. An A_d score of domain is calculated by summing up the top five A_k scores of the domain. Then, an A_k score, which is defined between a JFW and a keyword of a domain, is a measure that shows how strongly the JFW and the keyword are related. H&K adopt the χ^2 statistics to calculate an A_k score and use web pages as a corpus. The number of co-occurrences is approximated by the number of search engine hits when the two words are used as queries. A_k score between a JFW (jw) and a keyword (kw) is given as below.

$$A_k(jw, kw) = \frac{n(ad - bc)^2}{(a+b)(c+d)(a+c)(b+d)} \quad (1)$$

where n is the total number of Japanese web pages,

$$a = hits(jw \ \& \ kw), \quad b = hits(jw) - a,$$
$$c = hits(kw) - a, \quad d = n - (a + b + c).$$

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 69–72,
Columbus, Ohio, USA, June 2008.

Note that $hits(q)$ represents the number of search engine hits when q is used as a query.

③ **Manual Correction** Manual correction of the automatic association[2] is done to complete the dictionary. Since the accuracy of ② is 81.3%, manual correction is not time-consuming.

2.2 Distinctive Features

H&K's method is independent of what domains to assume. You can create your own dictionary. All you need is prepare keywords of your own domains. After that, the same construction process is applied.

Also note that H&K's method requires no text collection that is typically used for machine learning techniques. All you need is the access to the Web.

3 Blog Categorization

The categorization proceeds as follows: 1 Extract words from an article, 2 Assign domains and IDFs to the words, 3 Sum up IDFs for each domain, 4 Categorize the article as the domain of the highest IDF.[3] As for 2, the IDF is calculated as follows:[4]

$$\text{IDF}(w) = log\frac{\text{Total \# of Japanese web pages}}{\text{\# of hits of } w} \quad (2)$$

Fundamental words are assigned with their domains and IDFs by the domain dictionary, while those for unknown words are dynamically estimated by the method described in §4.

4 Domain Estimation of Unknown Words

The domain (and IDF) of unknown word is dynamically estimated exploiting the Web. More specifically, we use Wikipedia and Snippets of Web search, in addition to the domain dictionary. The estimation proceeds as follows (Figure 1): ❶ Search the Web with an unknown word, acquire the top 100 records, and calculate the IDF. ❷ Get the Wikipedia article about the word from the search result if any, estimate the domain of the word with the Wikipedia-strict module (§4.1), and exit. ❸ When no Wikipedia article about the word is found, then get any Wikipedia article in the top 30 of the search result if any, estimate the domain with the Wikipedia-loose module (§4.1), and exit. ❹ If no Wikipedia article is found in the top 30 of the search result, then remove all corporate snippets. ❺ Estimate the domain with the Snippets module (§4.2) if any snippet is left in the search result, and exit. ❻ If no snippet is left but the unknown word is a compound word containing fundamental words, then estimate the domain with the Components module (§4.3), and exit. ❼ If no snippet is left and the word does not contain fundamental words, then the estimation is a failure.

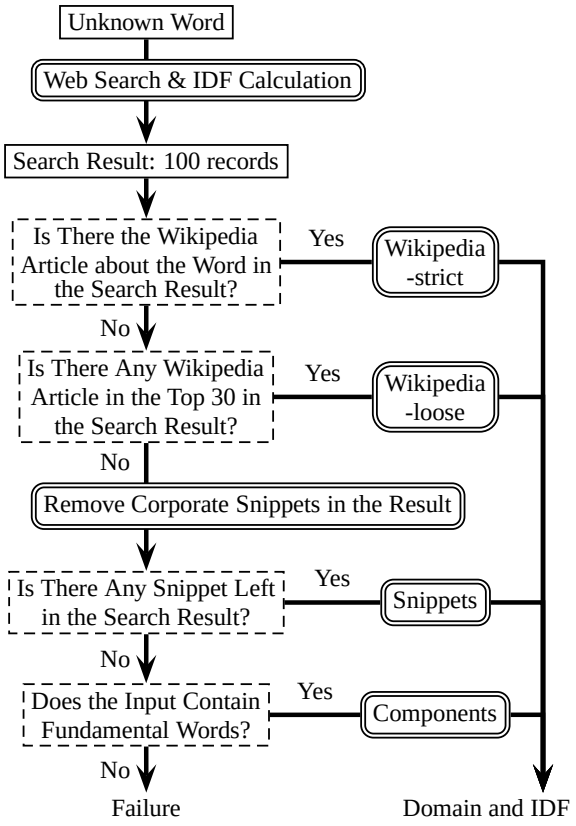


Figure 1: Domain Estimation Process

4.1 Wikipedia(-strict|-loose) Module

The two Wikipedia modules take the following procedure: 1 Extract only fundamental words from the Wikipedia article. 2 Assign domains and IDFs to the words using the domain dictionary. 3 Sum up IDFs for each domain. 4 Assign the domain of the highest IDF to the unknown word. If the domain is NODOMAIN, the second highest domain is chosen for the unknown word under the condition below:

[2] In H&K's method, reassociating JFWs with NODOMAIN is required before ③. We omit that due to the space limitation.

[3] If the domain of the highest IDF is NODOMAIN, the article is categorized as the second highest domain.

[4] We used 10,000,000,000 as the total number.

Second-highest-IDF/ NODOMAIN*'s-IDF*$>$0.15

4.2 Snippets Module

The Snippets module takes as input the snippets that are left in the search result after removing those of corporate web sites. We remove snippets in which corporate keywords like *sales* appear more than once. The keywords were collected from the analysis of our preliminary experiments. Removing corporate snippets is indispensable because they bias the estimation toward BUSINESS. This module is the same as the Wikipedia modules except that it extracts fundamental words from residual snippets.

4.3 Components Module

This is basically the same as the others except that it extracts fundamental words from the unknown word itself. For example, the domain of *finance market* is estimated from the domains of *finance* and *market*.

5 Evaluation

5.1 Experimental Condition

Data We categorized 600 Blog articles from Yahoo! Blog (`blogs.yahoo.co.jp`) into the 12 domains (50 articles for each domain). In Yahoo! Blog, articles are manually classified into Yahoo! Blog categories ($\simeq$ domains) by authors of the articles.

Evaluation Method We measured the accuracy of categorization and the domain estimation. In categorization, we tried three kinds of words to be extracted from articles: fundamental words (**F only** in Table 3), fundamental and simplex unknown words (i.e. no compound word) (**F+SU**), and fundamental and all unknown words (both simplex and compound, **F+AU**). Also, we measured the accuracy of N best outputs (**Top N**). During the categorization, about 12,000 unknown words were found in the 600 articles. Then, we sampled 500 estimation results from them. Table 2 shows the breakdown of the 500 unknown words in terms of their correct domains. The other 167 words belong to NODOMAIN.

5.2 Result of Blog Categorization

Table 3 shows the accuracy of categorization. The **F only** column indicates that a rather simple method like the one in §3 works well, if fundamental words are given good clues for categorization: the domain in our case. This is consistent with Kornai et al. (2003), who claim that only positive evidence matter in categorization. Also, **F+SU** slightly outperformed **F only**, and **F+AU** outperformed the others. This shows that the domain estimation of unknown words moderately improves Blog categorization.

Errors are mostly due to the system's incorrect focus on topics of secondary importance. For example, in an article on a sightseeing trip, which should be RECREATION, the author frequently mentions the means of transportation. As a result, the article was wrongly categorized as TRAFFIC.

Table 2: Breakdown of Unknown Words

CULT	42	LIVI	19	SCIE	38
RECR	15	DIET	19	BUSI	32
SPOR	27	TRAN	28	MEDI	23
HEAL	22	EDUC	24	GOVE	44

Table 3: Accuracy of Blog Categorization

Top N	F only	F+SU	F+AU
1.	0.89	0.91	0.94
2.	0.96	0.97	0.98
3.	0.98	0.98	0.99

5.3 Result of Domain Estimation

The accuracy of the domain estimation of unknown words was 77.2% (386/500). Table 4 shows the frequency in use and accuracy for each domain estimation module.[5] The Snippets module was used most frequently and achieved the reasonably good accuracy of 76%. Though the Wikipedia-strict module showed the best performance, it was used not

Table 4: Frequency and Accuracy for each Module

	Frequency	Accuracy
Wiki-s	0.146 (73/500)	0.85 (62/73)
Wiki-l	0.208 (104/500)	0.70 (73/104)
Snippt	0.614 (307/500)	0.76 (238/307)
Cmpnt	0.028 (14/500)	0.64 (9/14)
Failure	0.004 (2/500)	——

[5] Wiki-s, Wiki-l, Snippt and Cmpnt stand for Wikipedia-strict, Wikipedia-loose, Snippets and Components, respectively.

so often. However, we expect that as the number of Wikipedia articles increases, the best performing module will be used more frequently.

An example of newly coined words whose domains were estimated correctly is デイトレ, which is the abbreviation of デイトレード *day-trade*. It was correctly assigned with BUSINESS by the Wikipedia-loose module.

Errors were mostly due to the subtle boundary between NODOMAIN and the other particular domains. For instance, person's names that are common and popular should be NODOMAIN. But in most cases they were associated with some particular domain. This is due to the fact that virtually any person's name is linked to some particular domain in the Web.

6 Related Work

Previous text categorization methods like Joachims (1999) and Schapire and Singer (2000) are mostly based on machine learning. Those methods need huge quantities of training data, which is hard to obtain. Though there has been a growing interest in semi-supervised learning (Abney, 2007), it is in an early phase of development.

In contrast, our method requires no training data. All you need is a manageable amount of fundamental words with domains. Also note that our method is NOT tailored to the 12 domains. If you want your own domains to categorize, it is only necessary to construct your own dictionary, which is also domain-independent and not time-consuming.

In fact, there have been other proposals without the burden of preparing training data. Liu et al. (2004) prepare representative words for each class, by which they collect initial training data to build classifier. Ko and Seo (2004) automatically collect training data using a large amount of unlabeled data and a small amount of seed information. However, the novelty of this study is the on-the-fly estimation of unknown words' domains. This feature is very useful for categorizing Blog articles that are updated on a daily basis and filled with newly coined words.

Domain information has been used for many NLP tasks. Magnini et al. (2002) show the effectiveness of domain information for WSD. Piao et al. (2003) use domain tags to extract MWEs.

Previous domain resources include WordNet (Fellbaum, 1998) and HowNet (Dong and Dong, 2006), among others. H&K's dictionary is the first fully available domain resource for Japanese.

7 Conclusion

This paper presented a text categorization method that exploits H&K's domain dictionary and the dynamic domain estimation of unknown words. In the Blog categorization, the method achieved the accuracy of 94%, and the domain estimation of unknown words achieved the accuracy of 77%.

References

Steven Abney. 2007. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall.

Zhendong Dong and Qiang Dong. 2006. *HowNet and the Computation of Meaning*. World Scientific Pub Co Inc.

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

Chikara Hashimoto and Sadao Kurohashi. 2007. Construction of Domain Dictionary for Fundamental Vocabulary. In *ACL '07 Poster*, pages 137–140.

Thorsten Joachims. 1999. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209.

Youngjoong Ko and Jungyun Seo. 2004. Learning with Unlabeled Data for Text Categorization Using Bootstrapping and Feature Projection Techniques. In *ACL '04*, pages 255–262.

András Kornai, Marc Krellenstein, Michael Mulligan, David Twomey, Fruzsina Veress, and Alec Wysoker. 2003. Classifying the Hungarian web. In *EACL '03*, pages 203–210.

Bing Liu, Xiaoli Li, Wee Sun Lee, , and Philip Yu. 2004. Text Classification by Labeling Words. In *AAAI-2004*, pages 425–430.

Bernardo Magnini, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo. 2002. The Role of Domain Information in Word Sense Disambiguation. *Natural Language Engineering, special issue on Word Sense Disambiguation*, 8(3):359–373.

Scott S. L. Piao, Paul Rayson, Dawn Archer, Andrew Wilson, and Tony McEnery. 2003. Extracting multiword expressions with a semantic tagger. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, pages 49–56.

Robert E. Schapire and Yoram Singer. 2000. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168.

Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management

James Henderson
University of Geneva
Department of Computer Science
James.Henderson@cui.unige.ch

Oliver Lemon
University of Edinburgh
School of Informatics
olemon@inf.ed.ac.uk

Abstract

In spoken dialogue systems, Partially Observable Markov Decision Processes (POMDPs) provide a formal framework for making dialogue management decisions under uncertainty, but efficiency and interpretability considerations mean that most current statistical dialogue managers are only MDPs. These MDP systems encode uncertainty explicitly in a single state representation. We formalise such MDP states in terms of distributions over POMDP states, and propose a new dialogue system architecture (Mixture Model POMDPs) which uses mixtures of these distributions to efficiently represent uncertainty. We also provide initial evaluation results (with real users) for this architecture.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) provide a formal framework for making decisions under uncertainty. Recent research in spoken dialogue systems has used POMDPs for dialogue management (Williams and Young, 2007; Young et al., 2007). These systems represent the uncertainty about the dialogue history using a probability distribution over dialogue states, known as the POMDP's belief state, and they use approximate POMDP inference procedures to make dialogue management decisions. However, these inference procedures are too computationally intensive for most domains, and the system's behaviour can be difficult to predict. Instead, most current statistical dialogue managers use a single state to represent the dialogue history, thereby making them only Markov Decision Process models (MDPs). These state representations have been fine-tuned over many development cycles so that common types of uncertainty can be encoded in a single state. Examples of such representations include unspecified values, confidence scores, and confirmed/unconfirmed features. We formalise such MDP systems as compact encodings of POMDPs, where each MDP state represents a probability distribution over POMDP states. We call these distributions "MDP belief states".

Given this understanding of MDP dialogue managers, we propose a new POMDP spoken dialogue system architecture which uses mixtures of MDP belief states to encode uncertainty. A Mixture Model POMDP represents its belief state as a probability distribution over a finite set of MDP states. This extends the compact representations of uncertainty in MDP states to include arbitrary disjunction between MDP states. Efficiency is maintained because such arbitrary disjunction is not needed to encode the most common forms of uncertainty, and thus the number of MDP states in the set can be kept small without losing accuracy. On the other hand, allowing multiple MDP states provides the representational mechanism necessary to incorporate multiple speech recognition hypotheses into the belief state representation. In spoken dialogue systems, speech recognition is by far the most important source of uncertainty. By providing a mechanism to incorporate multiple arbitrary speech recognition hypotheses, the proposed architecture leverages the main advantage of POMDP systems while still maintaining the efficiency of MDP-based dialogue managers.

2 Mixture Model POMDPs

A POMDP belief state b_t is a probability distribution $P(s_t|V_{t-1}, u_t)$ over POMDP states s_t given the dia-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 73–76,
Columbus, Ohio, USA, June 2008.

logue history V_{t-1} and the most recent observation (i.e. user utterance) u_t. We formalise the meaning of an MDP state representation r_t as a distribution $b(r_t) = P(s_t|r_t)$ over POMDP states. We represent the belief state b_t as a list of pairs $\langle r_t^i, p_t^i \rangle$ such that $\sum_i p_t^i = 1$. This list is interpreted as a mixture of the $b(r_t^i)$.

$$b_t = \sum_i p_t^i b(r_t^i) \tag{1}$$

State transitions in MDPs are specified with an update function, $r_t = f(r_{t-1}, a_{t-1}, h_t)$, which maps the preceding state r_{t-1}, system action a_{t-1}, and user input h_t to a new state r_t. This function is intended to encode in r_t all the new information provided by a_{t-1} and h_t. The user input h_t is the result of automatic speech recognition (ASR) plus spoken language understanding (SLU) applied to u_t. Because there is no method for handling ambiguity in h_t, h_t is computed from the single best ASR-SLU hypothesis, plus some measure of ASR confidence.

In POMDPs, belief state transitions are done by changing the distribution over states to take into account the new information from the system action a_{t-1} and an n-best list of ASR-SLU hypotheses h_t^j. This new belief state can be estimated as

$$\begin{aligned} b_t &= P(s_t|V_{t-1}, u_t) \\ &= \sum_{h_t^j} \sum_{s_{t-1}} \frac{\begin{array}{l} P(s_{t-1}|V_{t-1}) P(h_t^j|V_{t-1}, s_{t-1}) \\ P(u_t|V_{t-1}, s_{t-1}, h_t^j) \\ P(s_t|V_{t-1}, s_{t-1}, h_t^j, u_t) \end{array}}{P(u_t|V_{t-1})} \\ &\approx \sum_{h_t^j} \sum_{s_{t-1}} \frac{\begin{array}{l} P(s_{t-1}|V_{t-2}, u_{t-1}) P(h_t^j|a_{t-1}, s_{t-1}) \\ P(h_t^j|u_t) P(s_t|a_{t-1}, s_{t-1}, h_t^j) \end{array}}{P(h_t^j) Z(V_t)} \end{aligned}$$

where $Z(V_t)$ is a normalising constant. $P(s_{t-1}|V_{t-2}, u_{t-1})$ is the previous belief state. $P(h_t^j|u_t)$ reflects the confidence of ASR-SLU in hypothesis h_t^j. $P(s_t|a_{t-1}, s_{t-1}, h_t^j)$ is normally 1 for $s_t = s_{t-1}$, but can be used to allow users to change their mind mid-dialogue. $P(h_t^j|a_{t-1}, s_{t-1})$ is a user model. $P(h_t^j)$ is a prior over ASR-SLU outputs.

Putting these two approaches together, we get the following update equation for our mixture of MDP belief states:

$$\begin{aligned} b_t &= P(s_t|V_{t-1}, u_t) \\ &\approx \sum_{h_t^j} \sum_{r_{t-1}^i} \frac{\begin{array}{l} p_{t-1}^i P(h_t^j|a_{t-1}, r_{t-1}^i) \\ P(h_t^j|u_t) b(f(r_{t-1}^i, a_{t-1}, h_t^j)) \end{array}}{P(h_t^j) Z(V_t)} \\ &= \sum_{i'} p_t^{i'} b(r_t^{i'}) \end{aligned} \tag{2}$$

where, for each i' there is one pair i, j such that

$$\begin{aligned} r_t^{i'} &= f(r_{t-1}^i, a_{t-1}, h_t^j) \\ p_t^{i'} &= \frac{p_{t-1}^i P(h_t^j|a_{t-1}, r_{t-1}^i) P(h_t^j|u_t)}{P(h_t^j) Z(V_t)}. \end{aligned} \tag{3}$$

For equation (2) to be true, we require that

$$b(f(r_{t-1}^i, a_{t-1}, h_t^j)) \approx P(s_t|a_{t-1}, r_{t-1}^i, h_t^j) \tag{4}$$

which simply ensures that the meaning assigned to MDP state representations and the MDP state transition function are compatible.

From equation (3), we see that the number of MDP states will grow exponentially with the length of the dialogue, proportionately to the number of ASR-SLU hypotheses. Some of the state-hypothesis pairs r_{t-1}^i, h_t^j may lead to equivalent states $f(r_{t-1}^i, a_{t-1}, h_t^j)$, but in general pruning is necessary. Pruning should be done so as to minimise the change to the belief state distribution, for example by minimising the KL divergence between the pre- and post- pruning belief states. We use two heuristic approximations to this optimisation problem. First, if two states share the same core features (e.g. filled slots, but not the history of user inputs), then the state with the lower probability is pruned, and its probability is added to the other state. Second, a fixed beam of the k most probable states is kept, and the other states are pruned. The probability p_t^i from a pruned state r_t^i is redistributed to unpruned states which are less informative than r_t^i in their core features.[1]

The interface between the ASR-SLU module and the dialogue manager is a set of hypotheses h_t^j paired with their confidence scores $P(h_t^j|u_t)$. These pairs are analogous to the state-probability pairs r_t^i, p_t^i within the dialogue manager, and we can extend our mixture model architecture to cover these pairs as well. Interpreting the set of $h_t^j, P(h_t^j|u_t)$ pairs as a

[1] In the current implementation, these pruned state probabilities are simply added to an uninformative "null" state, but in general we could check for logical subsumption between states.

mixture of distributions over more specific hypotheses becomes important when we consider pruning this set before passing it to the dialogue manager. As with the pruning of states, pruning should not simply remove a hypothesis and renormalise, it should redistribute the probability of a pruned hypothesis to similar hypotheses. This is not always computationally feasible, but all interfaces within the Mixture Model POMDP architecture are sets of hypothesis-probability pairs which can be interpreted as finite mixtures in some underlying hypothesis space.

Given an MDP state representation, this formalisation allows us to convert it into a Mixture Model POMDP. The only additional components of the model are the user model $P(h_t^j|a_{t-1}, r_{t-1}^i)$, the ASR-SLU prior $P(h_t^j)$, and the ASR-SLU confidence score $P(h_t^j|u_t)$. Note that there is no need to actually define $b(r_t^i)$, provided equation (4) holds.

3 Decision Making with MM POMDPs

Given this representation of the uncertainty in the current dialogue state, the spoken dialogue system needs to decide what system action to perform. There are several approaches to POMDP decision making which could be adapted to this representation, but to date we have only considered a method which allows us to directly derive a POMDP policy from the policy of the original MDP.

Here again we exploit the fact that the most frequent forms of uncertainty are already effectively handled in the MDP system (e.g. by filled vs. confirmed slot values). We propose that an effective dialogue management policy can be created by simply computing a mixture of the MDP policy applied to the MDP states in the belief state list. More precisely, we assume that the original MDP system specifies a Q function $Q_{\text{MDP}}(a_t, r_t)$ which estimates the expected future reward of performing action a_t in state r_t. We then estimate the expected future reward of performing action a_t in belief state b_t as the mixture of these MDP estimates.

$$Q(a_t, b_t) \approx \sum_i p_t^i Q_{\text{MDP}}(a_t, r_t^i) \qquad (5)$$

The dialogue management policy is to choose the action a_t with the largest value for $Q(a_t, b_t)$. This is known as a Q-MDP model (Littman et al., 1995), so we call this proposal a Mixture Model Q-MDP.

4 Related Work

Our representation of POMDP belief states using a set of distributions over POMDP states is similar to the approach in (Young et al., 2007), where POMDP belief states are represented using a set of partitions of POMDP states. For any set of partitions, the mixture model approach could express the same model by defining one MDP state per partition and giving it a uniform distribution inside its partition and zero probability outside. However, the mixture model approach is more flexible, because the distributions in the mixture do not have to be uniform within their non-zero region, and these regions do not have to be disjoint. A list of states was also used in (Higashinaka et al., 2003) to represent uncertainty, but no formal semantics was provided for this list, and therefore only heuristic uses were suggested for it.

5 Initial Experiments

We have implemented a Mixture Model POMDP architecture as a multi-state version of the DIPPER "Information State Update" dialogue manager (Bos et al., 2003). It uses equation (3) to compute belief state updates, given separate models for MDP state updates (for $f(r_{t-1}^i, a_{t-1}, h_t^j)$), statistical ASR-SLU (for $P(h_t^j|u_t)/P(h_t^j)$), and a statistical user model (for $P(h_t^j|a_{t-1}, r_{t-1}^i)$). The state list is pruned as described in section 2, where the "core features" are the filled information slot values and whether they have been confirmed. For example, the system will merge two states which agree that the user only wants a cheap hotel, even if they disagree on the sequence of dialogue acts which lead to this information. It also never prunes the "null" state, so that there is always some probability that the system knows nothing.

The system used in the experiments described below uses the MDP state representation and update function from (Lemon and Liu, 2007), which is designed for standard slot-filling dialogues. For the ASR model, it uses the HTK speech recogniser (Young et al., 2002) and an n-best list of three ASR hypotheses on each user turn. The prior over user inputs is assumed to be uniform. The ASR hypotheses are passed to the SLU model from (Meza-Ruiz et al., 2008), which produces a single user input for each ASR hypothesis. This SLU model was trained on

	TC %	Av. length (std. deviation)
Handcoded	56.0	7.2 (4.6)
MDP	66.6	7.2 (4.0)
MM Q-MDP	73.3	7.3 (3.7)

Table 1: Initial test results for human-machine dialogues, showing task completion and average length.

the TownInfo corpus of dialogues, which was collected using the TownInfo human-machine dialogue systems of (Lemon et al., 2006), transcribed, and hand annotated. ASR hypotheses which result in the same user input are merged (summing their probabilities), and the resulting list of at most three ASR-SLU hypotheses are passed to the dialogue manager. Thus the number of MDP states in the dialogue manager grows by up to three times at each step, before pruning. For the user model, the system uses an n-gram user model, as described in (Georgila et al., 2005), trained on the annotated TownInfo corpus.[2]

The system's dialogue management policy is a Mixture Model Q-MDP (MM Q-MDP) policy. As with the MDP states, the MDP Q function is from (Lemon and Liu, 2007). It was trained in an MDP system using reinforcement learning with simulated users (Lemon and Liu, 2007), and was not modified for use in our MM Q-MDP policy.

We tested this system with 10 different users, each attempting 9 tasks in the TownInfo domain (searching for hotels and restaurants in a fictitious town), resulting in 90 test dialogues. The users each attempted 3 tasks with the MDP system of (Lemon and Liu, 2007), 3 tasks with a state-of-the-art handcoded system (see (Lemon et al., 2006)), and 3 tasks with the MM Q-MDP system. Ordering of systems and tasks was controlled, and 3 of the users were not native speakers of English. We collected the Task Completion (TC), and dialogue length for each system, as reported in table 1. Task Completion is counted from the system logs when the user replies that they are happy with their chosen option. Such a small sample size means that these results are not statistically significant, but there is a clear trend showing the superiority of the the MM Q-MDP system, both in terms of more tasks being completed and less variability in overall dialogue length.

[2]Thanks to K. Georgilla for training this model.

6 Conclusions

Mixture Model POMDPs combine the efficiency of MDP spoken dialogue systems with the ability of POMDP models to make use of multiple ASR hypotheses. They can also be constructed from MDP models without additional training, using the Q-MDP approximation for the dialogue management policy. Initial results suggest that, despite its simplicity, this approach does lead to better spoken dialogue systems than MDP and hand-coded models.

Acknowledgments

This research received funding from UK EPSRC grant EP/E019501/1 and the European Community's FP7 under grant n° 216594 (CLASSIC project: www.classic-project.org).

References

J Bos, E Klein, O Lemon, and T Oka. 2003. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *Proc. SIGdial Workshop on Discourse and Dialogue*, Sapporo.

K Georgila, J Henderson, and O Lemon. 2005. Learning User Simulations for Information State Update Dialogue Systems. In *Proc. Eurospeech*.

H Higashinaka, M Nakano, and K Aikawa. 2003. Corpus-based discourse understanding in spoken dialogue systems. In *Proc. ACL*, Sapporo.

O Lemon and X Liu. 2007. Dialogue policy learning for combinations of noise and user simulation: transfer results. In *Proc. SIGdial*.

O Lemon, K Georgila, and J Henderson. 2006. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *Proc. ACL/IEEE SLT*.

ML Littman, AR Cassandra, and LP Kaelbling. 1995. Learning policies for partially observable environments: Scaling up. In *Proc. ICML*, pages 362–370.

I Meza-Ruiz, S Riedel, and O Lemon. 2008. Accurate statistical spoken language understanding from limited development resources. In *Proc. ICASSP*. (to appear).

JD Williams and SJ Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(2):231–422.

S Young, G Evermann, D Kershaw, G Moore, J Odell, D Ollason, D Povey, V Valtchev, and P Woodland. 2002. *The HTK Book*. Cambridge Univ. Eng. Dept.

SJ Young, J Schatzmann, K Weilhammer, and H Ye. 2007. The Hidden Information State Approach to Dialog Management. In *Proc. ICASSP*, Honolulu.

Recent Improvements in the CMU Large Scale Chinese-English SMT System

Almut Silja Hildebrand, Kay Rottmann, Mohamed Noamany, Qin Gao, Sanjika Hewavitharana, Nguyen Bach and **Stephan Vogel**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
silja, kayrm, mfn, qing, sanjika, nbach, vogel+@cs.cmu.edu

Abstract

In this paper we describe recent improvements to components and methods used in our statistical machine translation system for Chinese-English used in the January 2008 GALE evaluation. Main improvements are results of consistent data processing, larger statistical models and a POS-based word reordering approach.

1 Introduction

Building a full scale Statistical Machine Translation (SMT) system involves many preparation and training steps and it consists of several components, each of which contribute to the overall system performance. Between 2007 and 2008 our system improved by 5 points in BLEU from 26.60 to 31.85 for the unseen MT06 test set, which can be mainly attributed to two major points.

The fast growth of computing resources over the years make it possible to use larger and larger amounts of data in training. In Section 3 we show how parallelizing model training can reduce training time by an order of magnitude and how using larger training data as well as more extensive models improve translation quality.

Word reordering is still a difficult problem in SMT. In Section 4 we apply a Part Of Speech (POS) based syntactic reordering model successfully to our large Chinese system.

1.1 Decoder

Our translation system is based on the CMU SMT decoder as described in (Hewavitharana et al., 2005). Our decoder is a phrase-based beam search decoder, which combines multiple models e.g. phrase tables, several language models, a distortion model ect. in a log-linear fashion. In order to find an optimal set of weights, we use MER training as described in (Venugopal et al., 2005), which uses rescoring of the top n hypotheses to maximize an evaluation metric like BLEU or TER.

1.2 Evaluation

In this paper we report results using the BLEU metric (Papineni et al., 2002), however as the evaluation criterion in GALE is HTER (Snover et al., 2006), we also report in TER (Snover et al., 2005).

We used the test sets from the NIST MT evaluations from the years 2003 and 2006 as development and unseen test data.

1.3 Training Data

In translation model training we used the Chinese-English bilingual corpora relevant to GALE available through the LDC[1]. After sentence alignment these sources add up to 10.7 million sentences with 301 million running words on the English side. Our preprocessing steps include tokenization on the English side and for Chinese: automatic word segmentation using the revised version of the Stanford Chinese Word Segmenter[2] (Tseng et al., 2005) from 2007, replacement of traditional by simplified Chinese characters and 2-byte to 1-byte ASCII character normalization. After data cleaning steps like e.g. removal of sentence pairs with very unbalanced sen-

[1] http://projects.ldc.upenn.edu/gale/data/catalog.html
[2] http://nlp.stanford.edu/software/segmenter.shtml

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 77–80,
Columbus, Ohio, USA, June 2008.

tence length etc., we used the remaining 10 million sentences with 260 million words (English) in translation model training (260M system).

2 Number Tagging

Systematic tagging and pre-translation of numbers had shown significant improvements for our Arabic-English system, so we investigated this for Chinese-English. The baseline for these experiments was a smaller system with 67 million words (67M) bilingual training data (English) and a 500 million word 3-gram LM with a BLEU score of 27.61 on MT06. First we pre-translated all numbers in the testdata only, thus forcing the decoder to treat the numbers as unknown words. Probably because the system could not match longer phrases across the pre-translated numbers, the overall translation quality degraded by 1.6 BLEU to 26.05 (see Table 1).

We then tagged all numbers in the training corpus, replaced them with a placeholder tag and re-trained the translation model. This reduced the vocabulary and enabled the decoder to generalize longer phrases across numbers. This strategy did not lead to the expected result, the BLEU score for MT06 only reached 25.97 BLEU.

System	MT03	MT06
67M baseline	31.45/60.93	27.61/62.18
test data tagged	–	26.06/63.36
training data tagged	29.07/62.52	25.97/63.39

Table 1: Number tagging experiments, BLEU/TER

Analysing this in more detail, we found, the reason for this degradation in translation quality could be the unbalanced occurrence of number tags in the training data. From the bilingual sentence pairs, which contain number tags, 66.52% do not contain the same number of tags on the Chinese and the English side. As a consequence 52% of the phrase pairs in the phrase table, which contain number tags had to be removed, because the tags were unbalanced. This hurts system performance considerably.

3 Scaling up to Large Data

3.1 Language Model

Due to the availability of more computing resources, we were able to extend the language model history from 4- to 5-gram, which improved translation quality from 29.49 BLEU to 30.22 BLEU for our large scale 260M system (see Table 2). This shows, that longer LM histories help if we are able to use enough data in model training.

System	MT03	MT06
260M, 4gram	31.20/61.00	29.49/61.00
260M, 5gram	32.20/60.59	30.22/60.81

Table 2: 4- and 5-gram LM,260M system, BLEU/TER

The language model was trained on the sources from the English Gigaword Corpus V3, which contains several newspapers for the years between 1994 to 2006. We also included the English side of the bilingual training data, resulting in a total of 2.7 billion running words after tokenization.

We trained separate open vocabulary language models for each source and interpolated them using the SRI Language Modeling Toolkit (Stolcke, 2002). Table 3 shows the interpolation weights for the different sources. Apart from the English part of the bilingual data, the newswire data from the Chinese Xinhua News Agency and the Agence France Press have the largest weights. This reflects the makeup of the test data, which comes in large parts from these sources. Other sources, as for example the UN parlamentary speeches or the New York Times, differ significantly in style and vocabulary from the test data and therefore get small weights.

xin 0.30	cna 0.06	nyt 0.03
bil 0.26	un 0.07	ltw 0.01
afp 0.21	apw 0.05	

Table 3: LM interpolation weights per source

3.2 Speeding up Model Training

To accelerate the training of word alignment models we implemented a distributed version of GIZA++ (Och and Ney, 2003), based on the latest version of GIZA++ and a parallel version developed at Peking University (Lin et al., 2006). We divide the bilingual training data in equal parts and distribute it over several processing nodes, which perform alignment independently. In each iteration the nodes read the model from the previous step and output all necessary counts from the data for the models, e.g. the

co-occurrence or fertility model. A master process collects the counts from the nodes, normalizes them and outputs the intermediate model for each iteration.

This distributed GIZA++ version finished training the word alignment up to IBM Model 4 for both language directions on the full bilingual corpus (260 million words, English) in 39 hours. On average about 11 CPUs were running concurrently. In comparison the standard GIZA++ implementation finished the same training in 169 hours running on 2 CPUs, one for each language direction.

We used the Pharaoh/Moses package (Koehn et al., 2007) to extract and score phrase pairs using the grow-diag-final extraction method.

3.3 Translation Model

We trained two systems, one on the full data and one without the out-of-domain corpora: UN parlament, HK hansard and HK law parallel texts. These parlamentary sessions and law texts are very different in genre and style from the MT test data, which consists mainly of newspaper texts and in recent years also of weblogs, broadcast news and broadcast conversation. The in-domain training data had 3.8 million sentences and 67 million words (English). The 67 million word system reached a BLEU score of 29.65 on the unseeen MT06 testset. Even though the full 260M system was trained on almost four times as many running words, the baseline score for MT06 only increased by 0.6 to 30.22 BLEU (see Table 4).

System	MT03	MT06
67M in-domain	32.42/60.26	29.65/61.22
260M full	32.20/60.59	30.22/60.81

Table 4: In-domain only or all training data, BLEU/TER

The 67M system could not translate 752 Chinese words out of 38937, the number of unknown words decreased to 564 for the 260M system. To increase the unigram coverage of the phrase table, we added the lexicon entries that were not in the phrase table as one-word translations. This lowered the number of unknown words further to 410, but did not effect the translation score.

4 POS-based Reordering

As Chinese and English have very different word order, reordering over a rather limited distance during decoding is not sufficient. Also using a simple distance based distortion probability leaves it essentially to the language model to select among different reorderings. An alternative is to apply automatically learned reordering rules to the test sentences before decoding (Crego and Marino, 2006). We create a word lattice, which encodes many reorderings and allows long distance reordering. This keeps the translation process in the decoder monotone and makes it significantly faster compared to allowing long distance reordering at decoding time.

4.1 Learning Reordering Rules

We tag both language sides of the bilingual corpus with POS information using the Stanford Parser[3] and extract POS based reordering patterns from word alignment information. We use the context in which a reordering pattern is seen in the training data as an additional feature. Context refers to the words or tags to the left or to the right of the sequence for which a reordering pattern is extracted.

Relative frequencies are computed for every rule that has been seen more than n times in the training corpus (we observed good results for $n > 5$).

For the Chinese system we used only 350k bilingual sentence pairs to extract rules with length of up to 15. We did not reorder the training corpus to retrain the translation model on modified Chinese word order.

4.2 Applying Reordering Rules

To avoid hard decisions, we build a lattice structure for each source sentence as input for our decoder, which contains reordering alternatives consistent with the previously extracted rules.

Longer reordering patterns are applied first. Thereby shorter patterns can match along new paths, creating short distance reordering on top of long distance reordering. Every outgoing edge of a node is scored with the relative frequency of the pattern used on the following sub path (For details see (Rottmann and Vogel, 2007)). These model scores give this re-

[3] http://nlp.stanford.edu/software/lex-parser.shtml

ordering approach an advantage over a simple jump model with a sliding window.

System	MT03	MT06
260M, standard	32.20/60.59	30.22/60.81
260M, lattice	33.53/59.74	31.74/59.59

Table 5: Reordering lattice decoding in BLEU/TER

The system with reordering lattice input outperforms the system with a reordering window of 4 words by 1.5 BLEU (see Table 5).

5 Summary

The recent improvements to our Chinese-English SMT system (see Fig. 1) can be mainly attributed to a POS based word reordering method and the possibility to work with larger statistical models.

We used the lattice translation functionality of our decoder to translate reordering lattices. They are built using reordering rules extracted from tagged and aligned parallel data. There is further potential for improvement in this approach, as we did not yet reorder the training corpus and retrain the translation model on modified Chinese word order.

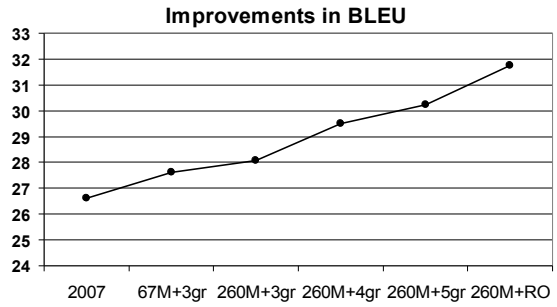


Figure 1: Improvements for MT06 in BLEU

We modified GIZA++ to run in parallel, which enabled us to include especially longer sentences into translation model training. We also extended our decoder to use 5-gram language models and were able to train an interpolated LM from all sources of the English GigaWord Corpus.

Acknowledgments

This work was partly funded by DARPA under the project GALE (Grant number #HR0011-06-2-0001).

References

Josep M. Crego and Jose B. Marino. 2006. Reordering Experiments for N-Gram-Based SMT. *Spoken Language Technology Workshop*, Palm Beach, Aruba.

Sanjika Hewavitharana, Bing Zhao, Almut Silja Hildebrand, Matthias Eck, Chiori Hori, Stephan Vogel and Alex Waibel. 2005. The CMU Statistical Machine Translation System for IWSLT 2005. *IWSLT 2005*, Pittsburgh, PA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. *ACL 2007, Demonstration Session*, Prague, Czech Republic.

Xiaojun Lin, Xinhao Wang, and Xihong Wu. 2006. NLMP System Description for the 2006 NIST MT Evaluation. *NIST 2006 MT Evaluation*.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Poukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *ACL 2002*, Philadelphia, USA.

Kay Rottmann and Stephan Vogel. 2007. Word Reordering in Statistical Machine Translation with a POS-based Distortion Model. *TMI-2007: 11th International Conference on Theoretical and Methodological Issues in MT,* Skvde, Sweden.

Mathew Snover, Bonnie Dorr, Richard Schwartz, John Makhoul, Linnea Micciula and Ralph Weischedel. 2005. A Study of Translation Error Rate with Targeted Human Annotation. *LAMP-TR-126*, University of Maryland, College Park and BBN Technologies.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. *7th Conference of AMTA*, Cambridge, Massachusetts, USA.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. *ICSLP*, Denver, Colorado.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky and Christopher Manning. 2005. A Conditional Random Field Word Segmenter. *Fourth SIGHAN Workshop on Chinese Language Processing*.

Ashish Venugopal, Andreas Zollman and Alex Waibel. 2005. Training and Evaluation Error Minimization Rules for Statistical Machine Translation. *ACL 2005, WPT-05, Ann Arbor, MI*

Machine Translation System Combination using ITG-based Alignments*

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, Markus Dreyer
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218
{damianos,eisner,khudanpur,dreyer}@jhu.edu

Abstract

Given several systems' automatic translations of the same sentence, we show how to combine them into a confusion network, whose various paths represent composite translations that could be considered in a subsequent rescoring step. We build our confusion networks using the method of Rosti et al. (2007), but, instead of forming alignments using the *tercom* script (Snover et al., 2006), we create alignments that minimize invWER (Leusch et al., 2003), a form of edit distance that permits properly nested block movements of substrings. Oracle experiments with Chinese newswire and weblog translations show that our confusion networks contain paths which are significantly better (in terms of BLEU and TER) than those in *tercom*-based confusion networks.

1 Introduction

Large improvements in machine translation (MT) may result from combining different approaches to MT with mutually complementary strengths. System-level combination of translation outputs is a promising path towards such improvements. Yet there are some significant hurdles in this path. One must somehow align the multiple outputs—to identify where different hypotheses reinforce each other and where they offer alternatives. One must then use this alignment to hypothesize a set of new, composite translations, and select the *best* composite hypothesis from this set. The alignment step is difficult because different MT approaches usually reorder the translated words differently. Training the selection step is difficult because identifying the best hypothesis (relative to a known reference translation) means scoring all the composite hypotheses, of which there may be exponentially many.

Most MT combination methods do create an exponentially large hypothesis set, representing it as a *confusion network* of strings in the target language (e.g., English). (A confusion network is a lattice where every node is on every path; i.e., each time step presents an *independent* choice among several phrases. Note that our contributions in this paper could be applied to arbitrary lattice topologies.) For example, Bangalore et al. (2001) show how to build a confusion network following a *multistring* alignment procedure of several MT outputs. The procedure (used primarily in biology, (Thompson et al., 1994)) yields monotone alignments that minimize the number of insertions, deletions, and substitutions. Unfortunately, monotone alignments are often poor, since machine translations (particularly from different models) can vary significantly in their word order. Thus, when Matusov et al. (2006) use this procedure, they deterministically reorder each translation prior to the monotone alignment.

The procedure described by Rosti et al. (2007) has been shown to yield significant improvements in translation quality, and uses an estimate of *Translation Error Rate* (TER) to guide the alignment. (TER is defined as the *minimum* number of inser-

*This work was partially supported by the DARPA GALE program (Contract No HR0011-06-2-0001). Also, we would like to thank the IBM Rosetta team for the availability of several MT system outputs.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 81–84,
Columbus, Ohio, USA, June 2008.

tions, deletions, substitutions and *block shifts* between two strings.) A remarkable feature of that procedure is that it performs the alignment of the output translations (i) without any knowledge of the translation model used to generate the translations, and (ii) without any knowledge of how the target words in each translation align back to the source words. In fact, it only requires a procedure for creating pairwise alignments of translations that allow appropriate re-orderings. For this, Rosti et al. (2007) use the *tercom* script (Snover et al., 2006), which uses a number of heuristics (as well as dynamic programming) for finding a sequence of edits (insertions, deletions, substitutions and block shifts) that convert an input string to another. In this paper, we show that one can build *better* confusion networks (in terms of the *best* translation possible from the confusion network) when the pairwise alignments are computed not by *tercom*, which *approximately* minimizes TER, but instead by an *exact* minimization of *invWER* (Leusch et al., 2003), which is a restricted version of TER that permits only properly nested sets of block shifts, and can be computed in polynomial time.

The paper is organized as follows: a summary of TER, *tercom*, and invWER, is presented in Section 2. The system combination procedure is summarized in Section 3, while experimental (oracle) results are presented in Section 4. Conclusions are given in Section 5.

2 Comparing *tercom* and invWER

The *tercom* script was created mainly in order to measure translation quality based on TER. As is proved by Shapira and Storer (2002), computation of TER is an NP-complete problem. For this reason, *tercom* uses some heuristics in order to compute *an approximation to TER* in polynomial time. In the rest of the paper, we will denote this approximation as *tercomTER*, to distinguish it from (the intractable) TER. The block shifts which are allowed in *tercom* have to adhere to the following constraints: (i) A block that has an exact match cannot be moved, and (ii) for a block to be moved, it should have an *exact* match in its new position. However, this sometimes leads to counter-intuitive sequences of edits; for instance, for the sentence pair

"thomas jefferson says eat your vegetables"
"eat your cereal thomas edison says",

tercom finds an edit sequence of cost 5, instead of the optimum 3. Furthermore, the block selection is done in a greedy manner, and the final outcome is dependent on the shift order, even when the above constraints are imposed.

An alternative to *tercom*, considered in this paper, is to use the Inversion Transduction Grammar (ITG) formalism (Wu, 1997) which allows one to view the problem of alignment as a problem of bilingual parsing. Specifically, ITGs can be used to find the optimal edit sequence under the restriction that block moves must be properly nested, like parentheses. That is, if an edit sequence swaps adjacent substrings A and B of the original string, then any other block move that affects A (or B) must stay completely within A (or B). An edit sequence with this restriction corresponds to a synchronous parse tree under a simple ITG that has one nonterminal and whose terminal symbols allow insertion, deletion, and substitution.

The minimum-cost ITG tree can be found by dynamic programming. This leads to *invWER* (Leusch et al., 2003), which is defined as the minimum number of edits (insertions, deletions, substitutions and block shifts allowed by the ITG) needed to convert one string to another. In this paper, the minimum-invWER alignments are used for generating confusion networks. The alignments are found with a 11-rule Dyna program (Dyna is an environment that facilitates the development of dynamic programs—see (Eisner et al., 2005) for more details). This program was further sped up (by about a factor of 2) with an A^* search heuristic computed by additional code. Specifically, our admissible outside heuristic for aligning two substrings estimated the cost of aligning the words *outside* those substrings as if reordering those words were free. This was complicated somewhat by type/token issues and by the fact that we were aligning (possibly weighted) lattices. Moreover, the *same* Dyna program was used for the computation of the minimum invWER path in these confusion networks (oracle path), without having to invoke *tercom* numerous times to compute the best sentence in an N-best list.

The two competing alignment procedures were

Lang. / Genre	tercomTER	invWER
Arabic NW	15.1%	14.9%
Arabic WB	26.0%	25.8%
Chinese NW	26.1%	25.6%
Chinese WB	30.9%	30.4%

Table 1: Comparison of average per-document tercomTER with invWER on the EVAL07 GALE Newswire ("NW") and Weblogs ("WB") data sets.

used to *estimate* the TER between machine translation system outputs and reference translations. Table 1 shows the TER estimates using *tercom* and invWER. These were computed on the translations submitted by a system to NIST for the GALE evaluation in June 2007. The references used are the post-edited translations for that system (i.e., these are "HTER" approximations). As can be seen from the table, in *all* language and genre conditions, invWER gives a *better* approximation to TER than tercomTER. In fact, out of the roughly 2000 total segments in all languages/genres, tercomTER gives a lower number of edits in only 8 cases! This is a clear indication that ITGs can explore the space of string permutations more effectively than *tercom*.

3 The System Combination Approach

ITG-based alignments and *tercom*-based alignments were also compared in oracle experiments involving confusion networks created through the algorithm of Rosti et al. (2007). The algorithm entails the following steps:

- Computation of all pairwise alignments between system hypotheses (either using ITGs or *tercom*); for each pair, one of the hypotheses plays the role of the "reference".
- Selection of a system output as the "skeleton" of the confusion network, whose words are used as anchors for aligning all other machine translation outputs together. Each arc has a translation output word as its label, with the special token "NULL" used to denote an insertion/deletion between the skeleton and another system output.
- Multiple consecutive words which are inserted relative to the skeleton form a phrase that gets aligned with an *epsilon* arc of the confusion network.
- Setting the weight of each arc equal to the negative log (posterior) probability of its label; this probability is proportional to the number of systems which output the word that gets aligned in that location. Note that the algorithm of Rosti et al. (2007) used N-best lists in the combination. Instead, we used the single-best output of each system; this was done because not all systems were providing N-best lists, and an unbalanced inclusion would favor some systems much more than others. Furthermore, for each genre, one of our MT systems was significantly better than the others in terms of word order, and it was chosen as the skeleton.

Genre	*CNs with tercom*	CNs with ITG
NW	50.1% (27.7%)	**48.8%** (**28.3%**)
WB	51.0% (25.5%)	**50.5%** (**26.0%**)

Table 2: TercomTERs of invWER-oracles and (in parentheses) oracle BLEU scores of confusion networks generated with *tercom* and ITG alignments. The best results per row are shown in bold.

4 Experimental Results

Table 2 shows tercomTERs of invWER-oracles (as computed by the aforementioned Dyna program) and oracle BLEU scores of the confusion networks. The confusion networks were generated using 9 MT systems applied to the Chinese GALE 2007 Dev set, which consists of roughly 550 Newswire segments, and 650 Weblog segments. The confusion networks which were generated with the ITG-based alignments gave significantly better oracle tercomTERs (significance tested with a Fisher sign test, $p - 0.02$) and better oracle BLEU scores. The BLEU oracle sentences were found using the dynamic-programming algorithm given in Dreyer et al. (2007) and measured using Philipp Koehn's evaluation script. On the other hand, a comparison between the 1-best paths did not reveal significant differences that would favor one approach or the other (either in terms of tercomTER or BLEU).

We also tried to understand which alignment method gives higher probability to paths "close" to the corresponding oracle. To do that, we computed the probability that a random path from a confusion network is within x edits from its oracle. This computation was done efficiently using finite-state-machine operations, and did not involve any randomization. Preliminary experiments with the invWER-oracles show that the probability of all paths which are within $x = 3$ edits from the oracle is roughly the same for ITG-based and tercom-based confusion networks. We plan to report our findings for a whole range of x-values in future work. Finally, a runtime comparison of the two techniques shows that ITGs are much more computationally intensive: on average, ITG-based alignments took 1.5 hours/sentence (owing to their $O(n^6)$ complexity), while *tercom*-based alignments only took 0.4 sec/sentence.

5 Concluding Remarks

We compared alignments obtained using the widely used program *tercom* with alignments obtained with ITGs and we established that the ITG alignments are superior in two ways. Specifically: (a) we showed that invWER (computed using the ITG alignments) gives a better approximation to TER between machine translation outputs and human references than *tercom*; and (b) in an oracle system combination experiment, we found that confusion networks generated with ITG alignments contain better oracles, both in terms of tercomTER and in terms of BLEU.

Future work will include rescoring results with a language model, as well as exploration of heuristics (e.g., allowing only "short" block moves) that can reduce the ITG alignment complexity to $O(n^4)$.

References

S. Bangalore, G. Bordel, and G. Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Proceedings of ASRU*, pages 351–354.

M. Dreyer, K. Hall, and S. Khudanpur. 2007. Comparing reordering constraints for smt using efficient bleu oracle computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 103–110, Rochester, New York, April. Association for Computational Linguistics.

Jason Eisner, Eric Goldlust, and Noah A. Smith. 2005. Compiling comp ling: Weighted dynamic programming and the Dyna language. In *Proceedings of HLT-EMNLP*, pages 281–290. Association for Computational Linguistics, October.

G. Leusch, N. Ueffing, and H. Ney. 2003. A novel string-to-string distance measure with applications to machine translation evaluation. In *Proceedings of the Machine Translation Summit 2003*, pages 240–247, September.

E. Matusov, N. Ueffing, and H. Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proceedings of EACL*, pages 33–40.

A.-V.I. Rosti, S. Matsoukas, and R. Schwartz. 2007. Improved word-level system combination for machine translation. In *Proceedings of the ACL*, pages 312–319, June.

D. Shapira and J. A. Storer. 2002. Edit distance with move operations. In *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching*, volume 2373/2002, pages 85–98, Fukuoka, Japan, July.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, Cambridge, MA, August.

J. D. Thompson, D. G. Higgins, and T. J. Gibson. 1994. Clustalw: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680.

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

Dictionary Definitions based Homograph Identification using a Generative Hierarchical Model

Anagha Kulkarni **Jamie Callan**
Language Technologies Institute
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213, USA
{anaghak, callan}@cs.cmu.edu

Abstract

A solution to the problem of *homograph* (words with multiple distinct meanings) identification is proposed and evaluated in this paper. It is demonstrated that a mixture model based framework is better suited for this task than the standard classification algorithms – relative improvement of 7% in F1 measure and 14% in Cohen's kappa score is observed.

1 Introduction

Lexical ambiguity resolution is an important research problem for the fields of information retrieval and machine translation (Sanderson, 2000; Chan et al., 2007). However, making fine-grained sense distinctions for words with multiple closely-related meanings is a subjective task (Jorgenson, 1990; Palmer et al., 2005), which makes it difficult and error-prone. Fine-grained sense distinctions aren't necessary for many tasks, thus a possibly-simpler alternative is lexical disambiguation at the level of homographs (Ide and Wilks, 2006). *Homographs* are a special case of semantically ambiguous words: Words that can convey multiple *distinct* meanings. For example, the word *bark* can imply two very different concepts – 'outer layer of a tree trunk', or, 'the sound made by a dog' and thus is a homograph. Ironically, the definition of the word 'homograph' is itself ambiguous and much debated; however, in this paper we consistently use the above definition.

If the goal is to do word-sense disambiguation of homographs in a very large corpus, a manually-generated homograph inventory may be impractical. In this case, the first step is to determine which words in a lexicon are homographs. This problem is the subject of this paper.

2 Finding the Homographs in a Lexicon

Our goal is to identify the homographs in a large lexicon. We assume that manual labor is a scarce resource, but that online dictionaries are plentiful (as is the case on the web). Given a word from the lexicon, definitions are obtained from eight dictionaries: Cambridge Advanced Learners Dictionary (CALD), Compact Oxford English Dictionary, MSN Encarta, Longman Dictionary of Contemporary English (LDOCE), The Online Plain Text English Dictionary, Wiktionary, WordNet and Wordsmyth. Using multiple dictionaries provides more evidence for the inferences to be made and also minimizes the risk of missing meanings because a particular dictionary did not include one or more meanings of a word (a surprisingly common situation). We can now rephrase the problem definition as that of determining which words in the lexicon are homographs given a set of dictionary definitions for each of the words.

2.1 Features

We use nine meta-features in our algorithm. Instead of directly using common lexical features such as n-grams we use meta-features which are functions defined on the lexical features. This ab-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 85–88,
Columbus, Ohio, USA, June 2008.

straction is essential in this setup for the generality of the approach. For each word *w* to be classified each of the following meta-features are computed.

1. **Cohesiveness Score:** Mean of the cosine similarities between each pair of definitions of *w*.
2. **Average Number of Definitions:** The average number of definitions per dictionary.
3. **Average Definition Length:** The average length (in words) of definitions of *w*.
4. **Average Number of Null Similarities:** The number of definition pairs that have zero cosine similarity score (no word overlap).
5. **Number of Tokens:** The sum of the lengths (in words) of the definitions of *w*.
6. **Number of Types:** The size of the vocabulary used by the set of definitions of *w*.
7. **Number of Definition Pairs with n Word Overlaps:** The number of definition pairs that have more than n=2 words in common.
8. **Number of Definition Pairs with m Word Overlaps:** The number of definition pairs that have more than m=4 words in common.
9. **Post Pruning Maximum Similarity:** (below)

The last feature sorts the pair-wise cosine similarity scores in ascending order, prunes the top n% of the scores, and uses the maximum remaining score as the feature value. This feature is less ad-hoc than it may seem. The set of definitions is formed from eight dictionaries, so almost identical definitions are a frequent phenomenon, which makes the maximum cosine similarity a useless feature. A pruned maximum turns out to be useful information. In this work n=15 was found to be most informative using a tuning dataset.

Each of the above features provides some amount of discriminative power to the algorithm. For example, we hypothesized that on average the cohesiveness score will be lower for homographs than for non-homographs. Figure 1 provides an illustration. If empirical support was observed for such a hypothesis about a candidate feature then the feature was selected. This empirical evidence was derived from only the training portion of the data (Section 3.1).

The above features are computed on definitions stemmed with the Porter Stemmer. Closed class words, such as articles and prepositions, and dictionary-specific stopwords, such as 'transitive', 'intransitive', and 'countable', were also removed.

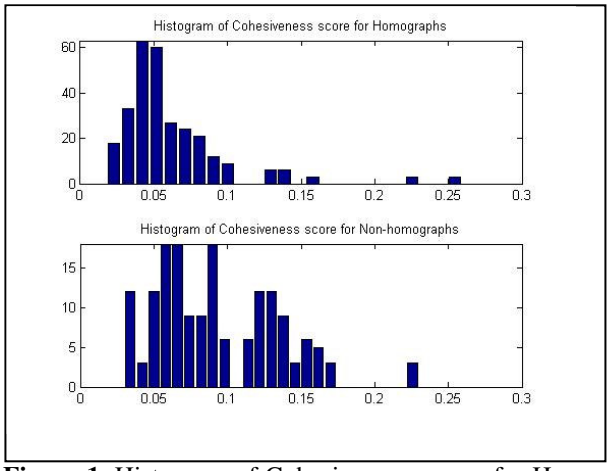


Figure 1. Histogram of Cohesiveness scores for Homographs and Non-homographs.

2.2 Models

We formulate the homograph detection process as a generative hierarchical model. Figure 2 provides the plate notation of the graphical model. The latent (unobserved) variable *Z* models the class information: homograph or non-homograph. Node ***X*** is the conditioned random vector (*Z* is the conditioning variable) that models the feature vector.

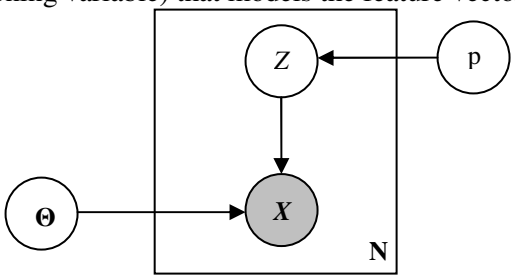


Figure 2. Plate notation for the proposed model.

This setup results in a mixture model with two components, one for each class. The *Z* is assumed to be Bernoulli distributed and thus parameterized by a single parameter *p*. We experiment with two continuous multivariate distributions, Dirichlet and Multivariate Normal (MVN), for the conditional distribution of ***X***|*Z*.

$$Z \sim \text{Bernoulli } (p)$$
$$\boldsymbol{X}|Z \sim \text{Dirichlet } (\mathbf{a_z})$$
OR
$$\boldsymbol{X}|Z \sim \text{MVN } (\mathbf{mu_z}, \mathbf{cov_z})$$

We will refer to the parameters of the conditional distribution as $\boldsymbol{\Theta}_z$. For the Dirichlet distribution, $\boldsymbol{\Theta}_\mathbf{z}$ is a ten-dimensional vector $\mathbf{a_z} = (a_{z1}, .., a_{z10})$. For the MVN, $\boldsymbol{\Theta}_\mathbf{z}$ represents a nine-dimensional mean vector $\mathbf{mu_z} = (mu_{z1}, .., mu_{z9})$

and a nine-by-nine-dimensional covariance matrix $\mathbf{cov_z}$. We use maximum likelihood estimators (MLE) for estimating the parameters (p, $\mathbf{\Theta_z}$). The MLEs for Bernoulli and MVN parameters have analytical solutions. Dirichlet parameters were estimated using an estimation method proposed and implemented by Tom Minka[1].

We experiment with three model setups: Supervised, semi-supervised, and unsupervised. In the supervised setup we use the training data described in Section 3.1 for parameter estimation and then use thus fitted models to classify the tuning and test dataset. We refer to this as the Model I. In Model II, the semi-supervised setup, the training data is used to initialize the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) and the unlabeled data, described in Section 3.1, updates the initial estimates. The Viterbi (hard) EM algorithm was used in these experiments. The E-step was modified to include only those unlabeled data-points for which the posterior probability was above certain threshold. As a result, the M-step operates only on these high posterior data-points. The optimal threshold value was selected using a tuning set (Section 3.1). The unsupervised setup, Model III, is similar to the semi-supervised setup except that the EM algorithm is initialized using an informed guess by the authors.

3 Data

In this study, we concentrate on recognizing homographic nouns, because homographic ambiguity is much more common in nouns than in verbs, adverbs or adjectives.

3.1 Gold Standard Data

A set of potentially-homographic nouns was identified by selecting all words with at least two noun definitions in both CALD and LDOCE. This set contained 3,348 words.

225 words were selected for manual annotation as homograph or non-homograph by random sampling of words that were on the above list and used in prior psycholinguistic studies of homographs (Twilley et al., 1994; Azuma, 1996) or on the Academic Word List (Coxhead, 2000).

Four annotators at, the Qualitative Data Analysis Program at the University of Pittsburgh, were trained to identify homographs using sets of dictionary definitions. After training, each of the 225 words was annotated by each annotator. On average, annotators categorized each word in just 19 seconds. The inter-annotator agreement was 0.68, measured by Fleiss' Kappa.

23 words on which annotators disagreed (2/2 vote) were discarded, leaving a set of 202 words (the "gold standard") on which at least 3 of the 4 annotators agreed. The best agreement between the gold standard and a human annotator was 0.87 kappa, and the worst was 0.78. The class distribution (homographs and non-homographs) was 0.63, 0.37. The set of 3,123 words that were not annotated was the unlabeled data for the EM algorithm.

4 Experiments and Results

A stratified division of the gold standard data in the proportion of 0.75 and 0.25 was done in the first step. The smaller portion of this division was held out as the testing dataset. The bigger portion was further divided into two portions of 0.75 and 0.25 for the training set and the tuning set, respectively. The best and the worst kappa between a human annotator and the test set are 0.92 and 0.78.

Each of the three models described in Section 2.2 were experimented with both Dirichlet and MVN as the conditional. An additional experiment using two standard classification algorithms – Kernel Based Naïve Bayes (NB) and Support Vector Machines (SVM) was performed. We refer to this as the baseline experiment. The Naïve Bayes classifier outperformed SVM on the tuning as well as the test set and thus we report NB results only. A four-fold cross-validation was employed for the all the experiments on the tuning set. The results are summarized in Table 1. The reported precision, recall and F1 values are for the homograph class.

The naïve assumption of class conditional feature independence is common to simple Naïve Bayes classifier, a kernel based NB classifier; however, unlike simple NB it is capable of modeling non-Gaussian distributions. Note that in spite of this advantage the kernel based NB is outperformed by the MVN based hierarchical model. Our nine features are by definition correlated and thus it was our hypothesis that a multivariate distribution such as MVN which can capture the covariance amongst the features will be a better fit. The above finding confirms this hypothesis.

[1] http://research.microsoft.com/~minka/software/fastfit/

	Tuning Set				Test Set			
	Precision	**Recall**	**F1**	**Kappa**	**Precision**	**Recall**	**F1**	**Kappa**
Model I – Dirichlet	0.84	0.74	**0.78**	**0.47**	0.81	0.62	0.70	0.34
Model II – Dirichlet	0.85	0.71	0.77	0.45	0.81	0.60	0.68	0.33
Model III – Dirichlet	0.78	0.74	0.76	0.37	0.82	0.56	0.67	0.32
Model I – MVN	0.70	0.75	0.78	0.32	0.80	0.73	**0.76**	**0.41**
Model II – MVN	0.74	0.82	0.78	0.34	0.71	0.79	0.74	0.25
Model III – MVN	0.69	0.89	0.77	0.22	0.64	0.84	0.72	0.22
Baseline – NB	0.82	0.73	**0.77**	**0.43**	0.82	0.63	**0.71**	**0.36**

Table 1. Results for the six models and the baseline on the tuning and test set.

One of the known situations when mixture models out-perform standard classification algorithms is when the data comes from highly overlapping distributions. In such cases the classification algorithms that try to place the decision boundary in a sparse area are prone to higher error-rates than mixture model based approach. We believe that this is explanations of the observed results. On the test set a relative improvement of 7% in F1 and 14% in kappa statistic is obtained using the MVN mixture model.

The results for the semi-supervised models are non-conclusive. Our post-experimental analysis reveals that the parameter updation process using the unlabeled data has an effect of overly separating the two overlapping distributions. This is triggered by our threshold based EM methodology which includes only those data-points for which the model is highly confident; however such data-points are invariable from the non-overlapping regions of the distribution, which gives a false view to the learner that the distributions are less overlapping. We believe that the unsupervised models also suffer from the above problem in addition to the possibility of poor initializations.

5 Conclusions

We have demonstrated in this paper that the problem of homograph identification can be approached using dictionary definitions as the source of information about the word. Further more, using multiple dictionaries provides more evidence for the inferences to be made and also minimizes the risk of missing few meanings of the word.

We can conclude that by modeling the underlying data generation process as a mixture model, the problem of homograph identification can be performed with reasonable accuracy.

The capability of identifying homographs from non-homographs enables us to take on the next steps of sense-inventory generation and lexical ambiguity resolution.

Acknowledgments

We thank Shay Cohen and Dr. Matthew Harrison for the helpful discussions. This work was supported in part by the Pittsburgh Science of Learning Center which is funded by the National Science Foundation, award number SBE-0354420.

References

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B, 39(1):1–38.

A. Coxhead. 2000. A New Academic Word List. TESOL, Quarterly, 34(2): 213-238.

J. Jorgenson. 1990. The psychological reality of word senses. Journal of Psycholinguistic Research 19:167-190.

L. Twilley, P. Dixon, D. Taylor, and K. Clark. 1994. University of Alberta norms of relative meaning frequency for 566 homographs. Memory and Cognition. 22(1): 111-126.

M. Sanderson. 2000. Retrieving with good sense. Information Retrieval, 2(1): 49-69.

M. Palmer, H. Dang, C. Fellbaum, 2005. Making fine-grained and coarse-grained sense distinctions. Journal of Natural Language Engineering. 13: 137-163.

N. Ide and Y. Wilks. 2006. Word Sense Disambiguation, Algorithms and Applications. Springer, Dordrecht, The Netherlands.

T. Azuma. 1996. Familiarity and Relatedness of Word Meanings: Ratings for 110 Homographs. Behavior Research Methods, Instruments and Computers. 28(1): 109-124.

Y. Chan, H. Ng, and D. Chiang. 2007. Proceeding of Association for Computational Linguistics, Prague, Czech Republic.

A Novel Feature-based Approach to Chinese Entity Relation Extraction

Wenjie Li[1], Peng Zhang[1,2], Furu Wei[1], Yuexian Hou[2] and Qin Lu[1]
[1]Department of Computing
The Hong Kong Polytechnic University, Hong Kong
{cswjli,csfwei,csluqin}@comp.polyu.edu.hk
[2]School of Computer Science and Technology
Tianjin University, China
{pzhang,yxhou}@tju.edu.cn

Abstract

Relation extraction is the task of finding semantic relations between two entities from text. In this paper, we propose a novel feature-based Chinese relation extraction approach that explicitly defines and explores nine positional structures between two entities. We also suggest some correction and inference mechanisms based on relation hierarchy and co-reference information etc. The approach is effective when evaluated on the ACE 2005 Chinese data set.

1 Introduction

Relation extraction is promoted by the ACE program. It is the task of finding predefined semantic relations between two entities from text. For example, the sentence "Bill Gates is the chairman and chief software architect of Microsoft Corporation" conveys the ACE-style relation "ORG-AFFILIATION" between the two entities "Bill Gates (PER)" and "Microsoft Corporation (ORG)".

The task of relation extraction has been extensively studied in English over the past years. It is typically cast as a classification problem. Existing approaches include feature-based and kernel-based classification. Feature-based approaches transform the context of two entities into a liner vector of carefully selected linguistic features, varying from entity semantic information to lexical and syntactic features of the context. Kernel-based approaches, on the other hand, explore structured representation such as parse tree and dependency tree and directly compute the similarity between trees. Comparably, feature-based approaches are easier to implement and achieve much success.

In contrast to the significant achievements concerning English and other Western languages, research progress in Chinese relation extraction is quite limited. This may be attributed to the different characteristic of Chinese language, e.g. no word boundaries and lack of morphologic variations, etc. In this paper, we propose a character-based Chinese entity relation extraction approach that complements entity context (both internal and external) character *N*-grams with four word lists extracted from a published Chinese dictionary. In addition to entity semantic information, we define and examine nine positional structures between two entities. To cope with the data sparseness problem, we also suggest some correction and inference mechanisms according to the given ACE relation hierarchy and co-reference information. Experiments on the ACE 2005 data set show that the positional structure feature can provide stronger support for Chinese relation extraction. Meanwhile, it can be captured with less effort than applying deep natural language processing. But unfortunately, entity co-reference does not help as much as we have expected. The lack of necessary co-referenced mentions might be the main reason.

2 Related Work

Many approaches have been proposed in the literature of relation extraction. Among them, feature-based and kernel-based approaches are most popular.

Kernel-based approaches exploit the structure of the tree that connects two entities. Zelenko et al (2003) proposed a kernel over two parse trees, which recursively matched nodes from roots to leaves in a top-down manner. Culotta and Sorensen (2004) extended this work to estimate similarity between augmented dependency trees. The above two work was further advanced by Bunescu and Mooney (2005) who argued that the information to extract a relation between two entities can be typically captured by the shortest path between them in the dependency graph. Later, Zhang et al (2006) developed a composite kernel that combined parse tree kernel with entity kernel and Zhou et al (2007) experimented with a context-sensitive kernel by automatically determining context-sensitive tree spans.

In the feature-based framework, Kambhatla (2004) employed ME models to combine diverse lexical, syntactic and semantic features derived from word, entity type, mention level, overlap, dependency and parse tree. Based on his work, Zhou et al (2005)

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 89–92,
Columbus, Ohio, USA, June 2008.

further incorporated the base phrase chunking information and semi-automatically collected country name list and personal relative trigger word list. Jiang and Zhai (2007) then systematically explored a large space of features and evaluated the effectiveness of different feature subspaces corresponding to sequence, syntactic parse tree and dependency parse tree. Their experiments showed that using only the basic unit features within each feature subspace can already achieve state-of-art performance, while over-inclusion of complex features might hurt the performance.

Previous approaches mainly focused on English relations. Most of them were evaluated on the ACE 2004 data set (or a sub set of it) which defined 7 relation types and 23 subtypes. Although Chinese processing is of the same importance as English and other Western language processing, unfortunately few work has been published on Chinese relation extraction. Che et al (2005) defined an improved edit distance kernel over the original Chinese string representation around particular entities. The only relation they studied is PERSON-AFFLIATION. The insufficient study in Chinese relation extraction drives us to investigate how to find an approach that is particularly appropriate for Chinese.

3 A Chinese Relation Extraction Model

Due to the aforementioned reasons, entity relation extraction in Chinese is more challenging than in English. The system segmented words are already not error free, saying nothing of the quality of the generated parse trees. All these errors will undoubtedly propagate to the subsequent processing, such as relation extraction. It is therefore reasonable to conclude that kernel-based especially tree-kernel approaches are not suitable for Chinese, at least at current stage. In this paper, we study a feature-based approach that basically integrates entity related information with context information.

3.1 Classification Features

The classification is based on the following four types of features.

- **Entity Positional Structure Features**

We define and examine nine finer positional structures between two entities (see Appendix). They can be merged into three coarser structures.

- **Entity Features**

Entity types and subtypes are concerned.

- **Entity Context Features**

These are character-based features. We consider both internal and external context. Internal context includes the characters inside two entities and the characters inside the heads of two entities. External context involves the characters around two entities within a given window size (it is set to 4 in this study). All the internal and external context characters are transformed to Uni-grams and Bi-grams.

- **Word List Features**

Although Uni-grams and Bi-grams should be able to cover most of Chinese words given sufficient training data, many discriminative words might not be discovered by classifiers due to the severe sparseness problem of Bi-grams. We complement character-based context features with four word lists which are extracted from a published Chinese dictionary. The word lists include 165 prepositions, 105 orientations, 20 auxiliaries and 25 conjunctions.

3.2 Correction with Relation/Argument Constraints and Type/Subtype Consistency Check

An identified relation is said to be correct only when its type/subtype (R) is correct and at the same time its two arguments (ARG-1 and ARG-2) must be of the correct entity types/subtypes and of the correct order. One way to improve the previous feature-based classification approach is to make use of the prior knowledge of the task to find and rectify the incorrect results. Table 1 illustrates the examples of possible relations between PER and ORG. We regard possible relations between two particular types of entity arguments as constraints. Some relations are symmetrical for two arguments, such as PER_SOCIAL.FAMILY, but others not, such as ORG_AFF.EMPLOYMENT. Argument orders are important for asymmetrical relations.

	PER	ORG
PER	PER_SOCIAL.BUS, PER_SOCIAL.FAMILY, …	ORG_AFF.EMPLOYMENT, ORG_AFF.OWNERSHIP, …
ORG		PART_WHOLE.SUBSIDIARY, ORG_AFF.INVESTOR/SHARE, …

Table 1 Possible Relations between ARG-1 and ARG-2

Since our classifiers are trained on relations instead of arguments, we simply select the first (as in adjacent and separate structures) and outer (as in nested structures) as the first argument. This setting works at most of cases, but still fails sometimes. The correction works in this way. Given two entities, if the identified type/subtype is an impossible one, it is revised to NONE (it means no relation at all). If the identified type/subtype is possible, but the order of arguments does not consist with the given relation definition, the order of arguments is adjusted.

Another source of incorrect results is the inconsistency between the identified types and subtypes, since they are typically classified separately.

This type of errors can be checked against the provided hierarchy of relations, such as the subtypes OWNERSHIP and EMPLOYMENT must belong to the ORG_AFF type. There are existing strategies to deal with this problem, such as strictly bottom-up (i.e. use the identified subtype to choose the type it belongs to), guiding top-down (i.e. to classify types first and then subtypes under a certain type). However, these two strategies lack of interaction between the two classification levels. To insure consistency in an interactive manner, we rank the first *n* numbers of the most likely classified types and then check them against the classified subtype one by one until the subtype conforms to a type. The matched type is selected as the result. If the last type still fails, both type and subtype are revised to NONE. We call this strategy type selection. Alternatively, we can choose the most likely classified subtypes, and check them with the classified type (i.e. subtype selection strategy). Currently, *n* is 2.

3.2 Inference with Co-reference Information and Linguistic Patterns

Each entity can be mentioned in different places in text. Two mentions are said to be co-referenced to one entity if they refers to the same entity in the world though they may have different surface expressions. For example, both "he" and "Gates" may refer to "Bill Gates of Microsoft". If a relation "ORG-AFFILIATION" is held between "Bill Gates" and "Microsoft", it must be also held between "he" and "Microsoft". Formally, given two entities E1={EM_{11}, EM_{12}, …, EM_{1n}} and E2={EM_{21}, EM_{22}, …, EM_{2m}} (E_i is an entity, EM_{ij} is a mention of E_i), it is true that $R(EM_{11}, EM_{21}) \Rightarrow R(EM_{1l}, EM_{2k})$. This nature allows us to infer more relations which may not be identified by classifiers.

Our previous experiments show that the performance of the nested and the adjacent relations is much better than the performance of other structured relations which suffer from unbearable low recall due to insufficient training data. Intuitively we can follow the path of "Nested ⇒ Adjacent ⇒ Separated ⇒ Others" (Nested, Adjacent and Separated structures are majority in the corpus) to perform the inference. But soon we have an interesting finding. If two related entities are nested, almost all the mentions of them are nested. So basically inference works on "Adjacent ⇒ Separated".

When considering the co-reference information, we may find another type of inconsistency, i.e. the one raised from co-referenced entity mentions. It is possible that $R(EM_{11}, EM_{21}) \neq R(EM_{12}, EM_{22})$ when R is identified based on the context of EM. Co-reference not only helps for inference but also provides the second chance to check the consistency among entity mention pairs so that we can revise accordingly. As the classification results of SVM can be transformed to probabilities with a sigmoid function, the relations of lower probability mention pairs are revised according to the relation of highest probability mention pairs.

The above inference strategy is called coreference-based inference. Besides, we find that pattern-based inference is also necessary. The relations of adjacent structure can infer the relations of separated structure if there are certain linguistic indicators in the local context. For example, given a local context "EM_1 and EM_2 located EM_3", if the relation of EM_2 and EM_3 has been identified, EM_1 and EM_3 will take the relation type/subtype that EM_2 and EM_3 holds. Currently, the only indicators under consideration are "and" and "or". However, more patterns can be included in the future.

4 Experimental Results

The experiments are conducted on the ACE 2005 Chinese RDC training data (with true entities) where 6 types and 18 subtypes of relations are annotated. We use 75% of it to train SVM classifiers and the remaining to evaluate results.

The aim of the first set of experiments is to examine the role of structure features. In these experiments, a "NONE" class is added to indicate a null type/subtype. With entity features and entity context features and word list features, we consider three different classification contexts: (1), only three coarser structures[1], i.e. nested, adjacent and separated, are used as feature, and a classifier is trained for each relation type and subtype; (2) similar to (1) but all nine structures are concerned; and (3) similar to (2) but the training data is divided into 9 parts according to structure, i.e. type and subtype classifiers are trained on the data with the same structures. The results presented in Table 2 show that 9-structure is much more discriminative than 3-structure. Also, the performance can be improved significantly by dividing training data based on nine structures.

Type / Subtype	Precision	Recall	F-measure
3-Structure	0.7918/0.7356	0.3123/0.2923	0.4479/0.4183
9-Structure	0.7533/0.7502	0.4389/0.3773	0.5546/0.5021
9-Structure_Divide	0.7733/0.7485	0.5506/0.5301	0.6432/0.6209

Table 2 Evaluation on Structure Features

Structure	Positive Class	Negative Class	Ratio
Nested	6332	4612	1 : 0.7283
Adjacent	2028	27100	1 : 13.3629

[1] Nine structures are combined to three by merging (b) and (c) to (a), (e) and (f) to (d), (h) and (i) to (g).

Separated	939	79989	1 : 85.1853
Total	9299	111701	1 : 12.01

Table 3 Imbalance Training Class Problem

In the experiments, we find that the training class imbalance problem is quite serious, especially for the separated structure (see Table 3 above where "Positive" and "Negative" mean there exists a relation between two entities and otherwise). A possible solution to alleviate this problem is to detect whether the given two entities have some relation first and if they do then to classify the relation types and subtypes instead of combining detection and classification in one process. The second set of experiment is to examine the difference between these two implementations. Against our expectation, the sequence implementation does better than the combination implementation, but not significantly, as shown in Table 4 below.

Type / Subtype	Precision	Recall	F-measure
Combination	0.7733/0.7485	0.5506/0.5301	0.6432/0.6206
Sequence	0.7374/0.7151	0.5860/0.5683	0.6530/0.6333

Table 4 Evaluation of Two Detection and Classification Modes

Based on the sequence implementation, we set up the third set of experiments to examine the correction and inference mechanisms. The results are illustrated in Table 5. The correction with constraints and consistency check is clearly contributing. It improves F-measure 7.40% and 6.47% in type and subtype classification respectively. We further compare four possible consistency check strategies in Table 6 and find that the strategies using subtypes to determine or select types perform better than top down strategies. This can be attributed to the fact that correction with relation/argument constraints in subtype is tighter than the ones in type.

Type / Subtype	Precision	Recall	F-measure
Seq. + Cor.	0.8198/0.7872	0.6127/0.5883	0.7013/0.6734
Seq. + Cor. + Inf.	0.8167/0.7832	0.6170/0.5917	0.7029/0.6741

Table 5 Evaluation of Correction and Inference Mechanisms

Type / Subtype	Precision	Recall	F-measure
Guiding Top-Down	0.7644/0.7853	0.6074/0.5783	0.6770/0.6661
Subtype Selection	0.8069/0.7738	0.6065/0.5817	0.6925/0.6641
Strictly Bottom-Up	0.8120/0.7798	0.6146/0.5903	0.6996/0.6719
Type Selection	0.8198/0.7872	0.6127/0.5883	0.7013/0.6734

Table 6 Comparison of Different Consistency Check Strategies

Finally, we provide our findings from the fourth set of experiments which looks at the detailed contributions from four feature types. Entity type features themselves do not work. We incrementally add the structures, the external contexts and internal contexts, Uni-grams and Bi-grams, and at last the word lists on them. The observations are: Uni-grams provide more discriminative information than Bi-grams; external context seems more useful than internal context; positional structure provides stronger support than other individual recognized features such as entity type and context; but word list feature can not further boost the performance.

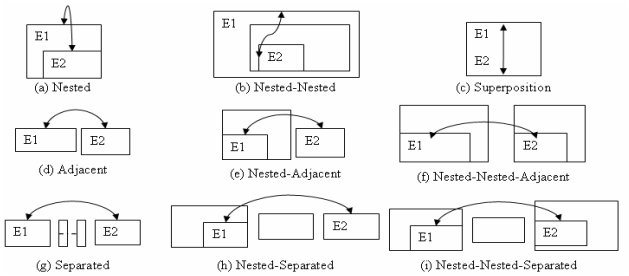
Type / Subtype	Precision	Recall	F-measure
Entity Type + Structure	0.7288/0.6902	0.4876/0.4618	0.5843/0.5534
+ External (Uni-)	0.7935/0.7492	0.5817/0.5478	0.6713/0.6321
+ Internal (Uni-)	0.8137/0.7769	0.6113/0.5836	0.6981/0.6665
+ Bi- (Internal & External)	0.8144/0.7828	0.6141/0.5902	0.7002/0.6730
+ Wordlist	0.8167/0.7832	0.6170/0.5917	0.7029/0.6741

Table 6 Evaluation of Feature and Their Combinations

5 Conclusion

In this paper, we study feature-based Chinese relation extraction. The proposed approach is effective on the ACE 2005 data set. Unfortunately, there is no result reported on the same data so that we can compare.

6 Appendix: Nine Positional Structures



Acknowledgments

This work was supported by HK RGC (CERG PolyU5211/05E) and China NSF (60603027).

References

Razvan Bunescu and Raymond Mooney. 2005. A Shortest Path Dependency Tree Kernel for Relation Extraction, In Proceedings of HLT/EMNLP, pages 724-731.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction, in Proceedings of ACL, pages 423-429.

Jing Jiang, Chengxiang Zhai. 2007. A Systematic Exploration of the Feature Space for Relation Extraction. In proceedings of NAACL/HLT, pages 113-120.

Nanda Kambhatla. 2004. Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. In Proceedings of ACL, pages 178-181.

Dmitry Zelenko, Chinatsu Aone and Anthony Richardella. 2003. Kernel Methods for Relation Extraction. Journal of Machine Learning Research 3:1083-1106

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features, in Proceedings of COLING/ACL, pages 825-832.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring Various Knowledge in Relation Extraction. In Proceedings of ACL, pages 427-434.

GuoDong Zhou, Min Zhang, Donghong Ji and Qiaoming Zhu. 2007. Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. In Proceedings of EMNLP, pages 728-736.

Wanxiang Che et al. 2005. Improved-Edit-Distance Kernel for Chinese Relation Extraction. In Proceedings of IJCNLP, pages 132-137.

Using Structural Information for Identifying Similar Chinese Characters

Chao-Lin Liu **Jen-Hsiang Lin**
Department of Computer Science, National Chengchi University, Taipei 11605, Taiwan
{chaolin, g9429}@cs.nccu.edu.tw

Abstract

Chinese characters that are similar in their pronunciations or in their internal structures are useful for computer-assisted language learning and for psycholinguistic studies. Although it is possible for us to employ image-based methods to identify visually similar characters, the resulting computational costs can be very high. We propose methods for identifying visually similar Chinese characters by adopting and extending the basic concepts of a proven Chinese input method--Cangjie. We present the methods, illustrate how they work, and discuss their weakness in this paper.

1 Introduction

A Chinese sentence consists of a sequence of characters that are not separated by spaces. The function of a Chinese character is not exactly the same as the function of an English word. Normally, two or more Chinese characters form a Chinese word to carry a meaning, although there are Chinese words that contain only one Chinese character. For instance, a translation for "conference" is "研討會" and a translation for "go" is "去". Here "研討會" is a word formed by three characters, and "去" is a word with only one character.

Just like that there are English words that are spelled similarly, there are Chinese characters that are pronounced or written alike. For instance, in English, the sentence "John plays an important roll in this event." contains an incorrect word. We should replace "roll" with "role". In Chinese, the sentence "今天上午我們來**試場**買菜" contains an incorrect word. We should replace "試場" (a place for taking examinations) with "市場" (a market). These two words have the same pronunciation, shi(4) chang(3)[†], and both represent locations. The sentence "經理要我**構買**一部計算機" also contains an error, and we need to replace "構買" with "購買". "構買" is considered an incorrect word, but can be confused with "購買" because the first characters in these words look similar.

Characters that are similar in their appearances or in their pronunciations are useful for computer-assisted language learning (cf. Burstein & Leacock, 2005). When preparing test items for testing students' knowledge about correct words in a computer-assisted environment, a teacher provides a sentence which contains the character that will be replaced by an incorrect character. The teacher needs to specify the answer character, and the software will provide two types of incorrect characters which the teachers will use as distracters in the test items. The first type includes characters that look similar to the answer character, and the second includes characters that have the same or similar pronunciations with the answer character.

Similar characters are also useful for studies in Psycholinguistics. Yeh and Li (2002) studied how similar characters influenced the judgments made by skilled readers of Chinese. Taft, Zhu, and Peng (1999) investigated the effects of positions of radicals on subjects' lexical decisions and naming responses. Computer programs that can automatically provide similar characters are thus potentially helpful for designing related experiments.

2 Identifying Similar Characters with Information about the Internal Structures

We present some similar Chinese characters in the first subsection, illustrate how we encode Chinese characters in the second subsection, elaborate how we improve the current encoding method to facilitate the identification of similar characters in the third subsection, and discuss the weakness of our current approach in the last subsection.

2.1 Examples of Similar Chinese Characters

We show three categories of confusing Chinese characters in Figures 1, 2, and 3. Groups of similar

[†] We use Arabic digits to denote the four tones in Mandarin.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 93–96,
Columbus, Ohio, USA, June 2008.

士土工干千 戌戍成 田由甲申
母毋 勿匆 人入 未末 采采 凹凸

Figure 1. Some similar Chinese characters

頸勁 搆溝 陪倍 硯現 裸棵 搞篙
列刑 盆盎盂盅 因困囚 間閒閃開

Figure 2. Some similar Chinese characters that have different pronunciations

形刑型 踵種腫 購構搆 紀記計
園圓員 脛逕徑痙勁

Figure 3. Homophones with a shared component

characters are separated by spaces in these figures. In Figure 1, characters in each group differ at the stroke level. Similar characters in every group in the first row in Figure 2 share a common part, but the shared part is not the radical of these characters. Similar characters in every group in the second row in Figure 2 share a common part, which is the radical of these characters. Similar characters in every group in Figure 2 have different pronunciations. We show six groups of homophones that also share a component in Figure 3. Characters that are similar in both pronunciations and internal structures are most confusing to new learners.

It is not difficult to list all of those characters that have the same or similar pronunciations, e.g., "試場" and "市場", if we have a machine readable lexicon that provides information about pronunciations of characters and when we ignore special patterns for tone sandhi in Chinese (Chen, 2000).

In contrast, it is relatively difficult to find characters that are written in similar ways, e.g., "構" with "購", in an efficient way. It is intriguing to resort to image processing methods to find such structurally similar words, but the computational costs can be very high, considering that there can be tens of thousands of Chinese characters. There are more than 22000 different characters in large corpus of Chinese documents (Juang et al., 2005), so directly computing the similarity between images of these characters demands a lot of computation. There can be more than 4.9 billion combinations of character pairs. The Ministry of Education in Taiwan suggests that about 5000 characters are needed for ordinary usage. In this case, there are about 25 million pairs.

The quantity of combinations is just one of the bottlenecks. We may have to shift the positions of the characters "appropriately" to find the common part of a character pair. The appropriateness for shifting characters is not easy to define, making the image-based method less directly useful; for instance, the common part of the characters in the right group in the second row in Figure 3 appears in different places in the characters.

Lexicographers employ radicals of Chinese characters to organize Chinese characters into sections in dictionaries. Hence, the information should be useful. The groups in the second row in Figure 2 show some examples. The shared components in these groups are radicals of the characters, so we can find the characters of the same group in the same section in a Chinese dictionary. However, information about radicals as they are defined by the lexicographers is not sufficient. The groups of characters shown in the first row in Figure 2 have shared components. Nevertheless, the shared components are not considered as radicals, so the characters, e.g., "頸"and "勁", are listed in different sections in the dictionary.

2.2 Encoding the Chinese Characters

The Cangjie[‡] method is one of the most popular methods for people to enter Chinese into computers. The designer of the Cangjie method, Mr. Bong-Foo Chu, selected a set of 24 basic elements in Chinese characters, and proposed a set of rules to decompose Chinese characters into elements that belong to this set of building blocks (Chu, 2008). Hence, it is possible to define the similarity between two Chinese characters based on the similarity between their Cangjie codes.

Table 1, not counting the first row, has three

	Cangjie Codes		Cangjie Codes
士	十一	土	土
工	一中一	干	一十
勿	心竹竹	匆	竹田心
未	十木	末	木十
頸	一一一月金	勁	一一大尸
硯	一口月山山	現	一土月山山
搞	手卜口月	篙	竹卜口月
列	一弓中弓	刑	一廿中弓
因	田大	困	田木
間	日弓日	閒	日弓月
踵	口一竹十土	種	竹木竹十土
腫	月竹十土	紀	女火尸山
購	月金廿廿月	構	木廿廿月
記	卜口尸山	計	卜口十
圓	田口月金	員	口月山金
脛	月一女一	逕	卜一女一
徑	竹人一女一	痙	大一女一

Table 1. Cangjie codes for some characters

[‡] http://en.wikipedia.org/wiki/Cangjie_method

sections, each showing the Cangjie codes for some characters in Figures 1, 2, and 3. Every Chinese character is decomposed into an ordered sequence of *elements*. (We will find that a subsequence of these elements comes from a major *component* of a character, shortly.) Evidently, computing the number of shared elements provides a viable way to determine "visually similar" characters for characters that appeared in Figure 2 and Figure 3. For instance, we can tell that "搞" and "篙" are similar because their Cangjie codes share "卜ロ月", which in fact represent "高".

Unfortunately, the Cangjie codes do not appear to be as helpful for identifying the similarities between characters that differ subtly at the stroke level, e.g., "士土工干" and other characters listed in Figure 1. There are special rules for decomposing these relatively basic characters in the Cangjie method, and these special encodings make the resulting codes less useful for our tasks.

The Cangjie codes for characters that contain multiple components were intentionally simplified to allow users to input Chinese characters more efficiently. The longest Cangjie code for any Chinese character contains no more than five elements. In the Cangjie codes for "脛" and "徑", we see "一女一" for the component "巠", but this component is represented only by "一一" in the Cangjie codes for "頸" and "勁". The simplification makes it relatively harder to identify visually similar characters by comparing the actual Cangjie codes.

2.3 Engineering the Original Cangjie Codes

Although useful for the sake of designing input method, the simplification of Cangjie codes causes difficulties when we use the codes to find similar characters. Hence, we choose to use the complete codes for the components in our database. For instance, in our database, the codes for "巠", "脛", "徑", "頸", and "勁" are, respectively, "一女女一", "月一女女一", "竹人一女女一", "一女女一一月山金", and "一女女一大尸".

The knowledge about the graphical structures of the Chinese characters (cf. Juang et al., 2005; Lee, 2008) can be instrumental as well. Consider the examples in Figure 2. Some characters can be decomposed vertically; e.g., "盅" can be split into two smaller components, i.e., "中" and "皿". Some characters can be decomposed horizontally; e.g., "現" is consisted of "王" and "見". Some have enclosing components; e.g., "人" is enclosed in "口" in "囚". Hence, we can consider the locations of the components as well as the number of shared components in determining the similarity between characters.

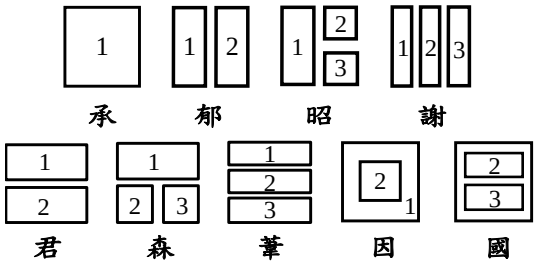


Figure 4. Arrangements of components in Chinese

Figure 4 illustrates possible layouts of the components in Chinese characters that were adopted by the Cangjie method (cf. Lee, 2008). A sample character is placed below each of these layouts. A box in a layout indicates a component in a character, and there can be at most three components in a character. We use digits to indicate the ordering the components. Notice that, in the second row, there are two boxes in the second to the rightmost layout. A larger box contains a smaller one. There are three boxes in the rightmost layout, and two smaller boxes are inside the outer box. Due to space limits, we do not show "1" for this outer box.

After recovering the simplified Cangjie code for a character, we can associate the character with a tag that indicates the overall layout of its components, and separate the code sequence of the character according to the layout of its components. Hence, the information about a character includes the tag for its layout and between one to three sequences of code elements. Table 2 shows the anno-

	Layout	Part 1	Part 2	Part 3
承	1	弓弓手人		
郁	2	大月	弓中	
昭	3	日	尸竹	口
謝	4	卜一一口	竹難竹	木戈
君	5	尸大	口	
森	6	木	木	木
葦	7	廿	木一	手
因	8	田	大	
國	9	田	戈	口一
頸	2	一女女一	一月山金	
徑	2	竹人	一女女一	
員	5	口	月山金	
圓	9	田	口	月山金
相	2	木	月山	
想	5	木月山	心	
箱	6	竹	木	月山

Table 2. Annotated and expanded code

tated and expanded codes of the sample characters in Figure 4 and the codes for some characters that we will discuss. The layouts are numbered from left to right and from top to bottom in Figure 4. Elements that do not belong to the original Canjie codes of the characters are shown in smaller font.

Recovering the elements that were dropped out by the Cangjie method and organizing the subsequences of elements into parts facilitate the identification of similar characters. It is now easier to find that the character (頸) that is represented by "一女女一" and "一月山金" looks similar to the character (徑) that is represented by "竹人" and "一女女一" in our database than using their original Cangjie codes in Table 1. Checking the codes for "員" and "圓" in Table 1 and Table 2 will offer an additional support for our design decisions.

In the worst case, we have to compare nine pairs of code sequences for two characters that both have three components. Since we do not simplify codes for components and all components have no more than five elements, conducting the comparisons operations are simple.

2.4 Drawbacks of Using the Cangjie Codes

Using the Cangjie codes as the basis for comparing the similarity between characters introduces some potential problems.

It appears that the Cangjie codes for some characters, particular those simple ones, were not assigned without ambiguous principles. Relying on Cangjie codes to compute the similarity between such characters can be difficult. For instance, "分" uses the fifth layout, but "兌" uses the first layout in Figure 4. The first section in Table 1 shows the Cangjie codes for some character pairs that are difficult to compare.

Due to the design of the Cangjie codes, there can be at most one component at the left hand side and at most one component at the top in the layouts. The last three entries in Table 2 provide an example for these constraints. As a standalone character, "相" uses the second layout. Like the standalone "相", the "相" in "箱" was divided into two parts. However, in "想", "相" is treated as an individual component because it is on top of "想". Similar problems may occur elsewhere, e.g., "森焚" and "恩因". There are also some exceptional cases; e.g., "品" uses the sixth layout, but "闆" uses the fifth layout.

3 Concluding Remarks

We adopt the Cangjie alphabet to encode Chinese characters, but choose not to simplify the code sequences, and annotate the characters with the layout information of their components. The resulting method is not perfect, but allows us to find visually similar characters more efficient than employing the image-based methods.

Trying to find conceptually similar but contextually inappropriate characters should be a natural step after being able to find characters that have similar pronunciations and that are visually similar.

Acknowledgments

Work reported in this paper was supported in part by the plan NSC-95-2221-E-004-013-MY2 from the National Science Council and in part by the plan ATU-NCCU-96H061 from the Ministry of Education of Taiwan.

References

Jill Burstein and Claudia Leacock. editors. 2005. *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, ACL.

Matthew Y. Chen. 2000. *Tone Sandhi: Patterns across Chinese Dialects*. (Cambridge. Studies in Linguistics 92.) Cambridge: Cambridge University Press.

Bong-Foo Chu. 2008. *Handbook of the Fifth Generation of the Cangjie Input Method*, web version, available at http://www.cbflabs.com/book/ocj5/ocj5/index.html. Last visited on 14 Mar. 2008.

Hsiang Lee. 2008. *Cangjie Input Methods in 30 Days*, http://input.foruto.com/cjdict/Search_1.php, Foruto Company, Hong Kong. Last visited on 14 Mar. 2008.

Derming Juang, Jenq-Haur Wang, Chen-Yu Lai, Ching-Chun Hsieh, Lee-Feng Chien, and Jan-Ming Ho. 2005. Resolving the unencoded character problem for Chinese digital libraries. *Proceedings of the Fifth ACM/IEEE Joint Conference on Digital Libraries*, 311–319.

Marcus Taft, Xiaoping Zhu, and Danling Peng. 1999. Positional specificity of radicals in Chinese character recognition, *Journal of Memory and Language*, **40**, 498–519.

Su-Ling Yeh and Jing-Ling Li. 2002. Role of structure and component in judgments of visual similarity of Chinese characters, *Journal of Experimental Psychology: Human Perception and Performance*, **28**(4), 933–947.

You've Got Answers: Towards Personalized Models for Predicting Success in Community Question Answering

Yandong Liu and **Eugene Agichtein**
Emory University
{yliu49,eugene}@mathcs.emory.edu

Abstract

Question answering communities such as Yahoo! Answers have emerged as a popular alternative to general-purpose web search. By directly interacting with other participants, information seekers can obtain specific answers to their questions. However, user success in obtaining satisfactory answers varies greatly. We hypothesize that satisfaction with the contributed answers is largely determined by the asker's prior experience, expectations, and personal preferences. Hence, we begin to develop *personalized* models of asker satisfaction to predict whether a particular question author will be satisfied with the answers contributed by the community participants. We formalize this problem, and explore a variety of content, structure, and interaction features for this task using standard machine learning techniques. Our experimental evaluation over thousands of real questions indicates that indeed it is beneficial to personalize satisfaction predictions when sufficient prior user history exists, significantly improving accuracy over a "one-size-fits-all" prediction model.

1 Introduction

Community Question Answering (CQA) has recently become a viable method for seeking information online. As an alternative to using general-purpose web search engines, information seekers now have an option to post their questions (often complex, specific, and subjective) on Community QA sites such as Yahoo! Answers, and have their questions answered by other users. Hundreds of millions of answers have already been posted for tens of millions of questions in Yahoo! Answers. However, the success of obtaining *satisfactory* answers in the available CQA portals varies greatly. In many cases, the questions posted by askers go un-answered, or are answered poorly, never obtaining a satisfactory answer.

In our recent work (Liu et al., 2008) we have introduced a general model for predicting asker satisfaction in community question answering. We found that previous asker history is a significant factor that correlates with satisfaction. We hypothesize that asker's satisfaction with contributed answers is largely determined by the asker expectations, prior knowledge and previous experience with using the CQA site. Therefore, in this paper we begin to explore how to *personalize* satisfaction prediction - that is, to attempt to predict whether a *specific* information seeker will be satisfied with any of the contributed answers. Our aim is to provide a "personalized" recommendation to the user that they've got answers that satisfy their information need.

To the best of our knowledge, ours is the first exploration of personalizing prediction of user satisfaction in complex and subjective information seeking environments. While information seeker satisfaction has been studied in ad-hoc IR context (see (Kobayashi and Takeda, 2000) for an overview), previous studies have been limited by the lack of realistic user feedback. In contrast, we deal with complex information needs and community-provided answers, trying to predict subjective ratings provided by users themselves. Furthermore, while automatic complex QA has been an active area of research, ranging from simple modification to factoid QA technique (e.g., (Soricut and Brill, 2004)) to knowledge intensive approaches for specialized domains, the technology does not yet exist to automatically answer open domain, complex, and subjective questions. Hence, this paper contributes to both the understanding of complex question answering, and explores evaluation issues in a new setting.

The rest of the paper is organized as follows. We describe the problem and our approach in Section 2, including our initial attempt at personalizing satisfaction prediction. We report results of a large-scale evaluation over thousands of real users and

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 97–100,
Columbus, Ohio, USA, June 2008.

tens of thousands of questions in Section 3. Our results demonstrate that when sufficient prior asker history exists, even simple personalized models result in significant improvement over a general prediction model. We discuss our findings and future work in Section 4.

2 Predicting Asker Satisfaction in CQA

We first briefly review the life of a question in a QA community. A user (the *asker*) posts a question by selecting a topical category (e.g., "History"), and then enters the question and, optionally, additional details. After a short delay the question appears in the respective category list of *open* questions. At this point, other users can *answer* the question, *vote* on other users' answers, or interact in other ways. The asker may be notified of the answers as they are submitted, or may check the contributed answers periodically. If the asker is satisfied with any of the answers, she can choose it as *best*, and rate the answer by assigning *stars*. At that point, the question is considered as *closed by asker*. For more detailed treatment of user interactions in CQA see (Liu et al., 2008). If the asker rates the best answer with at least three out of five "stars", we believe the asker is satisfied with the response. But often the asker never closes the answer personally, and instead, after a period of time, the question is *closed automatically*. In this case, the "best" answer may be chosen by the votes, or alternatively by automatically predicting answer quality (e.g., (Jeon et al., 2006) or (Agichtein et al., 2008)). While the best answer chosen automatically may be of high quality, it is unknown if the asker's information need was satisfied.

Based on our exploration we believe that the main reasons for not "closing" a question are a) the asker loses interest in the information and b) none of the answers are satisfactory. In both cases, the QA community has failed to provide satisfactory answers in a timely manner and "lost" the asker's interest. We consider this outcome to be "unsatisfied". We now define *asker satisfaction* more precisely:

Definition 1 *An asker in a QA community is considered* satisfied *iff: the asker personally has closed the question and rated the best answer with at least 3 "stars". Otherwise, the asker is* unsatisfied.

This definition captures a key aspect of asker satisfaction, namely that we can reliably identify when the asker is satisfied but not the converse.

2.1 Asker Satisfaction Prediction Framework

We now briefly review our ASP (Asker Satisfaction Prediction) framework that learns to classify whether a question has been satisfactorily answered, originally introduced in (Liu et al., 2008). ASP employs standard classification techniques to predict, given a *question thread*, whether an asker would be satisfied. A sample of features used to represent this problem is listed in Table 1. Our features are organized around the basic entities in a question answering community: questions, answers, question-answer pairs, users, and categories. In total, we developed 51 features for this task. A sample of the features used are listed in the Figure 1.

- *Question Features*: Traditional question answering features such as the wh-type of the question (e.g., "what" or "where"), and whether the question is similar to other questions in the category.
- *Question-Answer Relationship Features*: Overlap between question and answer, answer length, and number of candidate answers. We also use features such as the number of positive votes ("thumbs up" in Yahoo! Answers), negative votes ("thumbs down"), and derived statistics such as the maximum of positive or negative votes received for any answer (e.g., to detect cases of brilliant answers or, conversely, blatant abuse).
- *Asker User History*: Past asker activity history such as the most recent rating, average past satisfaction, and number of previous questions posted. Note that only the information available about the asker *prior* to posting the question was used.
- *Category Features*: We hypothesized that user behavior (and asker satisfaction) varies by topical question category, as recently shown in reference (Agichtein et al., 2008). Therefore we model the *prior* of asker satisfaction for the category, such as the average asker rating (satisfaction).
- *Text Features*: We also include word unigrams and bigrams to represent the text of the question subject, question detail, and the answer content. Separate feature spaces were used for each attribute to keep answer text distinct from question text, with frequency-based filtering.

Classification Algorithms: We experimented with a variety of classifiers in the Weka framework (Witten and Frank, 2005). In particular, we compared Support Vector Machines, Decision trees, and Boosting-based classifiers. SVM performed the best

Feature	Description
Question Features	
Q: Q_punctuation_density	Ratio of punctuation to words in the question
Q: Q_KL_div_wikipedia	KL divergence with Wikipedia corpus
Q: Q_KL_div_category	KL divergence with "satisfied" questions in category
Q: Q_KL_div_trec	KL divergence with TREC questions corpus
Question-Answer Relationship Features	
QA: QA_sum_pos_vote	Sum of positive votes for all the answers
QA: QA_sum_neg_vote	Sum of negative votes for all the answers
QA: QA_KL_div_wikipedia	KL Divergence of all answers with Wikipedia corpus
Asker User History Features	
UH: UH_questions_resolved	Number of questions resolved in the past
UH: UH_num_answers	Number of all answers this user has received in the past
UH: UH_more_recent_rating	Rating for the last question before current question
UH: UH_avg_past_rating	Average rating given when closing questions in the past
Category Features	
CA: CA_avg_time_to_close	Average interval between opening and closing
CA: CA_avg_num_answers	Average number of answers for that category
CA: CA_avg_asker_rating	Average rating given by asker for category
CA: CA_avg_num_votes	Average number of "best answer" votes in category

Table 1: Sample features: Question (Q), Question-Answer Relationship (QA), Asker history (UH), and Category (CA).

of the three during development, so we report results using SVM for all the subsequent experiments.

2.2 Personalizing Asker Satisfaction Prediction

We now describe our initial attempt at personalizing the ASP framework described above to each asker:

- **ASP_Pers+Text**: We first consider the naive personalization approach where we train a separate classifier for each user. That is, to predict a particular asker's satisfaction with the provided answers, we apply the individual classifier trained solely on the questions (and satisfaction labels) provided in the past by that user.
- **ASP_Group**: A more robust approach is to train a classifier on the questions from the group of users *similar* to each other. Our current grouping was done simply by the number of questions posted, essentially grouping users with similar levels of "activity". As we will show below, text features only help for users with at least 20 previous questions. So, we only include text features for groups of users with at least 20 questions.

Certainly, more sophisticated personalization models and user clustering methods could be devised. However, as we show next, even the simple models described above prove surprisingly effective.

3 Experimental Evaluation

We want to predict, for a given user and their *current* question whether the user will be satisfied, according to our definition in Section 2. In other words, our "truth" labels are based on the rating subsequently given to the best answer by the asker herself. It is usually more valuable to correctly predict whether a user is satisfied (e.g., to notify a user of success).

#Questions per Asker	*# Questions*	*# Answers*	*# Users*
1	132,279	1,197,089	132,279
2	31,692	287,681	15,846
3-4	23,296	213,507	7,048
5-9	15,811	143,483	2,568
10-14	5,554	54,781	481
15-19	2,304	21,835	137
20-29	2,226	23,729	93
30-49	1,866	16,982	49
50-100	842	4,528	14
Total:	216,170	1,963,615	158,515

Table 2: Distribution of questions, answers and askers

Hence, we focus on the *Precision*, *Recall*, and *F1* values for the *satisfied* class.

Datasets: Our data was based on a snapshot of Yahoo! Answers crawled in early 2008, containing 216,170 questions posted in 100 topical categories by 158,515 askers, with associated 1,963,615 answers in total. More detailed statistics, arranged by the number of questions posted by each asker are reported in (Table 2). The askers with only one question (i.e., no prior history) dominate the dataset, as many users try the service once and never come back. However, for personalized satisfaction, at least *some* prior history is needed. Therefore, in this early version of our work, we focus on users who have posted at least 2 questions - i.e., have the minimal history of at least one prior question. In the future, we plan to address the "cold start" problem of predicting satisfaction of new users.

Methods compared:

- **ASP**: A "one-size-fits-all" satisfaction predictor that is trained on 10,000 randomly sampled questions with only non-textual features (Section 2.1).
- **ASP+Text**: The ASP classifier with text features.
- **ASP_Pers+Text** and **ASP_Group**: A personalized classifiers described in Section 2.2.

3.1 Experimental Results

Figure 1 reports the satisfaction prediction accuracy for **ASP**, **ASP_Text**, **ASP_Pers+Text**, and **ASP_Group** for groups of askers with varying number of previous questions posted. Surprisingly, for **ASP_Text**, textual features only become helpful for users with more than 20 or 30 previous questions posted and degrade performance otherwise. Also note that baseline **ASP** classifier is not able to achieve higher accuracy even for users with large amount of past history. In contrast, the **ASP_Pers+Text** classifier, trained only on the past question(s) of each user, achieves surprisingly good accuracy – often significantly outperforming the **ASP** and **ASP_Text** classifiers. The improvement is especially dramatic for users with at least

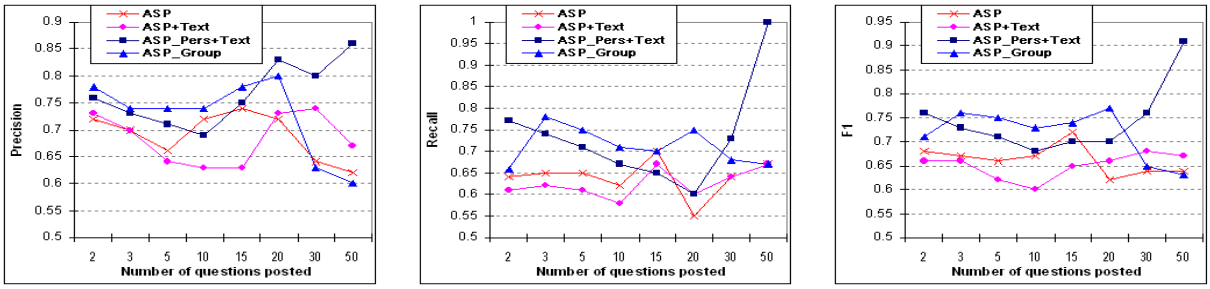


Figure 1: Precision, Recall, and F1 of ASP, ASP_Text, ASP_Pers+Text, and ASP_Group for predicting satisfaction of askers with varying number of questions

20 previous questions. Interestingly, the simple strategy of grouping users by number of previous questions (**ASP_Group**) is even more effective, resulting in accuracy higher than both other methods for users with moderate amount of history. Finally, for users with only 2 questions total (that is, only 1 previous question posted) the performance of **ASP_Pers+Text** is surprisingly high. We found that the classifier simply "memorizes" the outcome of the only available previous question, and uses it to predict the rating of the current question.

To better understand the improvement of personalized models, we report the most significant features, sorted by Information Gain (IG), for three sample **ASP_Pers+Text** models (Table 3). Interestingly, whereas for Pers 1 and Pers 2, textual features such as "good luck" in the answer are significant, for Pers 3 non-textual features are most significant.

We also report the top 10 features with the highest information gain for the **ASP** and **ASP_Group** models (Table 4). Interestingly, while asker's average previous rating is the top feature for **ASP**, the length of membership of the asker is the most important feature for **ASP_Group**, perhaps allowing the classifier to distinguish more expert users from the active newbies. In summary, we have demonstrated promising preliminary results on personalizing satisfaction prediction even with relatively simple personalization models.

Pers 1 (97 questions)	Pers 2 (49 questions)	Pers 3 (25 questions)
UH_total_answers_received	Q_avg_pos_votes	Q_content_kl_trec
UH_questions_resolved	"would" in answer	Q_content_kl_wikipedia
"good luck" in answer	"answer" in question	UH_total_answers_received
"is an" in answer	"just" in answer	UH_questions_resolved
"want to" in answer	"me" in answer	Q_content_kl_asker_all_cate
"we" in answer	"be" in answer	Q_prev_avg_rating
"want in" answer	"in the" in question	CA_avg_asker_rating
"adenocarcinoma" in question	CA_History	"anybody" in question
"was" in question	"who is" in question	Q_content_typo_density
"live" in answer	"those" in answer	Q_detail_len

Table 3: Top 10 features by Information Gain for three sample ASP_Pers+Text models

IG	ASP	IG	ASP_Group
0.104117	Q_prev_avg_rating	0.30981	UH_membersince_in_days
0.102117	Q_most_recent_rating	0.25541	Q_prev_avg_rating
0.047222	Q_avg_pos_vote	0.22556	Q_most_recent_rating
0.041773	Q_sum_pos_vote	0.15237	CA_avg_num_votes
0.041076	Q_max_pos_vote	0.14466	CA_avg_time_close
0.03535	A_ques_timediff_in_minutes	0.13489	CA_avg_asker_rating
0.032261	UH_membersince_in_days	0.13175	CA_num_ans_per_hour
0.031812	CA_avg_asker_rating	0.12437	CA_num_ques_per_hour
0.03001	CA_ratio_ans_ques	0.09314	Q_avg_pos_vote
0.029858	CA_num_ans_per_hour	0.08572	CA_ratio_ans_ques

Table 4: Top 10 features by information gain for ASP (trained for all askers) and ASP_Group (trained for the group of askers with 20 to 29 questions)

4 Conclusions

We have presented preliminary results on personalizing satisfaction prediction, demonstrating significant accuracy improvements over a "one-size-fits-all" satisfaction prediction model. In the future we plan to explore the personalization more deeply following the rich work in recommender systems and collaborative filtering, with the key difference that the asker satisfaction, and each question, are unique (instead of shared items such as movies). In summary, our work opens a promising direction towards modeling personalized user intent, expectations, and satisfaction.

References

E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. 2008. Finding high-quality content in social media with an application to community-based question answering. In *Proceedings of WSDM*.

J. Jeon, W.B. Croft, J.H. Lee, and S. Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of SIGIR*.

Mei Kobayashi and Koichi Takeda. 2000. Information retrieval on the web. *ACM Computing Surveys*, 32(2).

Y. Liu, J. Bian, and E. Agichtein. 2008. Predicting information seeker satisfaction in community question answering. In *Proceedings of SIGIR*.

R. Soricut and E. Brill. 2004. Automatic question answering: Beyond the factoid. In *HLT-NAACL*.

I. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufman, 2nd edition.

Self-Training for Biomedical Parsing

David McClosky and Eugene Charniak
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{dmcc|ec}@cs.brown.edu

Abstract

Parser self-training is the technique of taking an existing parser, parsing extra data and then creating a second parser by treating the extra data as further training data. Here we apply this technique to parser adaptation. In particular, we self-train the standard Charniak/Johnson Penn-Treebank parser using unlabeled biomedical abstracts. This achieves an f-score of 84.3% on a standard test set of biomedical abstracts from the Genia corpus. This is a 20% error reduction over the best previous result on biomedical data (80.2% on the same test set).

1 Introduction

Parser self-training is the technique of taking an existing parser, parsing extra data and then creating a second parser by treating the extra data as further training data. While for many years it was thought not to help state-of-the art parsers, more recent work has shown otherwise. In this paper we apply this technique to parser adaptation. In particular we self-train the standard Charniak/Johnson Penn-Treebank (C/J) parser using unannotated biomedical data. As is well known, biomedical data is hard on parsers because it is so far from more "standard" English. To our knowledge this is the first application of self-training where the gap between the training and self-training data is so large.

In section two, we look at previous work. In particular we note that there is, in fact, very little data on self-training when the corpora for self-training is so different from the original labeled data. Section three describes our main experiment on standard test data (Clegg and Shepherd, 2005). Section four looks at some preliminary results we obtained on development data that show in slightly more detail how self-training improved the parser. We conclude in section five.

2 Previous Work

While self-training has worked in several domains, the early results on self-training for parsing were negative (Steedman et al., 2003; Charniak, 1997). However more recent results have shown that it can indeed improve parser performance (Bacchiani et al., 2006; McClosky et al., 2006a; McClosky et al., 2006b).

One possible use for this technique is for parser adaptation — initially training the parser on one type of data for which hand-labeled trees are available (e.g., Wall Street Journal (M. Marcus et al., 1993)) and then self-training on a second type of data in order to adapt the parser to the second domain. Interestingly, there is little to no data showing that this actually works. Two previous papers would seem to address this issue: the work by Bacchiani et al. (2006) and McClosky et al. (2006b). However, in both cases the evidence is equivocal.

Bacchiani and Roark train the Roark parser (Roark, 2001) on trees from the Brown treebank and then self-train and test on data from Wall Street Journal. While they show some improvement (from 75.7% to 80.5% f-score) there are several aspects of this work which leave its re-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 101–104,
Columbus, Ohio, USA, June 2008.

sults less than convincing as to the utility of self-training for adaptation. The first is the parsing results are quite poor by modern standards.[1] Steedman et al. (2003) generally found that self-training does not work, but found that it does help if the baseline results were sufficiently bad.

Secondly, the difference between the Brown corpus treebank and the Wall Street Journal corpus is not that great. One way to see this is to look at out-of-vocabulary statistics. The Brown corpus has an out-of-vocabulary rate of approximately 6% when given WSJ training as the lexicon. In contrast, the out-of-vocabulary rate of biomedical abstracts given the same lexicon is significantly higher at about 25% (Lease and Charniak, 2005). Thus the bridge the self-trained parser is asked to build is quite short.

This second point is emphasized by the second paper on self-training for adaptation (McClosky et al., 2006b). This paper is based on the C/J parser and thus its results are much more in line with modern expectations. In particular, it was able to achieve an f-score of 87% on Brown treebank test data when trained and self-trained on WSJ-like data. Note this last point. It was not the case that it used the self-training to bridge the corpora difference. It self-trained on NANC, *not* Brown. NANC is a news corpus, quite like WSJ data. Thus the point of that paper was that self-training a WSJ parser on similar data makes the parser more flexible, not better adapted to the target domain in particular. It said nothing about the task we address here. Thus our claim is that previous results are quite ambiguous on the issue of bridging corpora for parser adaptation.

Turning briefly to previous results on Medline data, the best comparative study of parsers is that of Clegg and Shepherd (2005), which evaluates several statistical parsers. Their best result was an f-score of 80.2%. This was on the Lease/Charniak (L/C) parser (Lease and Charniak, 2005).[2] A close second (1% behind) was the parser of Bikel (2004). The other parsers were not close. However, several very good current parsers were not available when this paper was written (e.g., the Berkeley Parser (Petrov et al., 2006)). However, since the newer parsers do not perform quite as well as the C/J parser on WSJ data, it is probably the case that they would not significantly alter the landscape.

3 Central Experimental Result

We used as the base parser the standardly available C/J parser. We then self-trained the parser on approximately 270,000 sentences — a random selection of abstracts from Medline.[3] Medline is a large database of abstracts and citations from a wide variety of biomedical literature. As we note in the next section, the number 270,000 was selected by observing performance on a development set.

We weighted the original WSJ hand annotated sentences equally with self-trained Medline data. So, for example, McClosky et al. (2006a) found that the data from the hand-annotated WSJ data should be considered at least five times more important than NANC data on an event by event level. We did no tuning to find out if there is some better weighting for our domain than one-to-one.

The resulting parser was tested on a test corpus of hand-parsed sentences from the Genia Treebank (Tateisi et al., 2005). These are exactly the same sentences as used in the comparisons of the last section. Genia is a corpus of abstracts from the Medline database selected from a search with the keywords Human, Blood Cells, and Transcription Factors. Thus the Genia treebank data are all from a small domain within Biology. As already noted, the Medline abstracts used for self-training were chosen randomly and thus span a large number of biomedical sub-domains.

The results, the central results of this paper, are shown in Figure 1. Clegg and Shepherd (2005) do not provide separate precision and recall numbers. However we can see that the

[1] This is not a criticism of the work. The results are completely in line with what one would expect given the base parser and the relatively small size of the Brown treebank.

[2] This is the standard Charniak parser (without reranker) modified to use an in-domain tagger.

[3] `http://www.ncbi.nlm.nih.gov/PubMed/`

System	Precision	Recall	f-score
L/C	—	—	80.2%
Self-trained	86.3%	82.4%	84.3%

Figure 1: Comparison of the Medline self-trained parser against the previous best

Medline self-trained parser achieves an f-score of 84.3%, which is an absolute reduction in error of 4.1%. This corresponds to an error rate reduction of 20% over the L/C baseline.

4 Discussion

Prior to the above experiment on the test data, we did several preliminary experiments on development data from the Genia Treebank. These results are summarized in Figure 2. Here we show the f-score for four versions of the parser as a function of number of self-training sentences. The dashed line on the bottom is the raw C/J parser with no self-training. At 80.4, it is clearly the worst of the lot. On the other hand, it is already better than the 80.2% best previous result for biomedical data. This is solely due to the introduction of the 50-best reranker which distinguishes the C/J parser from the preceding Charniak parser.

The almost flat line above it is the C/J parser with NANC self-training data. As mentioned previously, NANC is a news corpus, quite like the original WSJ data. At 81.4% it gives us a one percent improvement over the original WSJ parser.

The topmost line, is the C/J parser trained on Medline data. As can be seen, even just a thousand lines of Medline is already enough to drive our results to a new level and it continues to improve until about 150,000 sentences at which point performance is nearly flat. However, as 270,000 sentences is fractionally better than 150,000 sentences that is the number of self-training sentences we used for our results on the test set.

Lastly, the middle jagged line is for an interesting idea that failed to work. We mention it in the hope that others might be able to succeed where we have failed.

We reasoned that textbooks would be a particularly good bridging corpus. After all, they are written to introduce someone ignorant of a field to the ideas and terminology within it. Thus one might expect that the English of a Biology textbook would be intermediate between the more typical English of a news article and the specialized English native to the domain.

To test this we created a corpus of seven texts ("BioBooks") on various areas of biology that were available on the web. We observe in Figure 2 that for all quantities of self-training data one does better with Medline than BioBooks. For example, at 37,000 sentences the BioBook corpus is only able to achieve and an f-measure of 82.8% while the Medline corpus is at 83.4%. Furthermore, BioBooks levels off in performance while Medline has significant improvement left in it. Thus, while the hypothesis seems reasonable, we were unable to make it work.

5 Conclusion

We self-trained the standard C/J parser on 270,000 sentences of Medline abstracts. By doing so we achieved a 20% error reduction over the best previous result for biomedical parsing. In terms of the gap between the supervised data and the self-trained data, this is the largest that has been attempted.

Furthermore, the resulting parser is of interest in its own right, being as it is the most accurate biomedical parser yet developed. This parser is available on the web.[4]

Finally, there is no reason to believe that 84.3% is an upper bound on what can be achieved with current techniques. Lease and Charniak (2005) achieve their results using small amounts of hand-annotated biomedical part-of-speech-tagged data and also explore other possible sources or information. It is reasonable to assume that its use would result in further improvement.

Acknowledgments

This work was supported by DARPA GALE contract HR0011-06-2-0001. We would like to thank the BLLIP team for their comments.

[4] `http://bllip.cs.brown.edu/biomedical/`

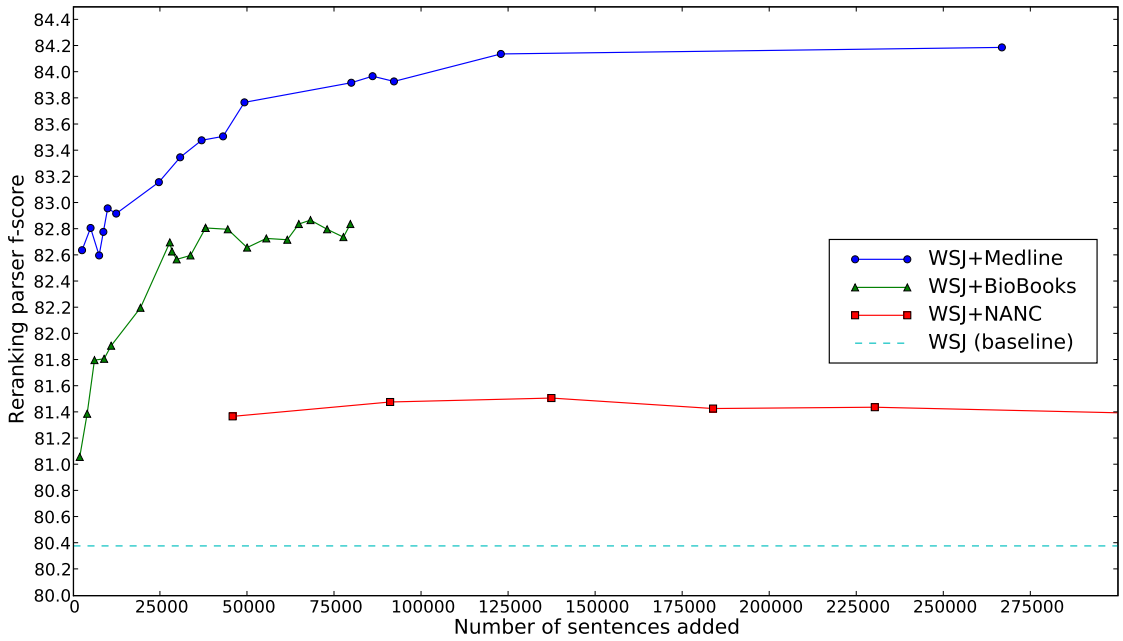


Figure 2: Labeled Precision-Recall results on development data for four versions of the parser as a function of number of self-training sentences

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Daniel M. Bikel. 2004. Intricacies of collins parsing model. *Computational Linguistics*, 30(4).

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. AAAI*, pages 598–603.

Andrew B. Clegg and Adrian Shepherd. 2005. Evaluating and integrating treebank parsers on a biomedical corpus. In *Proceedings of the ACL Workshop on Software*.

Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *Second International Joint Conference on Natural Language Processing (IJCNLP'05)*.

M. Marcus et al. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comp. Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of COLING-ACL 2006*, pages 337–344, Sydney, Australia, July. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL 2006*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proc. of European ACL (EACL)*, pages 331–338.

Y. Tateisi, A. Yakushiji, T. Ohta, and J. Tsujii. 2005. Syntax Annotation for the GENIA corpus. *Proc. IJCNLP 2005, Companion volume*, pages 222–227.

A Unified Syntactic Model for Parsing Fluent and Disfluent Speech*

Tim Miller
University of Minnesota
tmill@cs.umn.edu

William Schuler
University of Minnesota
schuler@cs.umn.edu

Abstract

This paper describes a syntactic representation for modeling speech repairs. This representation makes use of a right corner transform of syntax trees to produce a tree representation in which speech repairs require very few special syntax rules, making better use of training data. PCFGs trained on syntax trees using this model achieve high accuracy on the standard Switchboard parsing task.

1 Introduction

Speech repairs occur when a speaker makes a mistake and decides to partially retrace an utterance in order to correct it. Speech repairs are common in spontaneous speech – one study found 30% of dialogue turns contained repairs (Carletta et al., 1993) and another study found one repair every 4.8 seconds (Blackmer and Mitton, 1991). Because of the relatively high frequency of this phenomenon, spontaneous speech recognition systems will need to be able to deal with repairs to achieve high levels of accuracy.

The speech repair terminology used here follows that of Shriberg (1994). A speech repair consists of a *reparandum*, an *interruption point*, and the *alteration*. The reparandum contains the words that the speaker means to replace, including both words that are in error and words that will be retraced. The interruption point is the point in time where the stream of speech is actually stopped, and the repairing of the mistake can begin. The alteration contains the words that are meant to replace the words in the reparandum.

Recent advances in recognizing spontaneous speech with repairs (Hale et al., 2006; Johnson and Charniak, 2004) have used parsing approaches on transcribed speech to account for the structure inherent in speech repairs at the word level and above. One salient aspect of structure is the fact that there is often a good deal of overlap in words between the reparandum and the alteration, as speakers may trace back several words when restarting after an error. For instance, in the repair *. . . a flight to Boston, uh, I mean, to Denver on Friday . . .*, there is an exact match of the word 'to' between reparandum and repair, and a part of speech match between the words 'Boston' and 'Denver'.

Another sort of structure in repair is what Levelt (1983) called the well-formedness rule. This rule states that the constituent started in the reparandum and repair are ultimately of syntactic types that *could* be grammatically joined by a conjunction. For example, in the repair above, the well-formedness rule says that the repair is well formed if the fragment *. . . a flight to Boston and to Denver. . .* is grammatical. In this case the repair is well formed since the conjunction is grammatical, if not meaningful.

The approach described here makes use of a transform on a tree-annotated corpus to build a syntactic model of speech repair which takes advantage of the structure of speech repairs as described above, while also providing a representation of repair structure that more closely adheres to intuitions about what happens when speakers make repairs.

*This research was supported by NSF CAREER award 0447685. The views expressed are not necessarily endorsed by the sponsors.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 105–108,
Columbus, Ohio, USA, June 2008.

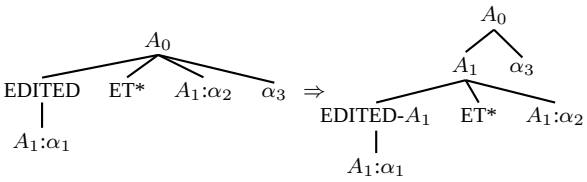
2 Speech repair representation

The representational scheme used for this work makes use of a *right-corner transform*, a way of rewriting syntax trees that turns all right recursion into left recursion, and leaves left recursion as is. As a result, constituent structure is built up during recognition in a left-to-right fashion, as words are read in. This arrangement is well-suited to recognition of speech with repairs, because it allows for constituent structure to be built up using fluent speech rules up until the moment of interruption, at which point a special repair rule may be applied. This property will be examined further in section 2.3, following a technical description of the representation scheme.

2.1 Binary branching structure

In order to obtain a linguistically plausible right-corner transform representation of incomplete constituents, the Switchboard corpus is subjected to a pre-process transform to introduce binary-branching nonterminal projections, and fold empty categories into nonterminal symbols in a manner similar to that proposed by Johnson (1998b) and Klein and Manning (2003). This binarization is done in in such a way as to preserve linguistic intuitions of head projection, so that the depth requirements of right-corner transformed trees will be reasonable approximations to the working memory requirements of a human reader or listener.

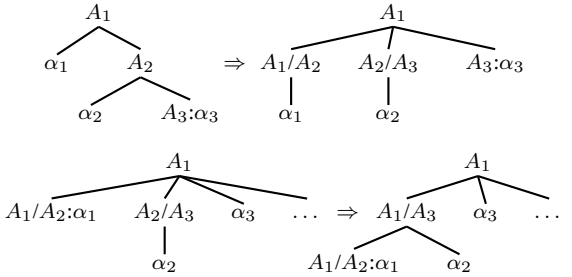
Trees containing speech repairs are reduced in arity by merging repair structure lower in the tree, when possible. As seen in the left tree below, [1] repair structure is annotated in a flat manner, which can lead to high-arity rules which are sparsely represented in the data set, and thus difficult to learn. This problem can be mitigated by using the rewrite rule shown below, which turns an EDITED-X constituent into the leftmost child of a tree of type X, as long as the original flat tree had X following an EDITED-X constituent and possibly some editing term (ET) categories. The INTJ category ('uh','um',etc.) and the PRN category ('I mean', 'that is', etc.) are considered to be editing term categories when they lie between EDITED-X and X constituents.



2.2 Right-corner transform

Binarized trees[2] are then transformed into *right-corner* trees using transform rules similar to those described by Johnson(1998a). This right-corner transform is simply the left-right dual of a left-corner transform. It transforms all right recursive sequences in each tree into left recursive sequences of symbols of the form A_1/A_2, denoting an incomplete instance of category A_1 lacking an instance of category A_2 to the right.

Rewrite rules for the right-corner transform are shown below:



Here, the first rewrite rule is applied iteratively (bottom-up on the tree) to flatten all right recursion, using incomplete constituents to record the original nonterminal ordering. The second rule is then applied to generate left recursive structure, preserving this ordering.

The incomplete constituent categories created by the right corner transform are similar in form and meaning to non-constituent categories used in Combinatorial Categorial Grammars (CCGs) (Steedman, 2000). Unlike CCGs, however, a right corner transformed grammar does not allow backward function application, composition, or raising. As a result, it does not introduce spurious ambiguity between forward and backward operations, but cannot be taken to explicitly encode argument structure, as CCGs can.

[1] Here, all A_i denote nonterminal symbols, and all α_i denote subtrees; the notation A_1:α_1 indicates a subtree α_1 with label A_1; and all rewrites are applied recursively, from leaves to root.

[2] All super-binary branches remaining after the above preprocess are 'nominally' decomposed into right-branching structures by introducing intermediate nodes with labels concatenated from the labels of its children, delimited by underscores

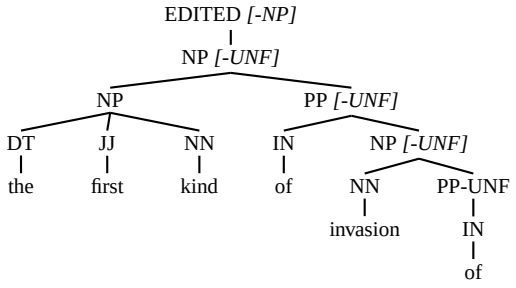


Figure 1: Standard tree repair structure, with -UNF propagation as in (Hale et al., 2006) shown in brackets.

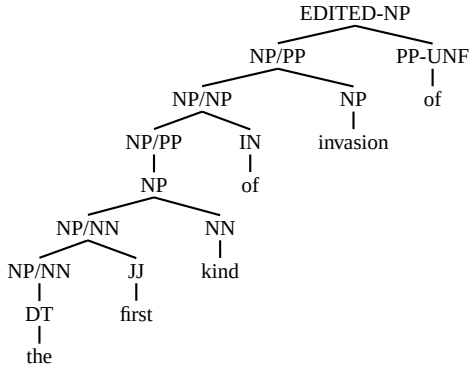


Figure 2: Right-corner transformed tree with repair structure

2.3 Application to speech repair

An example speech repair from the Switchboard corpus can be seen in Figures 1 and 2, in which the same repair fragment is shown in a standard state such as might be used to train a probabilistic context free grammar, and after the right-corner transform. Figure 1 also shows, in brackets, the augmented annotation used by Hale et al.(2006). This scheme consisted of adding -X to an EDITED label which produced a category X, as well as propagating the -UNF label at the right corner of the tree up through every parent below the EDITED root.

The standard annotation (without -UNF propagation) is deficient because even if an unfinished constituent like PP-UNF is correctly recognized, and the speaker is essentially in an error state, there may be several partially completed constituents above – in Figure 1, the NP, PP, and NP above the PP-UNF. These constituents need to be completed, but using the standard annotation there is only one chance to make use of the information about the error that has occurred – the NP → NP PP-UNF rule. Thus, by the time the error section is completed, there is no information by which a parsing algorithm could choose to reduce the topmost NP to EDITED other than independent rule probabilities.

The approach used by (Hale et al., 2006) works because the information about the transition to an error state is propagated up the tree, in the form of the -UNF tags. As the parsing chart is filled in bottom up, each rule applied is essentially coming out of a special repair rule set, and so at the top of the tree the EDITED hypothesis is much more likely. However, this requires that several fluent speech rules from the data set be modified for use in a special repair grammar, which not only reduces the amount of available training data, but violates our intuition that most reparanda are fluent up until the actual edit occurs.

The right corner transform model works in a different way, by building up constituent structure from left to right. In Figure 2, the same fragment is shown as it appears in the training data for this system. With this representation, the problem noticed by Hale and colleagues (2006) has been solved in a different way, by incrementally building up *left-branching* rather than right-branching structure, so that only a single special error rule is required at the end of the constituent. Whereas the -UNF propagation scheme often requires the entire reparandum to be generated from a speech repair rule set, this scheme only requires one special rule, where the moment of interruption actually occurred.

This is not only a pleasing parsimony, but it reduces the number of special speech repair rules that need to be learned and saves more potential examples of fluent speech rules, and therefore potentially makes better use of limited data.

3 Evaluation

The evaluation of this system was performed on the Switchboard corpus, using the *mrg* annotations in directories 2 and 3 for training, and the files sw4004.mrg to sw4153.mrg in directory 4 for evaluation, following Johnson and Charniak(2004).

The input to the system consists of the terminal symbols from the trees in the corpus section mentioned above. The terminal symbol strings are first pre-processed by stripping punctuation and other

System	Parseval F	EDIT F
Baseline	**60.86**	**42.39**
CYK (H06)	71.16	41.7
RCT	**68.36**	**64.41**
TAG-based model (JC04)	–	79.7

Table 1: Baseline results are from a standard CYK parser with binarized grammar. We were unable to find the correct configuration to match the baseline results from Hale et al. RCT results are on the right-corner transformed grammar (transformed back to flat treebank-style trees for scoring purposes). CYK and TAG lines show relevant results from related work.

non-vocalized terminal symbols, which could not be expected from the output of a speech recognizer. Crucially, any information about repair is stripped from the input, including partial words, repair symbols [3], and interruption point information. While an integrated system for processing and parsing speech may use both acoustic and syntactic information to find repairs, and thus may have access to some of this information about where interruptions occur, this experiment is intended to evaluate the use of the right corner transform and syntactic information on parsing speech repair. To make a fair comparison to the CYK baseline of (Hale et al., 2006), the recognizer was given correct part-of-speech tags as input along with words.

The results presented here use two standard metrics for assessing accuracy of transcribed speech with repairs. The first metric, Parseval F-measure, takes into account precision and recall of all non-terminal (and non pre-terminal) constituents in a hypothesized tree relative to the gold standard. The second metric, EDIT-finding F, measures precision and recall of the words tagged as EDITED in the hypothesized tree relative to those tagged EDITED in the gold standard. F score is defined as usual, $2pr/(p+r)$ for precision p and recall r.

The results in Table 1 show that this system performs comparably to the state of the art in overall parsing accuracy and reasonably well in edit detection. The TAG system (Johnson and Charniak, 2004) achieves a higher EDIT-F score, largely as a result of its explicit tracking of overlapping words between reparanda and alterations. A hybrid system using the right corner transform and keeping information about how a repair started may be able to improve EDIT-F accuracy over this system.

[3]The Switchboard corpus has special terminal symbols indicating e.g. the start and end of the reparandum.

4 Conclusion

This paper has described a novel method for parsing speech that contains speech repairs. This system achieves high accuracy in both parsing and detecting reparanda in text, by making use of transformations that create incomplete categories, which model the reparanda of speech repair well.

References

Elizabeth R. Blackmer and Janet L. Mitton. 1991. Theories of monitoring and the timing of repairs in spontaneous speech. *Cognition*, 39:173–194.

Jean Carletta, Richard Caley, and Stephen Isard. 1993. A collection of self-repairs from the map task corpus. Technical report, Human Communication Research Centre, University of Edinburgh.

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover, and Robin Stewart. 2006. PCFGs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (COLING-ACL)*.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, pages 33–39, Barcelona, Spain.

Mark Johnson. 1998a. Finite state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of COLING/ACL*, pages 619–623.

Mark Johnson. 1998b. PCFG models of linguistic tree representation. *Computational Linguistics*, 24:613–632.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

William J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14:41–104.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California at Berkeley.

Mark Steedman. 2000. *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.

The Good, the Bad, and the Unknown: Morphosyllabic Sentiment Tagging of Unseen Words

Karo Moilanen and Stephen Pulman
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
{ Karo.Moilanen | Stephen.Pulman }@comlab.ox.ac.uk

Abstract

The omnipresence of unknown words is a problem that any NLP component needs to address in some form. While there exist many established techniques for dealing with unknown words in the realm of POS-tagging, for example, guessing unknown words' semantic properties is a less-explored area with greater challenges. In this paper, we study the semantic field of sentiment and propose five methods for assigning prior sentiment polarities to unknown words based on known sentiment carriers. Tested on 2000 cases, the methods mirror human judgements closely in three- and two-way polarity classification tasks, and reach accuracies above 63% and 81%, respectively.

1 Introduction

One of the first challenges in sentiment analysis is the vast lexical diversity of subjective language. Gaps in lexical coverage will be a problem for any sentiment classification algorithm that does not have some way of intelligently guessing the polarity of unknown words. The problem is exacerbated further by misspellings of known words and POS-tagging errors which are often difficult to distinguish from genuinely unknown words. This study explores the extent to which it is possible to categorise words which present themselves as unknown, but which may contain known components using morphological, syllabic, and shallow parsing devices.

2 Morphosyllabic Modelling

Our core sentiment lexicon contains 41109 entries tagged with positive (+), neutral (N), or negative (-) prior polarities (e.g. *lovely*$^{(+)}$, *vast*$^{(N)}$, *murder*$^{(-)}$) across all word classes. Polarity reversal lexemes are tagged as [¬] (e.g. *never*$^{(N)[\neg]}$). We furthermore maintain an auxiliary lexicon of 314967 known neutral words such as names of people, organisations, and geographical locations.

Each unknown word is run through a series of sentiment indicator tests that aim at identifying in it at least one possible **sentiment stem** - the longest subpart of the word with a known (+), (N), or (-) prior polarity. An unknown word such as *healthcare-related*$^{(?)}$ can be traced back to the stems *health*$^{(N)(+)}$, *care*$^{(+)}$, *healthcare*$^{(+)}$, or *relate(d)*$^{(N)}$ which are all more likely to be found in the lexica, for example. Note that the term 'stem' here does not have its usual linguistic meaning but rather means 'known labelled form', whether complex or not.

We employ a classifier society of five rule-driven classifiers that require no training data. Each classifier adopts a specific analytical strategy within a specific window inside the unknown word, and outputs three separate polarity scores based on the number of stems founds (S_{pos}, S_{ntr}, S_{neg}) (initially 1). The score for polarity p for unknown word w is calculated as follows:

$$(1) \quad scr(p) = \Phi_p \frac{L_s}{L_w} \frac{1}{S_w} \frac{S_p}{S_{pos} + S_{ntr} + S_{neg}}$$

where Φ_p = polarity coefficient (default 1)
L_s = # of characters in the stem
L_w = # of characters in w
S_w = # of punctuation splits in w

Polarity coefficients balance the stem counts: in particular, (N) polarity is suppressed by a Φ_{ntr} of < 1

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 109–112,
Columbus, Ohio, USA, June 2008.

because (N) stem counts dominate in the vast majority of cases. L_s reflects differing degrees of reliability between short and long stems in order to favour the latter. S_w targets the increased ambiguity potential in longer punctuated constructs. The highest-scoring polarity across the three polarity scores from each of the five classifiers is assigned to *w*.

Conversion [A]. It is generally beneficial to impose word class polarity constraints in the lexicon (e.g. *[smart]*$^{(+)}$ ADJ vs. *[smart]*$^{(-)}$ V). Due to creative lexical conversion across word classes, hard constraints can however become counterproductive. The first classifier estimates zero-derived paronyms by retagging the unknown word with different POS tags and requerying the lexica.

Morphological Derivation [B]. The second classifier relies on regular derivational (e.g. *-ism*, *-ify*, *-esque*) and inflectional (e.g. *-est*, *-s*) morphology. The unknown word is transformed incrementally into shorter paronymic aliases using pure affixes and (pseudo and neo-classical) combining forms. A recursive derivation table of find/replace pairs is used to model individual affixes and their regular spelling alternations (e.g. *-pping*$\triangleright$*p*; *-ation*$\triangleright$*e*; *-iness*$\triangleright$*y*; *-some*$\triangleright\emptyset$; *re-*$\triangleright\emptyset$). Polarity reversal affixes such as *-less*$^{(N)[\neg]}$ and *not-so-*$^{(N)[\neg]}$ are supported. The table is traversed until a non-neutral sentiment (NB. not morphological) stem is found. Prefixes are matched first. Note that the prefix-driven configuration we have adopted is an approximation to a (theoretically) full morphemic parse. The derivation for *antirationalistic*$^{(?)}$, for example, first matches the prefix *anti-*$^{(N)[\neg]}$, and then truncates the immediate constituent *rationalistic*$^{(?)}$ incrementally until a sentiment stem (e.g. *rational*$^{(N)(+)}$) is encountered. The polarity reversal prefix *anti-*$^{(N)[\neg]}$ then reverses the polarity of the stem: hence, *antirationalistic*$^{(?)}\triangleright$*rationalistic*$^{(?)}\triangleright$*rationalist*$^{(?)}\triangleright$ *rational*$^{(+)}\triangleright$*antirationalistic*$^{(-)}$. 322 (N) and 67 [¬] prefixes, and 174 (N) and 28 [¬] suffixes were used.

Affix-like Polarity Markers [C]. Beyond the realm of pure morphemes, many non-neutral sentiment markers exist. Examples include prefix-like elements in *well-built*$^{(+)}$, *badly-behaving*$^{(-)}$, and *strange-looking*$^{(-)}$; and suffix-like ones in *rat-infested*$^{(-)}$, *burglarproof*$^{(+)}$, and *fruit-loving*$^{(+)}$. Because the polarity of a non-neutral marker commonly dominates over its host, the marker propagates its sentiment across the entire word. Hence, a full-blown derivation is not required (e.g. *easy-to-install*$^{(?)}\triangleright$*easy-to-install*$^{(+)}$; *necrophobia*$^{(?)}\triangleright$*necrophobia*$^{(-)}$). We experimented with 756 productive prefixes and 640 suffixes derived from hyphenated tokens with a frequency of $\geq$ 20 amongst 406253 words mined from the WAC 2006 corpus[1]. Sentiment markers are captured through simple regular expression-based longest-first matching.

Syllables [D]. We next split unknown words into individual syllables based on syllabic onset, nucleus, and coda boundaries obtained from our own rule-based syllable chunker. Starting with the longest, the resultant monosyllabic and permutative order-preserving polysyllabic words are used as aliases to search the lexica. Aliases not found in our lexica are treated as (N). Consider the unknown word *freedomfortibet*$^{(?)}$. In the syllabified set of singular syllables {*free, dom, for, ti, bet*} and combinatory permutations such as {*free.dom, dom.ti, for.ti.bet,* ...}, *free* or *free.dom* are identified as (+) while all others become (N). Depending on the Φ_{ntr} value, *free.dom.for.ti.bet*$^{(?)}$ can then be tagged as (+) due to the (+) stem(s). Note that cruder substring-based methods can always be used instead. However, a syllabic approach shrinks the search space and ensures the phonotactic well-formedness of the aliases.

Shallow Parsing [E]. At a deepest level, we approximate the internal quasi-syntactic structure of unknown words that can be split based on various punctuation characters. Both exotic phrasal nonce forms (e.g. *kill-the-monster-if-it's-green-and-ugly*$^{(-)}$) and simpler punctuated compounds (e.g. *butt-ugly*$^{(-)}$, *girl-friend*$^{(+)}$) follow observable syntactic hierarchies amongst their subconstituents. Similar rankings can be postulated for sentiment. Since not all constituents are of equal importance, the sentiment salience of each subconstituent is estimated using a subset of the grammatical polarity rankings and compositional processes proposed in Moilanen and Pulman (2007). The unknown word is split into a virtual sentence and POS-tagged[2]. The rightmost subconstituent

[1] Fletcher, W. H. (2007). English Web Corpus 2006. www.webascorpus.org/searchwc.html

[2] Connexor Machinese Syntax 3.8. www.connexor.com

Table 1: Average (A)ccuracy, *k*appa, and error distribution against ANN-2 and ANN-3

		ALL POL		NON-NTR		¬LAZY	ERROR DISTRIBUTION		
Classifier	Φ_{ntr}	A	*k*	A	*k*	A	FATAL	GREEDY	LAZY
[A] CONVERSION	.2	76.70	.03	96.88	.94	99.53	0.08	2.47	97.44
[B] DERIVATION	.8	74.15	.11	80.05	.59	93.90	2.81	22.86	74.33
[C] AFFIX MARKERS	.2	72.33	.21	77.93	.55	88.05	6.10	39.07	54.83
[D] SYLLABLES	.8	69.55	.23	71.88	.45	82.75	9.37	48.62	42.01
[E] PARSING	.7	64.33	.25	79.09	.59	73.50	9.03	65.40	25.57
ALL		63.20	.28	80.61	.61	70.20	9.49	71.41	19.10
ALL ¬UNSURE		64.60	.28	82.19	.64	69.71	7.43	77.95	14.62

in the word is expanded incrementally leftwards by combining it with its left neighbour until the whole word has been analysed. At each step, the sentiment grammar in *idem.* controls (i) non-neutral sentiment propagation and (ii) polarity conflict resolution to calculate a global polarity for the current composite construct. The unknown word *help-children-in-distress*$^{(?)}$ follows the sequence $_{N}$*:[*$\underline{\textit{distress}}^{(-)}$*]*$^{(-)}$$\triangleright_{PP}$*:[*$\underline{\textit{in}}^{(N)}$*distress*$^{(-)}$*]*$^{(-)}$$\triangleright_{NP}$*:[children*$^{(N)}$*[*$\underline{\textit{in}}$ $\underline{\textit{distress}}$*]*$^{(-)}$*]*$^{(-)}$$\triangleright_{VP}$*:[*$\underline{\textit{help}}^{(+)}$*[children in distress]*$^{(-)}$*]*$^{(+)}$, and is thus tagged as (+).

3 Evaluation

We compiled a dataset of 2000 infrequent words containing hapax legomena from the BNC[3] and "junk" entries from the WAC 2006 corpus (Footnote 1). The dataset contains simple, medium-complexity, and extreme complex cases covering single words, (non-)hyphenated compounds, nonce forms, and spelling anomalies (e.g. *anti-neo-nazi-initiatives*, *funny-because-its-true*, and *s'gonnacostyaguvna*). Three human annotators classified the entries as (+), (−), or (N) (with an optional UNSURE tag) with the following distribution:

(2)

Human	(+)	(N)	(−)	UNSURE
ANN-1	24.55	53.45	22	11.75
ANN-2	12.60	68.60	18.80	10.85
ANN-3	5.25	84.55	10.20	0.65

We report results using all polarities (ALL-POL) and non-neutral polarities (NON-NTR) resulting in average pairwise inter-annotator Kappa scores of .40 (ALL-POL) and .74 (NON-NTR), or .48 (ALL-POL) and .83 (NON-NTR) without UNSURE cases. We used ANN-1's data to adjust the Φ_{ntr} coefficients of individual classifiers, and evaluated the system against both ANN-2 and ANN-3. The average scores between ANN-2 and ANN-3 are given in Table 1.

Since even human polarity judgements become fuzzier near the neutral/non-neutral boundary due to differing personal degrees of sensitivity towards neutrality (cf. low (N) agreement in Ex. 2; Andreevskaia and Bergler (2006)), not all classification errors are equal for classifying a (+) case as (N) is more tolerable than classifying it as (−), for example. We therefore found it useful to characterise three distinct disagreement classes between human *H* and machine *M* encompassing FATAL ($H^{(+)}M^{(-)}$ or $H^{(-)}M^{(+)}$), GREEDY ($H^{(N)}M^{(-)}$ or $H^{(N)}M^{(+)}$), and LAZY ($H^{(+)}M^{(N)}$ or $H^{(-)}M^{(N)}$) cases.

The classifiers generally mimic human judgements in that accuracy is much lower in the three-way classification task - a pattern concurring with past observations (cf. Esuli and Sebastiani (2006); Andreevskaia and Bergler (2006)). Crucially, FATAL errors remain below 10% throughout. Further advances can be made by fine-tuning the Φ_{ntr} coefficients, and by learning weights for individual classifiers which can currently mask each other and suppress the correct analysis when run collectively.

4 Related Work

Past research in Sentiment Tagging (cf. Opinion Mining, Sentiment Extraction) has targeted classification along the subjectivity, sentiment polarity, and strength/degree dimensions towards a common goal

[3] Kilgarriff, A. (1995). BNC database and word frequency lists. www.kilgarriff.co.uk/bnc-readme.html

of (semi-)automatic compilation of sentiment lexica. The utility of word-internal sentiment clues has not yet been explored in the area, to our knowledge.

Lexicographic Methods. Static dictionary-/thesaurus-based methods rely on the lexical-semantic knowledge and glosses in existing lexicographic resources alongside known non-neutral seed words. The semi-supervised learning method in Esuli and Sebastiani (2005) involves constructing a training set of non-neutral words using WordNet synsets, glosses and examples by iteratively adding syn- and antonyms to it and learning a term classifier on the glosses of the terms in the training set. Esuli and Sebastiani (2006) used the method to cover objective (N) cases. Kamps et al. (2004) developed a graph-theoretic model of WordNet's synonymy relations to determine the polarity of adjectives based on their distance to words indicative of subjective evaluation, potency, and activity dimensions. Takamura et al. (2005) apply to words' polarities a physical spin model inspired by the behaviour of electrons with a (+) or (−) direction, and an iterative term-neighbourhood matrix which models magnetisation. Non-neutral adjectives were extracted from WordNet and assigned fuzzy sentiment category membership/centrality scores and tags in Andreevskaia and Bergler (2006).

Corpus-based Methods. Lexicographic methods are necessarily confined within the underlying resources. Much greater coverage can be had with syntactic or co-occurrence patterns across large corpora. Hatzivassiloglou and McKeown (1997) clustered adjectives into (+) and (−) sets based on conjunction constructions, weighted similarity graphs, minimum-cuts, supervised learning, and clustering. A popular, more general unsupervised method was introduced in Turney and Littman (2003) which induces the polarity of a word from its Pointwise Mutual Information (PMI) or Latent Semantic Analysis (LSA) scores obtained from a web search engine against a few paradigmatic (+) and (−) seeds. Kaji and Kitsuregawa (2007) describe a method for harvesting sentiment words from non-neutral sentences extracted from Japanese web documents based on structural layout clues. Strong adjectival subjectivity clues were mined in Wiebe (2000) with a distributional similarity-based word clustering method seeded by hand-labelled annotation. Riloff et al. (2003) mined subjective nouns from unannotated texts with two bootstrapping algorithms that exploit lexico-syntactic extraction patterns and manually-selected subjective seeds.

5 Conclusion

In this study of unknown words in the domain of sentiment analysis, we presented five methods for guessing the prior polarities of unknown words based on known sentiment carriers. The evaluation results, which mirror human sentiment judgements, indicate that the methods can account for many unknown words, and that over- and insensitivity towards neutral polarity is the main source of errors.

References

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

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the Semantic Orientation of Terms through Gloss Classification. In *Proceedings of CIKM 2005*.

Andrea Esuli and Fabrizio Sebastiani. 2006. Determining Term Subjectivity and Term Orientation for Opinion Mining. In *Proceedings of EACL 2006*.

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

Jaap Kamps, Maarten Marx, Robert J. Mokken and Maarten de Rijke 2004. Using WordNet to Measure Semantic Orientations of Adjectives. In *Proceedings of LREC 2004*.

Nabuhiro Kaji and Masaru Kitsuregawa. 2007. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. In *Proceedings of EMNLP-CoNLL 2007*.

Karo Moilanen and Stephen Pulman. 2007. Sentiment Composition. In *Proceedings of RANLP 2007*.

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

Hiroya Takamura, Takashi Inui and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of ACL 2005*.

Peter Turney and Michael Littman. 2003. Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Transactions on Information Systems*, October, 21(4): 315–46.

Janyce Wiebe. 2000. Learning Subjective Adjectives from Corpora. In *Proceedings of AAAI 2000*.

Kernels on Linguistic Structures for Answer Extraction

Alessandro Moschitti and **Silvia Quarteroni**
DISI, University of Trento
Via Sommarive 14
38100 POVO (TN) - Italy
{moschitti,silviaq}@disi.unitn.it

Abstract

Natural Language Processing (NLP) for Information Retrieval has always been an interesting and challenging research area. Despite the high expectations, most of the results indicate that successfully using NLP is very complex. In this paper, we show how Support Vector Machines along with kernel functions can effectively represent syntax and semantics. Our experiments on question/answer classification show that the above models highly improve on bag-of-words on a TREC dataset.

1 Introduction

Question Answering (QA) is an IR task where the major complexity resides in question processing and answer extraction (Chen et al., 2006; Collins-Thompson et al., 2004) rather than document retrieval (a step usually carried out by off-the shelf IR engines). In question processing, useful information is gathered from the question and a query is created. This is submitted to an IR module, which provides a ranked list of relevant documents. From these, the QA system extracts one or more candidate answers, which can then be re-ranked following various criteria. Although typical methods are based exclusively on word similarity between query and answer, recent work, e.g. (Shen and Lapata, 2007) has shown that shallow semantic information in the form of predicate argument structures (PASs) improves the automatic detection of correct answers to a target question. In (Moschitti et al., 2007), we proposed the Shallow Semantic Tree Kernel (SSTK) designed to encode PASs[1] in SVMs.

[1] in PropBank format, (www.cis.upenn.edu/~ace).

In this paper, similarly to our previous approach, we design an SVM-based answer extractor, that selects the correct answers from those provided by a basic QA system by applying tree kernel technology. However, we also provide: (i) a new kernel to process PASs based on the partial tree kernel algorithm (PAS-PTK), which is highly more efficient and more accurate than the SSTK and (ii) a new kernel called Part of Speech sequence kernel (POSSK), which proves very accurate to represent shallow syntactic information in the learning algorithm.

To experiment with our models, we built two different corpora, WEB-QA and TREC-QA by using the description questions from TREC 2001 (Voorhees, 2001) and annotating the answers retrieved from Web resp. TREC data (available at disi.unitn.it/~silviaq). Comparative experiments with re-ranking models of increasing complexity show that: (a) PAS-PTK is far more efficient and effective than SSTK, (b) POSSK provides a remarkable further improvement on previous models. Finally, our experiments on the TREC-QA dataset, un-biased by the presence of typical Web phrasings, show that BOW is inadequate to learn relations between questions and answers. This is the reason why our kernels on linguistic structures improve it by 63%, which is a remarkable result for an IR task (Allan, 2000).

2 Kernels for Q/A Classification

The design of an answer extractor basically depends on the design of a classifier that decides if an answer correctly responds to the target question. We design a classifier based on SVMs and different kernels applied to several forms of question and answer

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 113–116,
Columbus, Ohio, USA, June 2008.

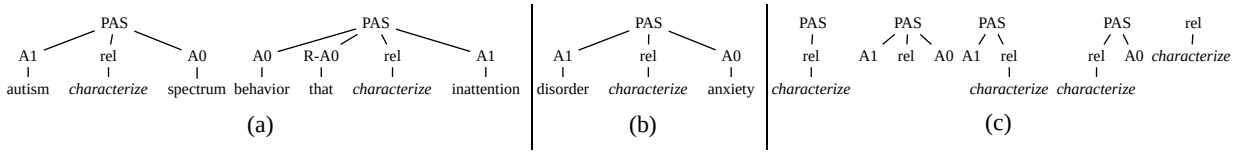


Figure 1: Compact PAS-PTK structures of s_1 (a) and s_2 (b) and some fragments they have in common as produced by the PTK (c). Arguments are replaced with their most important word (or semantic head) to reduce data sparseness.

representations:
(1) linear kernels on the bag-of-words (BOW) or bag-of-POS-tags (POS) features,
(2) the String Kernel (SK) (Shawe-Taylor and Cristianini, 2004) on word sequences (WSK) and POS-tag sequences (POSSK),
(3) the Syntactic Tree Kernel (STK) (Collins and Duffy, 2002) on syntactic parse trees (PTs),
(4) the Shallow Semantic Tree Kernel (SSTK) (Moschitti et al., 2007) and the Partial Tree Kernel (PTK) (Moschitti, 2006) on PASs.

In particular, POS-tag sequences and PAS trees used with SK and PTK yield to two innovative kernels, i.e. POSSK and PAS-PTK[2]. In the next sections, we describe in more detail the data structures on which we applied the above kernels.

2.1 Syntactic Structures

The POSSK is obtained by applying the String Kernel on the sequence of POS-tags of a question or a answer. For example, given sentence s_0: *What is autism?*, the associated POS sequence is *WP AUX NN ?* and some of the substrings extracted by POSSK are *WP NN* or *WP AUX*. A more complete structure is the full parse tree (PT) of the sentence, that constitutes the input of the STK. For instance, the STK accepts the syntactic parse: *(SBARQ (WHNP (WP What))(SQ (VP (AUX is)(NP (NN autism))))(. ?))*.

2.2 Semantic Structures

The intuition behind our semantic representation is the idea that when we ignore the answer to a definition question we check whether such answer is formulated as a "typical" definition and whether answers defining similar concepts are expressed in a similar way.

To take advantage of semantic representations, we work with two types of semantic structures; first, the Word Sequence Kernel applied to both question and answer; given s_0, sample substrings are: *What is autism*, *What is*, *What autism*, *is autism*, etc. Then, two PAS-based trees: Shallow Semantic Trees for SSTK and Shallow Semantic Trees for PTK, both based on PropBank structures (Kingsbury and Palmer, 2002) are automatically generated by our SRL system (Moschitti et al., 2005). As an example, let us consider an automatically annotated sentence from our TREC-QA corpus:
s_1: *[$_{A1}$ Autism] is [$_{rel}$ characterized] [$_{A0}$ by a broad spectrum of behavior] [$_{R-A0}$ that] [$_{rel}$includes] [$_{A1}$ extreme inattention to surroundings and hypersensitivity to sound and other stimuli].*
Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.
s_2: *[$_{A1}$ Panic disorder] is [$_{rel}$ characterized] [$_{A0}$ by unrealistic or excessive anxiety].*
It can be observed here that, although autism is a different disease from panic disorder, the structure of both definitions and the latent semantics they contain (inherent to behavior, disorder, anxiety) are similar. So for instance, s_2 appears as a definition even to someone who only knows what the definition of autism looks like.

The above annotation can be compactly represented by predicate argument structure trees (PASs) such as those in Figure 1. Here, we can notice that the semantic similarity between sentences is explicitly visible in terms of common fragments extracted by the PTK from their respective PASs. Instead, the similar PAS-SSTK representation in (Moschitti et al., 2007) does not take argument order into account, thus it fails to capture the linguistic rationale expressed above. Moreover, it is much heavier, causing large memory occupancy and, as shown by our experiments, much longer processing time.

[2]For example, let PTK$(t_1, t_2) = \phi(t_1) \cdot \phi(t_2)$, where t_1 and t_2 are two syntactic parse trees. If we map t_1 and t_2 into two new shallow semantic trees s_1 and s_2 with a mapping $\phi_M(\cdot)$, we obtain: PTK$(s_1, s_2) = \phi(s_1) \cdot \phi(s_2) = \phi(\phi_M(t_1)) \cdot \phi(\phi_M(t_2)) = \phi'(t_1) \cdot \phi'(t_2)$=PAS-PTK$(t_1, t_2)$, which is a noticeably different kernel induced by the mapping $\phi' = \phi \circ \phi_M$.

3 Experiments

In our experiments we show that (a) the PAS-PTK shallow semantic tree kernel is more efficient and effective than the SSTK proposed in (Moschitti et al., 2007), and (b) our POSSK jointly used with PAS-PTK and STK greatly improves on BOW.

3.1 Experimental Setup

In our experiments, we implemented the BOW and POS kernels, WSK, POSSK, STK (on syntactic PTs derived automatically with Charniak's parser), SSTK and PTK (on PASs derived automatically with our SRL system) as well as their combinations in SVM-light-TK[3]. Since answers often contain more than one PAS (see Figure 1), we sum PTK (or SSTK) applied to all pairs $P_1 \times P_2$, P_1 and P_2 being the sets of PASs of the first two answers.

The experimental datasets were created by submitting the 138 TREC 2001 test questions labeled as "description" in (Li and Roth, 2002) to our basic QA system, YourQA (Quarteroni and Manandhar, 2008) and by gathering the top 20 answer paragraphs.

YourQA was run on two sources: Web documents by exploiting Google (`code.google.com/apis/`) and the AQUAINT data used for TREC'07 (`trec.nist.gov/data/qa`) by exploiting Lucene (`lucene.apache.org`), yielding two different corpora: WEB-QA and TREC-QA. Each sentence of the returned paragraphs was manually evaluated based on whether it contained a correct answer to the corresponding question. To simplify our task, we isolated for each paragraph the sentence with the maximal judgment (such as s_1 and s_2 in Sec. 2.2) and labeled it as positive if it answered the question either concisely or with noise, negative otherwise. The resulting WEB-QA corpus contains 1309 sentences, 416 of which positive; the TREC-QA corpus contains 2256 sentences, 261 of which positive.

3.2 Results

In a first experiment, we compared the learning and classification efficiency of SVMs on PASs by applying either solely PAS-SSTK or solely PAS-PTK on the WEB-QA and TREC-QA sets. We divided the training data in 9 bins of increasing size (with a step of 200) and measured the training and test time[4] for each bin. Figure 2 shows that in both the test and training phases, PTK is much faster than SSTK. In training, PTK is 40 times faster, enabling the experimentation of SVMs with large datasets. This difference is due to the combination of our lighter semantic structures and the PTK's ability to extract from these at least the same information that SSTK derives from much larger structures.

Further interesting experiments regard the accu-

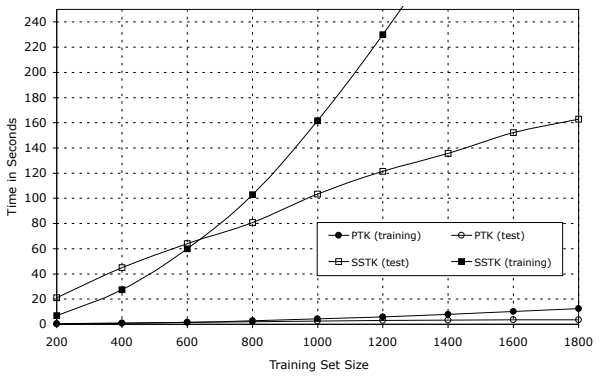


Figure 2: Efficiency of PTK and SSTK

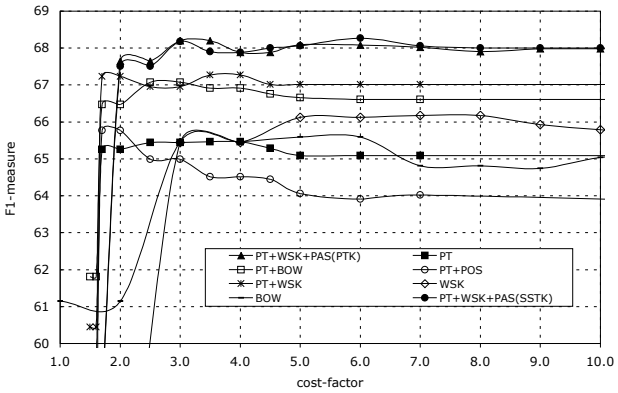


Figure 3: Impact of different kernels on WEB-QA

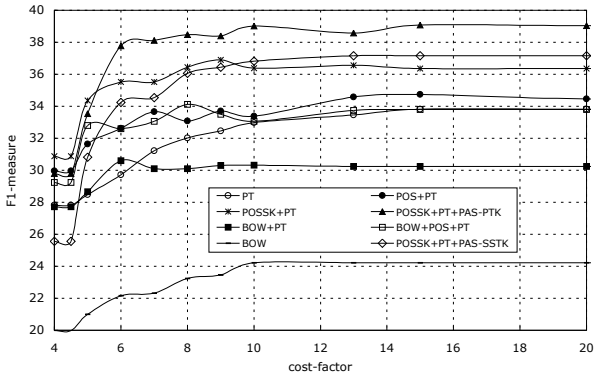


Figure 4: Impact of different kernels on TREC-QA

[3]Toolkit available at `dit.unitn.it/moschitti/`, based on SVM-light (Joachims, 1999)

[4]Processing time in seconds of a Mac-Book Pro 2.4 Ghz.

racy tests of different kernels and some of their most promising combinations. As a kernel operator, we applied the sum between kernels[5] that yields the joint feature space of the individual kernels (Shawe-Taylor and Cristianini, 2004).

Figure 3 shows the F1-plots of several kernels according to different cost-factor values (i.e. different Precision/Recall rates). Each F1 value is the average of 5 fold cross-validation. We note that (a) BOW achieves very high accuracy, comparable to the one produced by PT; (b) the BOW+PT combination improves on both single models; (c) WSK improves on BOW and it is enhanced by WSK+PT, demonstrating that word sequences and PTs are very relevant for this task; (d) both PAS-SSTK and PAS-PTK improve on previous models yielding the highest result.

The high accuracy of BOW is surprising as support vectors are compared with test examples which are in general different (there are no questions shared between training and test set). The explanation resides in the fact that WEB-QA contains common BOW patterns due to typical Web phrasings, e.g. `Learn more about X`, that facilitate the detection of incorrect answers.

Hence, to have un-biased results, we experimented with the TREC corpus which is cleaner from a linguistic viewpoint and also more complex from a QA perspective. A comparative analysis of Figure 4 suggests that: (a) the F1 of all models is much lower than for the WEB-QA dataset; (b) BOW denotes the lowest accuracy; (c) POS combined with PT improves on PT; (d) POSSK+PT improves on POS+PT; (f) finally, PAS adds further information as the best model is POSSK+PT+PAS-PTK (or PAS-SSTK).

4 Conclusions

With respect to our previous findings, experimenting with TREC-QA allowed us to show that BOW is not relevant to learn re-ranking functions from examples; indeed, while it is useful to establish an initial ranking by measuring the similarity between question and answer, BOW is almost irrelevant to grasp typical rules that suggest if a description is valid or not. Moreover, using the new POSSK and PAS-PTK kernels provides an improvement of 5 absolute percent points wrt our previous work.

Finally, error analysis revealed that PAS-PTK can provide patterns like `A1(X) R-A1(that) rel(result) A1(Y)` and `A1(X) rel(characterize) A0(Y)`, where `X` and `Y` need not necessarily be matched.

Acknowledgments

This work was partly supported by the FP6 IST LUNA project (contract No. 33549) and by the European Commission Marie Curie Excellence Grant for the ADAMACH project (contract No. 022593).

References

J. Allan. 2000. Natural language processing for information retrieval. In *Proceedings of NAACL/ANLP (tutorial notes)*.

Y. Chen, M. Zhou, and S. Wang. 2006. Reranking answers from definitional QA using language models. In *ACL'06*.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL'02*.

K. Collins-Thompson, J. Callan, E. Terra, and C. L.A. Clarke. 2004. The effect of document retrieval quality on factoid QA performance. In *SIGIR'04*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *LREC'02*.

X. Li and D. Roth. 2002. Learning question classifiers. In *ACL'02*.

A. Moschitti, B. Coppola, A. Giuglea, and R. Basili. 2005. Hierarchical semantic role labeling. In *CoNLL 2005 shared task*.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *ACL'07*.

A. Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML'06*.

S. Quarteroni and S. Manandhar. 2008. Designing an interactive open domain question answering system. *Journ. of Nat. Lang. Eng. (in press)*.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*.

E. M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *TREC'01*.

[5] All adding kernels are normalized to have a similarity score between 0 and 1, i.e. $K'(X_1, X_2) = \frac{K(X_1,X_2)}{\sqrt{K(X_1,X_1)\times K(X_2,X_2)}}$.

Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin
Center for Computational Learning Systems
Columbia University
New York, NY 10115 USA
{ryanr,rambow,habash,mdiab,rudin}@ccls.columbia.edu,

Abstract

We investigate the tasks of general morphological tagging, diacritization, and lemmatization for Arabic. We show that for all tasks we consider, both modeling the lexeme explicitly, and retuning the weights of individual classifiers for the specific task, improve the performance.

1 Previous Work

Arabic is a morphologically rich language: in our training corpus of about 288,000 words we find 3279 distinct morphological tags, with up to 100,000 possible tags.[1] Because of the large number of tags, it is clear that morphological tagging cannot be construed as a simple classification task. Hajič (2000) is the first to use a dictionary as a source of possible morphological analyses (and hence tags) for an inflected word form. He redefines the tagging task as a choice among the tags proposed by the dictionary, using a log-linear model trained on specific ambiguity classes for individual morphological features. Hajič et al. (2005) implement the approach of Hajič (2000) for Arabic. In previous work, we follow the same approach (Habash and Rambow, 2005), using SVM-classifiers for individual morphological features and a simple combining scheme for choosing among competing analyses proposed by the dictionary. Since the dictionary we use, BAMA (Buckwalter, 2004), also includes diacritics (orthographic marks not usually written), we extend this approach to the diacritization task in (Habash and Rambow, 2007). The work presented in this paper differs from this previous work in that (a) we introduce a new task for Arabic, namely **lemmatization**; (b) we use an explicit modeling of lexemes as a component in all tasks discussed in this paper (morphological tagging, diacritization, and lemmatization); and (c) we tune the weights of the feature classifiers on a tuning corpus (different tuning for different tasks).

[1]This work was funded under the DARPA GALE, program, contract HR0011-06-C-0023. We thank several anonymous reviewers for helpful comments. A longer version of this paper is available as a technical report.

2 Morphological Disambiguation Tasks for Arabic

We define the task of **morphological tagging** as choosing an inflectional morphological tag (in this paper, the term "morphological tagging" never refers to derivational morphology). The morphology of an Arabic word can be described by the 14 (nearly) orthogonal features shown in Figure 1. For different tasks, different subsets may be useful: for example, when translating into a language without case, we may want to omit the case feature. For the experiments we discuss in this paper, we investigate three variants of the morphological tagging tasks: **MorphPOS** (determining the feature POS, which is the core part-of-speech – verb, noun, adjective, etc.); **MorphPart** (determining the set of the first ten basic morphological features listed in Figure 1); and **MorphAll** (determining the full inflectional morphological tag, i.e., all 14 features).

The task of **diacritization** involves adding diacritics (short vowels, gemination marker shadda, and indefiniteness marker nunation) to the standard written form. We have two variants of the diacritization

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 117–120,
Columbus, Ohio, USA, June 2008.

Feature name	Explanation
POS	Simple part-of-speech
CNJ	Presence of a conjunction clitic
PRT	Presence of a particle clitic
PRO	Presence of a pronominal clitic
DET	Presence of the definite determiner
GEN	Gender
NUM	Number
PER	Person
VOX	Voice
ASP	Aspect
MOD	Mood
NUN	Presence of nunation (indefiniteness marker)
CON	Construct state (head of a genitive construction)
CAS	Case

Figure 1: List of (inflectional) morphological features used in our system; the first ten are features which (roughly) can be determined with higher accuracy since they rely less on syntactic context and more on visible inflectional morphology

tasks: **DiacFull** (predicting all diacritics of a given word), which relates to lexeme choice and morphology tagging, and **DiacPart** (predicting all diacritics of a given word except those associated with the final letter), which relates largely to lexeme choice.

Lemmatization (**LexChoice**) for Arabic has not been discussed in the literature to our knowledge. A **lexeme** is an abstraction over a set of inflected word forms, and it is usually represented by its **citation form**, also called **lemma**.

Finally, **AllChoice** is the combined task of choosing all inflectional and lexemic aspects of a word in context.

This gives us a total of seven tasks. **AllChoice** is the hardest of our tasks, since it subsumes all other tasks. **MorphAll** is the hardest of the three morphological tagging tasks, subsuming **MorphPart** and **MorphPOS**, and **DiacFull** is the hardest lexical task, subsuming **DiacPart**, which in turn subsumes **LexChoice**. However, **MorphAll** and **DiacFull** are (in general) orthogonal, since **MorphAll** has no lexemic component, while **DiacFull** does.

3 Our System

Our system, MADA, makes use of 19 orthogonal features to select, for each word, a proper **analysis** from a list of potential analyses provided by the BAMA dictionary. The BAMA analysis which matches the most of the predicted features wins; the weighting of the features is one of the topics of this paper. These 19 features consist of the 14 morphological features shown in Figure 1, which MADA predicts using 14 distinct Support Vector Machines trained on ATB3-Train (as defined by Zitouni et al. (2006)). In addition, MADA uses five additional features. **Spellmatch** determines whether the diacritized form of the suggested analysis and the input word match if both are stripped of all of their diacritics. This is useful because sometimes BAMA suggests analyses which imply a different spelling of the undiacritized word, but these analyses are often incorrect. **Isdefault** identifies those analyses that are the default output of BAMA (typically, these are guesses that the word in question is a proper noun); these analyses are less likely to be correct than others suggested by BAMA. MADA can derive the values of **Spellmatch** and **Isdefault** by direct examination of the analysis in question, and no predictive model is needed. The fourteen morphological features plus **Spellmatch** and **Isdefault** form a feature collection that is entirely based on morphological (rather than lexemic) features; we refer to this collection as **BASE-16**. **UnigramDiac** and **UnigramLex** are unigram models of the surface diacritized form and the lexeme respectively, and contain lexical information. We also build 4-gram lexeme models using an open-vocabulary language model with Kneser-Ney smoothing, by means of the SRILM toolkit (Stolcke, 2002). The model is trained on the same corpus used to train the other classifiers, ATB3-Train. (We also tested other n-gram models, and found that a 4-gram lexeme model outperforms the other orders with $n \leq 5$, although the improvement over the trigram and 5-gram models was less than 0.01%.) The 4-gram model, on its own, correctly selects the lexeme of words in ATB3-DevTest 94.1% of the time. The 4-gram lexeme model was incorporated into our system as a full feature (**NGRAM**). We refer to the feature set consisting of **BASE-16** plus the two unigram mod-

els and **NGRAM** as **FULL-19**.

Optimizing the feature weights is a machine learning task. To provide learning data for this task, we take the ATB3-DevTest data set and divide it into two sections; the first half (∼26K words) is used for tuning the weights and the second half (∼25K words) for testing. In a pre-processing step, each analysis in appended with a set of labels which indicate whether the analysis is correct according to seven different evaluation metrics. These metrics correspond in a one-to-one manner to the seven different disambiguation tasks discussed in Section 2, and we use the task name for the evaluation label. Specifically, the **MorphPOS** label is positive if the analysis has the same POS value as the correct analysis in the gold standard; the **LexChoice** label provides the same information about the lexeme choice. The **MorphPart** label is positive if the analysis agrees with the gold for each of the 10 basic features used by Habash and Rambow (2005). A positive **MorphAll** label requires that the analysis match the gold in all morphological features, i.e., in every feature except the lexeme choice and diacritics. The **DiacFull** label is only positive if the surface diacritics of the analysis match the gold diacritics exactly; **DiacPart** is less strict in that the trailing sequence diacritic markers in each surface diacritic are stripped before the analysis and the gold are compared. Finally, **AllChoice** is only positive if the analysis was one chosen as correct in the gold; this is the strictest form of evaluation, and there can be only one positive **AllChoice** label per word.

In addition to labeling as described in the preceding paragraph, we run MADA on the tuning and test sets. This gives us a set of model predictions for every feature of every word in the tuning and test sets. We use an implementation of a Downhill Simplex Method in many dimensions based on the method developed by Nelder and Mead (1965) to tune the weights applied to each feature. In a given iteration, the Simplex algorithm proposes a set of feature weights. These weights are given to a weight evaluation function; this function determines how effective a particular set of weights is at a given disambiguation task by calculating an **overall score** for the weight set: the number of words in the tuning set that were correctly disambiguated. In order to compute this score, the weight evaluation function examines each proposed analysis for each word in the tuning set. If the analysis and the model prediction for a feature of a given word agree, the **analysis score** for that analysis is incremented by the weight corresponding to that feature. The analysis with the highest analysis score is selected as the proper analysis for that word. If the selected analysis has a positive task label (i.e., it is a good answer for the disambiguation task in question), the overall score for the proposed weight set is incremented. The Simplex algorithm seeks to maximize this overall score (and thus choose the weight set that performs best for a given task).

Once the Simplex algorithm has converged, the optimal feature weights for a given task are known. Our system makes use of these weights to select a correct analysis in the test set. Each analysis of each word is given a score that is the sum of optimal feature weights for features where the model prediction and the analysis agree. The analysis with the highest score is then chosen as the correct analysis for that word. The system can be evaluated simply by comparing the chosen analysis to the gold standard. Since the Simplex weight evaluation function and the system use identical means of scoring analyses, the Simplex algorithm has the potential to find very optimized weights.

4 Experiments

We have three main research hypotheses: (1) Using lexemic features helps in all tasks, but especially in the diacritization and lexeme choice tasks. (2) Tuning the weights helps over using identical weights. (3) Tuning to the task that is evaluated improves over tuning to other tasks. For each of the two feature sets, **BASE-16** and **FULL-19**, we tune the weights using seven **tuning metrics**, producing seven sets of weights. We then evaluate the seven automatically weighted systems using seven **evaluation metrics**. The tuning metrics are identical to the evaluation metrics and they correspond to the seven tasks described in Section 2. Instead of showing 98 results, we show in Figure 2 four results for each of the seven tasks: for both the **BASE-16** and **FULL-19** feature sets, we give the untuned performance, and then the best-performing tuned performance. We indicate which tuning metric provided the best tun-

Task	Baseline	**BASE-16** (Morph Feats Only)			**FULL-19** (All Feats)		
		Not Tuned	Tuned	Tuning metric	Not Tuned	Tuned	Tuning metric
MorphPOS	95.5	95.6	96.0	**MorphAll**	96.0	96.4	**MorphPOS**
MorphPart	93.8	94.1	94.8	**AllChoice**	94.7	95.1	**DiacPart**
MorphAll	83.8	84.0	84.8	**AllChoice**	82.2	85.1	**MorphAll**
LexChoice	85.5	86.6	87.5	**MorphAll**	95.4	96.3	**LexChoice**
DiacPart	85.1	86.4	87.3	**AllChoice**	94.8	95.4	**DiacPart**
DiacFull	76.0	77.1	78.2	**MorphAll**	82.6	86.1	**MorphAll**
AllChoice	73.3	74.5	75.6	**AllChoice**	80.3	83.8	**MorphAll**

Figure 2: Results for morphological tagging tasks (percent correct); the baseline uses only 14 morphological features with identical weights; "Tuning Metric" refers to the tuning metric that produced the best tuned results, as shown in the "Tuned" column

ing performance. The Baseline indicated in Figure 2 uses the 14 morphological features (listed in Figure 1) only, with no tuning (i.e., all 14 features have a weight of 1). The untuned results were determined by also setting almost all feature weights to 1; the only exception is the **Isdefault** feature, which is given a weight of -(8/14) when included in untuned sets. Since this feature is meant to penalize analyses, its value must be negative; we use this particular value so that our results can be readily compared to previous work. All results are the best published results to date on these test sets; for a deeper discussion, see the longer version of this paper which is available as a technical report.

We thus find our three hypotheses confirmed: (1) Using lexemic features reduces error for the morphological tagging tasks (measured on tuned data) by 3% to 11%, but by 36% to 71% for the diacritic and lexeme choice tasks. The highest error reduction is indeed for the lexical choice task. (2) Tuning the weights helps over using identical weights. With only morphological features, we obtain an error reduction of between 4% and 12%; with all features, the error reduction from tuning ranges between 8% and 20%. (3) As for the correlation between tuning task and evaluation task, it turned out that when we use only morphological features, two tuning tasks worked best for all evaluation tasks, namely **MorphAll** and **AllChoice**, thus not confirming our hypothesis. We speculate that in the absence of the lexical features, more features is better (these two tasks are the two hardest tasks for morphological features only). If we add the lexemic features, we do find our hypothesis confirmed, with almost all evaluation tasks performing best when the weights are tuned for that task. In the case of the three exceptions, the differences between the best performance and performance when tuned to the same task are very slight ($< 0.06\%$).

References

Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL'05*, Ann Arbor, MI, USA.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *NAACL HLT 2007 Companion Volume, Short Papers*, Rochester, NY, USA.

Jan Hajič, Otakar Smrž, Tim Buckwalter, and Hubert Jin. 2005. Feature-based tagger of approximations of functional Arabic morphology. In *Proceedings of the Workshop on Treebanks and Linguistic Theories (TLT)*, Barcelona, Spain.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*, Seattle, WA.

J.A Nelder and R Mead. 1965. A simplex method for function minimization. In *Computer Journal*, pages 303–333.

Andreas Stolcke. 2002. Srilm - an extensible language toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Coling-ACL'06*, pages 577–584, Sydney, Australia.

Using Automatically Transcribed Dialogs to Learn User Models in a Spoken Dialog System

Umar Syed
Department of Computer Science
Princeton University
Princeton, NJ 08540, USA
usyed@cs.princeton.edu

Jason D. Williams
Shannon Laboratory
AT&T Labs — Research
Florham Park, NJ 07932, USA
jdw@research.att.com

Abstract

We use an EM algorithm to learn user models in a spoken dialog system. Our method requires automatically transcribed (with ASR) dialog corpora, plus a model of transcription errors, but does not otherwise need any manual transcription effort. We tested our method on a voice-controlled telephone directory application, and show that our learned models better replicate the true distribution of user actions than those trained by simpler methods and are very similar to user models estimated from manually transcribed dialogs.

1 Introduction and Background

When designing a dialog manager for a spoken dialog system, we would ideally like to try different dialog management strategies on the actual user population that will be using the system, and select the one that works best. However, users are typically unwilling to endure this kind of experimentation. The next-best approach is to build a model of user behavior. That way we can experiment with the model as much as we like without troubling actual users.

Of course, for these experiments to be useful, a high-quality user model is needed. The usual method of building a user model is to estimate it from transcribed corpora of human-computer dialogs. However, manually transcribing dialogs is expensive, and consequently these corpora are usually small and sparse. In this work, we propose a method of building user models that does not operate on manually transcribed dialogs, but instead uses dialogs that have been transcribed by an automatic speech recognition (ASR) engine. Since this process is error-prone, we cannot assume that the transcripts will accurately reflect the users' true actions and internal states. To handle this uncertainty, we employ an EM algorithm that treats this information as unobserved data. Although this approach does not directly employ manually transcribed dialogs, it does require a confusion model for the ASR engine, which *is* estimated from manually transcribed dialogs. The key benefit is that the number of manually transcribed dialogs required to estimate an ASR confusion model is much smaller, and is fixed with respect to the complexity of the user model.

Many works have estimated user models from transcribed data (Georgila et al., 2006; Levin et al., 2000; Pietquin, 2004; Schatzmann et al., 2007). Our work is novel in that we do not assume we have access to the correct transcriptions at all, but rather have a model of how errors are made. EM has previously been applied to estimation of user models: (Schatzmann et al., 2007) cast the user's internal state as a complex hidden variable and estimate its transitions using the true user actions with EM. Our work employs EM to infer the model of user actions, not the model of user goal evolution.

2 Method

Before we can estimate a user model, we must define a larger model of human-computer dialogs, of which the user model is just one component. In this section we give a general description of our dialog model; in Section 3 we instantiate the model for a voice-controlled telephone directory.

We adopt a probabilistic dialog model (similar

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 121–124,
Columbus, Ohio, USA, June 2008.

to (Williams and Young, 2007)), depicted schematically as a graphical model in Figure 1. Following the convention for graphical models, we use directed edges to denote conditional dependencies among the variables. In our dialog model, a dialog transcript $\mathbf{x}$ consists of an alternating sequence of system actions and observed user actions: $\mathbf{x} = (S_0, \tilde{A}_0, S_1, \tilde{A}_1, \ldots)$. Here S_t denotes the system action, and $\tilde{A}_t$ the output of the ASR engine when applied to the true user action A_t.

A dialog transcript $\mathbf{x}$ is generated by our model as follows: At each time t, the system action is S_t and the unobserved user state is U_t. The user state indicates the user's hidden goal and relevant dialog history which, due to ASR confusions, is known with certainty only to the user. Conditioned on (S_t, U_t), the user draws an unobserved action A_t from a distribution $\Pr(A_t \mid S_t, U_t; \theta)$ parameterized by an unknown parameter θ. For each user action A_t, the ASR engine produces a hypothesis $\tilde{A}_t$ of what the user said, drawn from a distribution $\Pr(\tilde{A}_t \mid A_t)$, which is the ASR confusion model. The user state U_t is updated to U_{t+1} according to a deterministic distribution $\Pr(U_{t+1} \mid S_{t+1}, U_t, A_t, \tilde{A}_t)$. The system outputs the next system action S_{t+1} according to its dialog management policy. Concretely, the values of S_t, U_t, A_t and $\tilde{A}_t$ are all assumed to belong to finite sets, and so all the conditional distributions in our model are multinomials. Hence θ is a vector that parameterizes the user model according to $\Pr(A_t = a \mid S_t = s, U_t = u; \theta) = \theta_{asu}$.

The problem we are interested in is estimating θ given the set of dialog transcripts $\mathcal{X}$, $\Pr(\tilde{A}_t \mid A_t)$ and $\Pr(U_{t+1} \mid S_{t+1}, U_t, A_t, \tilde{A}_t)$. Here, we assume that $\Pr(\tilde{A}_t \mid A_t)$ is relatively straightforward to estimate: for example, ASR models that rely a simple confusion rate and uniform substitutions (which can be estimated from small number of transcriptions) have been used to train dialog systems which outperform traditional systems (Thomson et al., 2007). Further, $\Pr(U_{t+1} \mid S_{t+1}, U_t, A_t, \tilde{A}_t)$ is often deterministic and tracks dialog history relevant to action selection — for example, whether the system correctly or incorrectly confirms a slot value. Here we assume that it can be easily hand-crafted.

Formally, given a set of dialog transcripts $\mathcal{X}$, our goal is find a set of parameters θ^* that maximizes the log-likelihood of the observed data, i.e.,

$$\theta^* = \arg\max_{\theta} \log \Pr(\mathcal{X} \mid \theta)$$

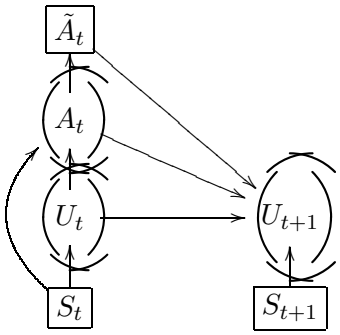


Figure 1: A probabilistic graphical model of a human-computer dialog. The boxed variables are observed; the circled variables are unobserved.

Unfortunately, directly computing θ^* in this equation is intractable. However, we can efficiently approximate θ^* via an expectation-maximization (EM) procedure (Dempster et al., 1977). For a dialog transcript $\mathbf{x}$, let $\mathbf{y}$ be the corresponding sequence of unobserved values: $\mathbf{y} = (U_0, A_0, U_1, A_1, \ldots)$. Let $\mathcal{Y}$ be the set of all sequences of unobserved values corresponding to the data set $\mathcal{X}$. Given an estimate $\theta^{(t-1)}$, a new estimate $\theta^{(t)}$ is produced by

$$\theta^{(t)} = \arg\max_{\theta} E_{\mathcal{Y}} \left[\log \Pr(\mathcal{X}, \mathcal{Y} \mid \theta) \;\middle|\; \mathcal{X}, \theta^{(t-1)} \right]$$

The expectation in this equation is taken over all possible values for $\mathcal{Y}$. Both the expectation and its maximization are easy to compute. This is because our dialog model has a chain-like structure that closely resembles an Hidden Markov Model, so a forward-backward procedure can be employed (Rabiner, 1990). Under fairly mild conditions, the sequence $\theta^{(0)}, \theta^{(1)}, \ldots$ converges to a stationary point estimate of θ^* that is usually a local maximum.

3 Target Application

To test the method, we applied it to a voice-controlled telephone directory. This system is currently in use in a large company with many thousands of employees. Users call the directory system and provide the name of a callee they wish to be connected to. The system then requests additional

information from the user, such as the callee's location and type of phone (office, cell). Here is a small fragment of a typical dialog with the system:

$S_0 =$ `First and last name?`
$A_0 =$ "John Doe" [$\tilde{A}_0 =$ *Jane Roe*]
$S_1 =$ `Jane Roe.  Office or cell?`
$A_1 =$ "No, no, John Doe" [$\tilde{A}_1 =$ *No*]
$S_2 =$ `First and last name?`
...

Because the telephone directory has many names, the number of possible values for A_t, $\tilde{A}_t$, and S_t is potentially very large. To control the size of the model, we first assumed that the user's intended callee does not change during the call, which allows us to group many user actions together into generic placeholders e.g. $A_t =$ `FirstNameLastName`. After doing this, there were a total of 13 possible values for A_t and $\tilde{A}_t$, and 14 values for S_t.

The user state consists of three bits: one bit indicating whether the system has correctly recognized the callee's name, one bit indicating whether the system has correctly recognized the callee's "phone type" (office or cell), and one bit indicating whether the user has said the callee's geographic location (needed for disambiguation when several different people share the same name). The deterministic distribution $\Pr(U_{t+1} \mid S_{t+1}, U_t, A_t, \tilde{A}_t)$ simply updates the user state after each dialog turn in the obvious way. For example, the "name is correct" bit of U_{t+1} is set to 0 when S_{t+1} is a confirmation of a name which doesn't match A_t.

Recall that the user model is a multinomial distribution $\Pr(A_t \mid S_t, U_t; \theta)$ parameterized by a vector θ. Based on the number user actions, system actions, and user states, θ is a vector of $(13-1) \times 14 \times 8 = 1344$ unknown parameters for our target application.

4 Experiments

We conducted two sets of experiments on the telephone directory application, one using simulated data, and the other using dialogs collected from actual users. Both sets of experiments assumed that all the distributions in Figure 1, except the user model, are known. The ASR confusion model was estimated by transcribing 50 randomly chosen dialogs from the training set in Section 4.2 and calculating the frequency with which the ASR engine recognized $\tilde{A}_t$ such that $\tilde{A}_t \neq A_t$. The probabilities $\Pr(\tilde{A}_t \mid A_t)$ were then constructed by assuming that, when the ASR engine makes an error recognizing a user action, it substitutes another randomly chosen action.

4.1 Simulated Data

Recall that, in our parameterization, the user model is $\Pr(A_t = a \mid S_t = s, U_t = u; \theta) = \theta_{asu}$. So in this set of experiments, we chose a reasonable, hand-crafted value for θ, and then generated synthetic dialogs by following the probabilistic process depicted in Figure 1. In this way, we were able to create synthetic training sets of varying sizes, as well as a test set of 1000 dialogs. Each generated dialog $\mathbf{d}$ in each training/test set consisted of a sequence of values for all the observed and unobserved variables: $\mathbf{d} = (S_0, U_0, A_0, \tilde{A}_0, \ldots)$.

For a training/test set $\mathcal{D}$, let $K^{\mathcal{D}}_{asu}$ be the number of times t, in all the dialogs in $\mathcal{D}$, that $A_t = a$, $S_t = s$, and $U_t = u$. Similarly, let $\widetilde{K}^{\mathcal{D}}_{as}$ be the number of times t that $\tilde{A}_t = a$ and $S_t = s$.

For each training set $\mathcal{D}$, we estimated θ using the following three methods:

1. *Manual*: Let θ be the maximum likelihood estimate using manually transcribed data, i.e., $\theta_{asu} = \frac{K^{\mathcal{D}}_{asu}}{\sum_a K^{\mathcal{D}}_{asu}}$.

2. *Automatic*: Let θ be the maximum likelihood estimate using automatically transcribed data, i.e., $\theta_{asu} = \frac{\widetilde{K}^{\mathcal{D}}_{as}}{\sum_a \widetilde{K}^{\mathcal{D}}_{as}}$. This approach ignores transcription errors and assumes that user behavior depends only on the observed data.

3. *EM*: Let θ be the estimate produced by the EM algorithm described in Section 2, which uses the automatically transcribed data and the ASR confusion model.

Now let $\mathcal{D}$ be the test set. We evaluated each user model by calculating the normalized log-likelihood of the model with respect to the *true* user actions in $\mathcal{D}$:

$$\ell(\theta) = \frac{\sum_{a,s,u} K^{\mathcal{D}}_{asu} \log \theta_{asu}}{|\mathcal{D}|}$$

$\ell(\theta)$ is essentially a measure of how well the user model parameterized by θ replicates the distribution

of user actions in the test set. The normalization is to allow for easier comparison across data sets of differing sizes.

We repeated this entire process (generating training and test sets, estimating and evaluating user models) 50 times. The results presented in Figure 2 are the average of those 50 runs. They are also compared to the normalized log-likelihood of the "Truth", which is the actual parameter θ used to generated the data.

The EM method has to estimate a larger number of parameters than the Automatic method (1344 vs. 168). But as Figure 2 shows, after observing enough dialogs, the EM method is able to leverage the hidden user state to learn a better model of user behavior, with an average normalized log-likelihood that falls about halfway between that of the models produced by the Automatic and Manual methods.

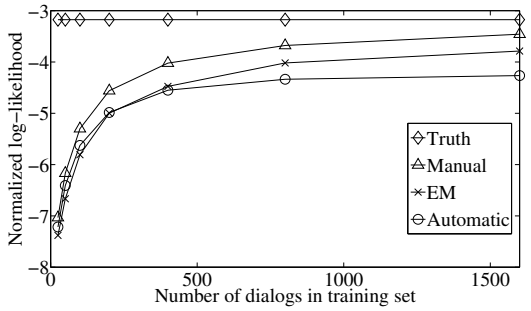


Figure 2: **Normalized log-likelihood of each model type with respect to the test set vs. size of training set.** Each data point is the average of 50 runs. For the largest training set, the EM models had higher normalized log-likelihood than the Automatic models in 48 out of 50 runs.

4.2 Real Data

We tested the three estimation methods from the previous section on a data set of 461 real dialogs, which we split into a training set of 315 dialogs and a test set of 146 dialogs. All the dialogs were both manually and automatically transcribed, so that each of the three methods was applicable. The normalized log-likelihood of each user model, with respect to both the training and test set, is given in Table 1. Since the output of the EM method depends on a random choice of starting point $\theta^{(0)}$, those results were averaged over 50 runs.

	Training Set $\ell(\theta)$	Test Set $\ell(\theta)$
Manual	-2.87	-3.73
EM	-3.90	-4.33
Automatic	-4.60	-5.80

Table 1: **Normalized log-likelihood of each model type with respect to the training set and the test set.** The EM values are the average of 50 runs. The EM models had higher normalized log-likelihood than the Automatic model in 50 out of 50 runs.

5 Conclusion

We have shown that user models can be estimated from automatically transcribed dialog corpora by modeling dialogs within a probabilistic framework that accounts for transcription errors in a principled way. This method may lead to many interesting future applications, such as continuous learning of a user model while the dialog system is on-line, enabling automatic adaptation.

References

AP Dempster, NM Laird, and DB Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Stat. Soc.*, 39:1–38.

K Georgila, J Henderson, and O Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Proc ICSLP, Pittsburgh, USA*.

E Levin, R Pieraccini, and W Eckert. 2000. A stochastic model of human-machine interaction for learning dialogue strategies. *IEEE Trans on Speech and Audio Processing*, 8(1):11–23.

O Pietquin. 2004. *A framework for unsupervised learning of dialogue strategies*. Ph.D. thesis, Faculty of Engineering, Mons (TCTS Lab), Belgium.

LR Rabiner, 1990. *A tutorial on hidden Markov models and selected applications in speech recognition*, pages 267–296. Morgan Kaufmann Publishers, Inc.

J Schatzmann, B Thomson, and SJ Young. 2007. Statistical user simulation with a hidden agenda. In *Proc SIGDial, Antwerp*, pages 273–282.

B Thomson, J Schatzmann, K Welhammer, H Ye, and SJ Young. 2007. Training a real-world POMDP-based dialog system. In *Proc NAACL-HLT Workshop Bridging the Gap: Academic and Industrial Research in Dialog Technologies, Rochester, New York, USA*, pages 9–17.

JD Williams and SJ Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Robust Extraction of Named Entity Including Unfamiliar Word

Masatoshi Tsuchiya[†] **Shinya Hida**[‡] **Seiichi Nakagawa**[‡]
[†]Information and Media Center / [‡]Department of Information and Computer Sciences,
Toyohashi University of Technology
tsuchiya@imc.tut.ac.jp, {hida,nakagawa}@slp.ics.tut.ac.jp

Abstract

This paper proposes a novel method to extract named entities including unfamiliar words which do not occur or occur few times in a training corpus using a large unannotated corpus. The proposed method consists of two steps. The first step is to assign the most similar and familiar word to each unfamiliar word based on their context vectors calculated from a large unannotated corpus. After that, traditional machine learning approaches are employed as the second step. The experiments of extracting Japanese named entities from IREX corpus and NHK corpus show the effectiveness of the proposed method.

1 Introduction

It is widely agreed that extraction of named entity (henceforth, denoted as NE) is an important subtask for various NLP applications. Various machine learning approaches such as maximum entropy(Uchimoto et al., 2000), decision list(Sassano and Utsuro, 2000; Isozaki, 2001), and Support Vector Machine(Yamada et al., 2002; Isozaki and Kazawa, 2002) were investigated for extracting NEs.

All of them require a corpus whose NEs are annotated properly as training data. However, it is difficult to obtain an enough corpus in the real world, because there are increasing the number of NEs like personal names and company names. For example, a large database of organization names(Nichigai Associates, 2007) already contains 171,708 entries and is still increasing. Therefore, a robust method to extract NEs including unfamiliar words which do not occur or occur few times in a training corpus is necessary.

This paper proposes a novel method of extracting NEs which contain unfamiliar morphemes using a large unannotated corpus, in order to resolve the above problem. The proposed method consists of two steps. The first step is to assign the most similar and familiar morpheme to each unfamiliar morpheme based on their context vectors calculated from a large unannotated corpus. The second step is to employ traditional machine learning approaches using both features of original morphemes and features of similar morphemes. The experiments of extracting Japanese NEs from IREX corpus and NHK corpus show the effectiveness of the proposed method.

Table 1: Statistics of NE Types of IREX Corpus

NE Type	Frequency	(%)
ARTIFACT	747	(4.0)
DATE	3567	(19.1)
LOCATION	5463	(29.2)
MONEY	390	(2.1)
ORGANIZATION	3676	(19.7)
PERCENT	492	(2.6)
PERSON	3840	(20.6)
TIME	502	(2.7)
Total	18677	

2 Extraction of Japanese Named Entity

2.1 Task of the IREX Workshop

The task of NE extraction of the IREX workshop (Sekine and Eriguchi, 2000) is to recognize eight NE types in Table 1. The organizer of the IREX workshop provided a training corpus, which consists of 1,174 newspaper articles published from January 1st 1995 to 10th which include 18,677 NEs. In the Japanese language, no other corpus whose NEs are annotated is publicly available as far as we know.[1]

2.2 Chunking of Named Entities

It is quite common that the task of extracting Japanese NEs from a sentence is formalized as a chunking problem against a sequence of mor-

[1]The organizer of the IREX workshop also provides the testing data to its participants, however, we cannot see it because we did not join it.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 125–128,
Columbus, Ohio, USA, June 2008.

phemes. For representing proper chunks, we employ IOB2 representation, one of those which have been studied well in various chunking tasks of NLP (Tjong Kim Sang, 1999). This representation uses the following three labels.

- **B** Current token is the beginning of a chunk.
- **I** Current token is a middle or the end of a chunk consisting of more than one token.
- **O** Current token is outside of any chunk.

Actually, we prepare the 16 derived labels from the label **B** and the label **I** for eight NE types, in order to distinguish them.

When the task of extracting Japanese NEs from a sentence is formalized as a chunking problem of a sequence of morphemes, the segmentation boundary problem arises as widely known. For example, the NE definition of IREX tells that a Chinese character "米 (bei)" must be extracted as an NE means *America* from a morpheme "訪米 (hou-bei)" which means *visiting America*. A naive chunker using a morpheme as a chunking unit cannot extract such kind of NEs. In order to cope this problem, (Uchimoto et al., 2000) proposed employing translation rules to modify problematic morphemes, and (Asahara and Matsumoto, 2003; Nakano and Hirai, 2004) formalized the task of extracting NEs as a chunking problem of a sequence of characters instead of a sequence of morphemes. In this paper, we keep the naive formalization, because it is still enough to compare performances of proposed methods and baseline methods.

3 Robust Extraction of Named Entities Including Unfamiliar Words

The proposed method of extracting NEs consists of two steps. Its first step is to assign the most similar and familiar morpheme to each unfamiliar morpheme based on their context vectors calculated from a large unannotated corpus. The second step is to employ traditional machine learning approaches using both features of original morphemes and features of similar morphemes. The following subsections describe these steps respectively.

3.1 Assignment of Similar Morpheme

A context vector V_m of a morpheme m is a vector consisting of frequencies of all possible unigrams and bigrams,

$$V_m = \begin{pmatrix} f(m,m_0), & \cdots & f(m,m_N), \\ f(m,m_0,m_0), & \cdots & f(m,m_N,m_N), \\ f(m_0,m), & \cdots & f(m_N,m), \\ f(m_0,m_0,m), & \cdots & f(m_N,m_N,m) \end{pmatrix},$$

where $M \equiv \{m_0, m_1, \ldots, m_N\}$ is a set of all morphemes of the unannotated corpus, $f(m_i, m_j)$ is a frequency that a sequence of a morpheme m_i and a morpheme m_j occurs in the unannotated corpus, and $f(m_i, m_j, m_k)$ is a frequency that a sequence of morphemes m_i, m_j and m_k occurs in the unannotated corpus.

Suppose an unfamiliar morpheme $m_u \in M \cap \overline{M_F}$, where M_F is a set of familiar morphemes that occur frequently in the annotated corpus. The most similar morpheme $\hat{m_u}$ to the morpheme m_u measured with their context vectors is given by the following equation,

$$\hat{m_u} = \underset{m \in M_F}{\operatorname{argmax}}\, sim(V_{m_u}, V_m), \qquad (1)$$

where $sim(V_i, V_j)$ is a similarity function between context vectors. In this paper, the cosine function is employed as it.

3.2 Features

The feature set F_i at i-th position is defined as a tuple of the *morpheme feature* $MF(m_i)$ of the i-th morpheme m_i, the *similar morpheme feature* $SF(m_i)$, and the *character type feature* $CF(m_i)$.

$$F_i = \langle\, MF(m_i),\ SF(m_i),\ CF(m_i)\, \rangle$$

The morpheme feature $MF(m_i)$ is a pair of the surface string and the part-of-speech of m_i. The similar morpheme feature $SF(m_i)$ is defined as

$$SF(m_i) = \begin{cases} MF(\hat{m_i}) & \text{if } m_i \in M \cap \overline{M_F} \\ MF(m_i) & \text{otherwise} \end{cases},$$

where $\hat{m_i}$ is the most similar and familiar morpheme to m_i given by Equation (1). The character type feature $CF(m_i)$ is a set of four binary flags to indicate that the surface string of m_i contains a Chinese character, a *hiragana* character, a *katakana* character, and an English alphabet respectively.

When we identify the chunk label c_i for the i-th morpheme m_i, the surrounding five feature sets $F_{i-2}, F_{i-1}, F_i, F_{i+1}, F_{i+2}$ and the preceding two chunk labels c_{i-2}, c_{i-1} are refered.

Morpheme Feature			Similar Morpheme Feature			Character Type Feature	Chunk Label
	(English translation)	POS		*(English translation)*	POS		
今日 (kyou)	(*today*)	Noun–Adverbial	今日 (kyou)	(*today*)	Noun–Adverbial	$\langle 1,0,0,0\rangle$	O
の (no)	gen	Particle	の (no)	gen	Particle	$\langle 0,1,0,0\rangle$	O
石狩 **(Ishikari)**	(*Ishikari*)	Noun–Proper	**関東 (Kantou)**	(*Kantou*)	Noun–Proper	$\langle 1,0,0,0\rangle$	B-LOCATION
平野 (heiya)	(*plain*)	Noun–Generic	平野 (heiya)	(*plain*)	Noun–Generic	$\langle 1,0,0,0\rangle$	I-LOCATION
の (no)	gen	Particle	の (no)	gen	Particle	$\langle 0,1,0,0\rangle$	O
天気 (tenki)	(*weather*)	Noun–Generic	天気 (tenki)	(*weather*)	Noun–Generic	$\langle 1,0,0,0\rangle$	O
は (ha)	top	Particle	は (ha)	top	Particle	$\langle 0,1,0,0\rangle$	O
晴れ (hare)	(*fine*)	Noun–Generic	晴れ (hare)	(*fine*)	Noun–Generic	$\langle 1,1,0,0\rangle$	O

Figure 1: Example of Training Instance for Proposed Method

$\longrightarrow$ Parsing Direction $\longrightarrow$

Feature set	F_{i-2}	F_{i-1}	F_i	F_{i+1}	F_{i+2}
Chunk label	c_{i-2}	c_{i-1}	$\boxed{c_i}$		

Figure 1 shows an example of training instance of the proposed method for the sentence "今日 (kyou) の (no) 石狩 (Ishikari) 平野 (heiya) の (no) 天気 (tenki) は (ha) 晴れ (hare)" which means "*It is fine at Ishikari-plain, today*". "関東 (Kantou)" is assigned as the most similar and familiar morpheme to "石狩 (Ishikari)" which is unfamiliar in the training corpus.

4 Experimental Evaluation

4.1 Experimental Setup

IREX Corpus is used as the annotated corpus to train statistical NE chunkers, and M_F is defined experimentally as a set of all morphemes which occur five or more times in IREX corpus. Mainichi Newspaper Corpus (1993–1995), which contains 3.5M sentences consisting of 140M words, is used as the unannotated corpus to calculate context vectors. MeCab[2](Kudo et al., 2004) is used as a preprocessing morphological analyzer through experiments.

In this paper, either Conditional Random Fields(CRF)[3](Lafferty et al., 2001) or Support Vector Machine(SVM)[4](Cristianini and Shawe-Taylor, 2000) is employed to train a statistical NE chunker.

4.2 Experiment of IREX Corpus

Table 2 shows the results of extracting NEs of IREX corpus, which are measured with F-measure through 5-fold cross validation. The columns of "Proposed" show the results with SF, and the ones of "Baseline" show the results without SF. The column of "NExT" shows the result of using NExT(Masui et al., 2002), an NE chunker based on hand-crafted rules, without 5-fold cross validation.

Table 2: NE Extraction Performance of IREX Corpus

	Proposed		Baseline		NExT
	CRF	SVM	CRF	SVM	
ARTIFACT	0.487	0.518	0.458	0.457	-
DATE	0.921	0.909	0.916	0.916	0.682
LOCATION	0.866	0.863	0.847	0.846	0.696
MONEY	0.951	0.610	0.937	0.937	0.895
ORGANIZATION	0.774	0.766	0.744	0.742	0.506
PERCENT	0.936	0.863	0.928	0.928	0.821
PERSON	0.825	0.842	0.788	0.787	0.672
TIME	0.901	0.903	0.902	0.901	0.800
Total	0.842	0.834	0.821	0.820	0.732

Table 3: Statistics of NE Types of NHK Corpus

NE Type	Frequency (%)
DATE	755 (19%)
LOCATION	1465 (36%)
MONEY	124 (3%)
ORGANIZATION	1056 (26%)
PERCENT	55 (1%)
PERSON	516 (13%)
TIME	101 (2%)
Total	4072

As shown in Table 2, machine learning approaches with SF outperform ones without SF. Please note that the result of SVM without SF and the result of (Yamada et al., 2002) are comparable, because our using feature set without SF is quite similar to their feature set. This fact suggests that SF is effective to achieve better performances than the previous research. CRF with SF achieves better performance than SVM with SF, although CRF and SVM are comparable in the case without SF. NExT achieves poorer performance than CRF and SVM.

4.3 Experiment of NHK Corpus

Nippon Housou Kyoukai (NHK) corpus is a set of transcriptions of 30 broadcast news programs which were broadcasted from June 1st 1996 to 12th. Table 3 shows the statistics of NEs of NHK corpus which were annotated by a graduate student except

[2] http://mecab.sourceforge.net/

[3] http://chasen.org/~taku/software/CRF++/

[4] http://chasen.org/~taku/software/yamcha/

Table 4: NE Extraction Performance of NHK Corpus

	Proposed		Baseline		NExT
	CRF	SVM	CRF	SVM	
DATE	0.630	0.595	0.571	0.569	0.523
LOCATION	0.837	0.825	0.797	0.811	0.741
MONEY	0.988	0.660	0.971	0.623	0.996
ORGANIZATION	0.662	0.636	0.601	0.598	0.612
PERCENT	0.538	0.430	0.539	0.435	0.254
PERSON	0.794	0.813	0.752	0.787	0.622
TIME	0.250	0.224	0.200	0.247	0.260
Total	0.746	0.719	0.702	0.697	0.615

Table 5: Extraction of Familiar/Unfamiliar NEs

	Familiar	Unfamiliar	Other
CRF (Proposed)	0.789	0.654	0.621
CRF (Baseline)	0.757	0.556	0.614

for ARTIFACT in accordance with the NE definition of IREX. Because all articles of IREX corpus had been published earlier than broadcasting programs of NHK corpus, we can suppose that NHK corpus contains unfamiliar NEs like real input texts.

Table 4 shows the results of chunkers trained from whole IREX corpus against NHK corpus. The methods with SF outperform the ones without SF. Furthermore, performance improvements between the ones with SF and the ones without SF are greater than Table 2.

The performance of CRF with SF and one of CRF without SF are compared in Table 5. The column "Familiar" shows the results of extracting NEs which consist of familiar morphemes, as well as the column "Unfamiliar" shows the results of extracting NEs which consist of unfamiliar morphemes. The column "Other" shows the results of extracting NEs which contain both familiar morpheme and unfamiliar one. These results indicate that SF is especially effective to extract NEs consisting of unfamiliar morphemes.

5 Concluding Remarks

This paper proposes a novel method to extract NEs including unfamiliar morphemes which do not occur or occur few times in a training corpus using a large unannotated corpus. The experimental results show that SF is effective for robust extracting NEs which consist of unfamiliar morphemes. There are other effective features of extracting NEs like N-best morpheme sequences described in (Asahara and Matsumoto, 2003) and features of surrounding phrases described in (Nakano and Hirai, 2004). We will investigate incorporating SF and these features in the near future.

References

Masayuki Asahara and Yuji Matsumoto. 2003. Japanese named entity extraction with redundant morphological analysis. In *Proc. of HLT–NAACL '03*, pages 8–15.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proc. of the 19th COLING*, pages 1–7.

Hideki Isozaki. 2001. Japanese named entity recognition based on a simple rule generator and decision tree learning. In *Proc. of ACL '01*, pages 314–321.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Appliying conditional random fields to japanese morphological analysis. In *Proc. of EMNLP2004*, pages 230–237.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*, pages 282–289.

Fumito Masui, Shinya Suzuki, and Junichi Fukumoto. 2002. Development of named entity extraction tool NExT for text processing. In *Proceedings of the 8th Annual Meeting of the Association for Natural Language Processing*, pages 176–179. (in Japanese).

Keigo Nakano and Yuzo Hirai. 2004. Japanese named entity extraction with bunsetsu features. *Transactions of Information Processing Society of Japan*, 45(3):934–941, Mar. (in Japanese).

Nichigai Associates, editor. 2007. *DCS Kikan-mei Jisho*. Nichigai Associates. (in Japanese).

Manabu Sassano and Takehito Utsuro. 2000. Named entity chunking techniques in supervised learning for japanese named entity recognition. In *Proc. of the 18th COLING*, pages 705–711.

Satoshi Sekine and Yoshio Eriguchi. 2000. Japanese named entity extraction evaluation: analysis of results. In *Proc. of the 18th COLING*, pages 1106–1110.

E. Tjong Kim Sang. 1999. Representing text chunks. In *Proc. of the 9th EACL*, pages 173–179.

Kiyotaka Uchimoto, Ma Qing, Masaki Murata, Hiromi Ozaku, Masao Utiyama, and Hitoshi Isahara. 2000. Named entity extraction based on a maximum entropy model and transformation rules. *Journal of Natural Language Processing*, 7(2):63–90, Apr. (in Japanese).

Hiroyasu Yamada, Taku Kudo, and Yuji Matsumoto. 2002. Japanese named entity extraction using support vector machine. *Transactions of Information Processing Society of Japan*, 43(1):44–53, Jan. (in Japanese).

In-Browser Summarisation: Generating Elaborative Summaries Biased Towards the Reading Context

Stephen Wan and Cécile Paris
ICT Centre*
CSIRO
Locked Bag 17, North Ryde, Sydney
NSW 1670, Australia
Firstname.Lastname@csiro.au

Abstract

We investigate *elaborative summarisation*, where the aim is to identify supplementary information that expands upon a key fact. We envisage such summaries being useful when browsing certain kinds of (hyper-)linked document sets, such as Wikipedia articles or repositories of publications linked by citations. For these collections, an elaborative summary is intended to provide additional information on the linking anchor text. Our contribution in this paper focuses on identifying and exploring a real task in which summarisation is situated, realised as an *In-Browser* tool. We also introduce a *neighbourhood* scoring heuristic as a means of scoring matches to relevant passages of the document. In a preliminary evaluation using this method, our summarisation system scores above our baselines and achieves a recall of 57% annotated gold standard sentences.

1 Introduction

It has long been held that a summary is useful, particularly if it supports the underlying task of the user — for an overview of summarisation scenarios see Spark Jones (1998). For example, *generic* (that is, not query-specific) summaries, which are often indicative, providing just the gist of a document, are only useful if they happen to address the underlying need of the user.

In a push to make summaries more responsive to user needs, the field of summarisation has explored the overlap with complex question-answering research to produce query-focused summaries. Such work includes the recent DUC challenges on query-focused summarisation,[1] in which the user needs are represented by short paragraphs of text written by human judges. These are then used as input to the summarisation process. However, modelling user needs is a difficult task. DUC descriptions of information needs are only an artificial stipulation of a user's interest.

In this work, we propose a tool built into an internet browser that makes use of a very simple heuristic for determining user interest.[2] The basic premise of the heuristic is that the text currently being read provides an approximation of the current user interest. Specifically, as a user reads a sentence, it potentially represents a fine-grained information need. We identify the sentence of interest without complex methods, relying instead on the user to move the mouse over the anchor text link to request a summary of the *linked* document, thus identifying to the browser plug-in which sentence is now in focus.

To generate the summary, the whole document, specifically the *linking* sentence that contains the anchor text, serves as the *reading context*, a potential indicator of the user interest. An example of the current output on Wikipedia text is presented in Figure 1. It shows an *elaborative summary* of a document about the Space Shuttle Discovery expanding on the content of the linking sentence. In this case, it gives further information about a space walk in which the shuttle was repaired inflight.

Our summarisation tool, the In-Browser Elabora-

*Information and Communication Technologies Centre

[1] http://duc.nist.gov/guidelines/2006.html

[2] We currently work with the Firefox browser.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 129–132,
Columbus, Ohio, USA, June 2008.

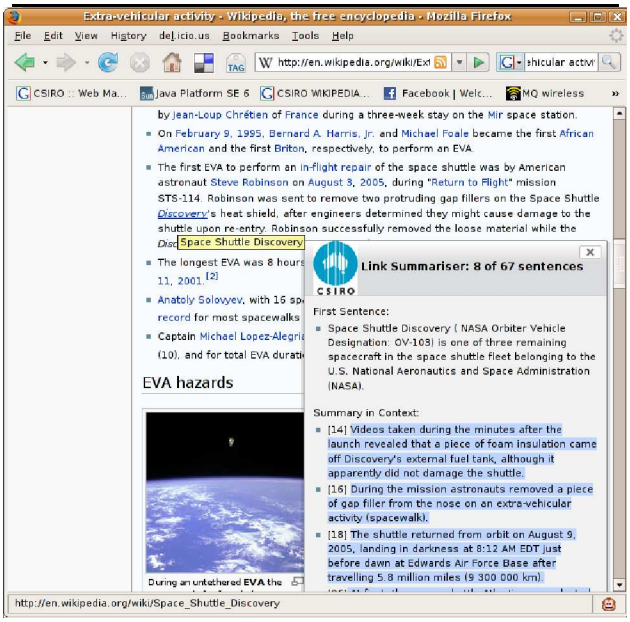


Figure 1: A summary generated when moving the mouse over the link "Discovery's" (mouse pointer omitted).

tive Summariser (IBES), complements generic summaries in providing additional information about a particular aspect of a page.[3] Generic summaries themselves are easy to generate due to rules enforced by the Wikipedia style-guide, which dictates that all titles be noun phrases describing an entity, thus serving as a short generic summary. Furthermore, the first sentence of the article should contain the title in subject position, which tends to create sentences that define the main entity of the article.

For the elaborative summarisation scenario described, we are interested in exploring ways in which the reading context can be leveraged to produce the elaborative summary. One method explored in this paper attempts to map the content of the *linked* document into the semantic space of the reading context, as defined in vector-space. We use Singular Value Decomposition (SVD), the underlying method behind Latent Semantic Analysis (Deerwester et al., 1990), as a means of identifying latent topics in the reading context, against which we compare the linked document. We present our system and the results from our preliminary investigation in the remainder of this paper.

[3]http://www.ict.csiro.au/staff/stephen.wan/ibes/

2 Related Work

Using link text for summarisation has been explored previously by Amitay and Paris (2000). They identified situations when it was possible to generate summaries of web-pages by recycling human-authored descriptions of links from anchor text. In our work, we use the anchor text as the reading context to provide an elaborative summary for the linked document.

Our work is similar in domain to that of the 2007 CLEF WiQA shared task.[4] However, in contrast to our application scenario, the end goal of the shared task focuses on suggesting editing updates for a particular document and not on elaborating on the user's reading context.

A related task was explored at the Document Understanding Conference (DUC) in 2007.[5] Here the goal was to find new information with respect to a previously seen set of documents. This is similar to the elaborative goal of our summary in the sense that one could answer the question: "What else can I say about topic X (that hasn't already been mentioned in the reading context)". However, whereas DUC focused on unlinked news wire text, we explore a different genre of text.

3 Algorithm

Our approach is designed to *select* justification sentences and *expand* upon them by finding elaborative material. The first stage identifies those sentences in the linked document that support the semantic content of the anchor text. We call those sentences justification material. The second stage finds material that is supplementary yet relevant for the user. In this paper, we report on the first of these tasks, though ultimately both are required for elaborative summaries.

To locate justification material, we implemented two known summarisation techniques. The first compares word overlap between the anchor text and the linked document. The second approach attempts to discover a semantic space, as defined by the reading context. The linked document is then mapped into this semantic space. These are referred to as the Simple Link method and the SVD method, where

[4]http://ilps.science.uva.nl/WiQA/

[5]http://duc.nist.gov/guidelines/2007.html

the latter divides further into two variants: SVD-Link and SVD-topic.

3.1 Simple Link Method

The first strategy, Simple Link, makes use of standard vector space approaches from Information Retrieval. A vector of word frequencies, omitting stopwords, is used to represent each sentence in the reading context and in the linked document. The vector for the anchor sentence is compared with vectors for each linked document sentence, using the cosine similarity metric. The highest scoring sentences are then retrieved as the summary.

3.2 Two Singular Value Decomposition (SVD) Methods

In these approaches, the semantic space of the linked document is mapped into that of the reading context. Intuitively, only those sentences that map well into the reading context space and are similar to the linking sentence would be good justification material.

To begin with, the reading context document is represented as a term-by-sentence matrix, A, where stop words are omitted and frequencies are weighted using inverse document frequency. A Singular Value Decomposition (SVD) analysis is performed (using the JAMA package[6]) on this matrix which provides three resulting matrices: $A = USV^{tr}$.

The S-matrix defines the *themes* of the reading context. The U-matrix relates the reading context vocabulary to the discovered themes. Finally, the V-matrix relates the original sentences to each of the themes. The point of the SVD analysis is to discover these themes based on co-variance between the word frequencies. If words occur together, they are semantically related and the co-variance is marked as a theme, allowing one to capture fuzzy matches between related words. Crucially, each sentence can now be represented with a vector of membership scores to each theme.

The first of the semantic space mapping methods, SVD-link, finds the theme that the anchor text belongs to best. This is done by consulting the V-matrix of the SVD analysis to find the highest scoring theme for that sentence, which we call the linking theme. Each sentence in the linked document, after mapping it to the SVD-derived vector space, is then examined. The highest scoring sentences that belong to the linking theme are then extracted.

The second method, SVD-topic, makes a different assumption about the nature of the reading context. Instead of taking the anchor text as an indicator of the user's information need, it assumes that the top n themes of the reading context document represent the user's interest. Of the linked document sentences, for each of those top n reading context themes, the best scoring sentence is extracted.

4 Evaluation

In lieu of a user-centered experiment, our preliminary experiments evaluated the effectiveness of the tool in terms of finding justification material for an elaborative summary. We evaluated the three systems described in Section 3. Each system selected 5 sentences. We tested against two baselines. The first simply returns the first 5 sentences. The second produces a generic summary based on Gong and Liu (2001), *independently* of the reading context.

4.1 Data

The data used is a collection of Wikipedia articles obtained automatically from the web. The snapshot of the corpus was collected in 2007. Of these, links from about 600 randomly chosen documents were filtered with a heuristic that enforced a sentence length of at least 10 words such that the link in the anchor text occurred after this minimum length. This heuristic was used as an approximate means of filtering out sentences where the linking sentence was simply a definition of the entity linked. In these cases, the justification material is usually trivially identified as the first sentence of the linked document. This leaves us with links that potentially require more complicated summarisation methods.

Of these cases, 125 cases were randomly selected and the linked documents annotated for varying degrees of relevancy. This resulted in 50 relevant document links, which we further annotated, selecting sentences supporting the anchor sentence, with a Cohen's Kappa of 0.55. The intersection of the selected sentences was then used as a gold standard for each test case.

[6] http://math.nist.gov/javanumerics/jama/

System	Recall	Precision
generic	0.13	0.05
SVD-topic	0.14	0.06
SVD-link	0.22	0.09
simple-link	0.28	0.11

Table 1: Recall and Precision figures for all summarisers without the first 5 sentences.

System	Recall	Precision
generic	0.27	0.04
SVD-topic	0.27	0.04
SVD-link	0.30	0.05
simple-link	0.38	0.06

Table 2: Recall and Precision figures using the *neighbourhood* heuristic (without the first 5 sentences).

4.2 Results

It is difficult to beat the first-5 baseline, which attains the best recall of 0.52 and a precision of 0.2, with all other strategies falling behind. However, we believe that this may be due to the presence of some types of Wikipedia articles that are narrow in scope and centered on specific events. For such articles, we would naturally advocate using the first N sentences as a summary.

To examine the performance of the summarisation strategies on sentences beyond the top-N, we filtered the gold standard sets to remove sentences occurring in positions 1-5 in the linked document, and tested recall and precision on the remaining sentences. This reduces our test set by 10 cases. Since documents may be lengthy (more than 100 sentences), selecting justification material is a difficult task. The results are shown in Table 1 and indicate that systems using reading context do better than a generic summariser.

Thinking ahead to the second expansion step in which we find elaborative material, good candidates for such sentences may be found in the immediate vicinity of justification sentences. If so, near matches for justification sentences may still be useful in indicating that, at least, the right portion of the document was identified. Thus, to test for near matches, we scored a match if the gold sentence occurred on either side of the system-selected sentence. We refer to this as the *neighbourhood heuristic*.

Table 2 shows the effect on recall and precision if we treat each selected sentence as defining a neighbourhood of relevance in the linked document. Again, performance on the first 5 sentences were ignored. Recall improved by up to 10% with only a small drop in precision (6%). When the neighbourhood heuristic is run on the original gold sentence set (with the first 5 sentences), recall reaches 0.57, which lies above an amended 0.55 baseline.

5 Future Work and Conclusions

We introduced the concept of a user-biased elaborative summarisation, using the reading context as an indicator of the information need. Our paper presents a scenario in which elaborative summarisation may be useful and explored simple summarisation strategies to perform this role. Results are encouraging and our preliminary evaluation shows that reading context is helpful, achieving a recall of 57% when identifying sentences that justify content in the linking sentence of the reading context. In future work, we intend to explore other latent topic methods to improve recall and precision performance. Further development of elaborative summarisation strategies and a user-centered evaluation are also planned.

References

Einat Amitay and Cécile Paris. 2000. Automatically summarising web sites: is there a way around it? In *Proceedings of the 9th international conference on Information and knowledge management*, NY, USA.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th ACM SIGIR conference*. New Orleans, USA.

Karen Spark Jones. 1998. Automatic summarizing: factors and directions. In I. Mani and M. Maybury (ed.), *Advances in Automatic Text Summarisation*. MIT Press, Cambridge MA.

Sentiment Vector Space Model for Lyric-based Song Sentiment Classification

Yunqing Xia
Center for Speech and language Tech.
RIIT, Tsinghua University
Beijing 100084, China
yqxia@tsinghua.edu.cn

Linlin Wang
State Key Lab of Intelligent Tech. and Sys.
Dept. of CST, Tsinghua University
Beijing 100084, China
wangll07@mails.tsinghua.edu.cn

Kam-Fai Wong
Dept. of SE&EM
The Chinese University of Hong Kong
Shatin, Hong Kong
kfwong@se.cuhk.edu.hk

Mingxing Xu
Dept. of CST
Tsinghua University
Beijing 100084, China
xumx@tsinghua.edu.cn

Abstract

Lyric-based song sentiment classification seeks to assign songs appropriate sentiment labels such as *light-hearted* and *heavy-hearted*. Four problems render vector space model (VSM)-based text classification approach ineffective: 1) Many words within song lyrics actually contribute little to sentiment; 2) Nouns and verbs used to express sentiment are ambiguous; 3) Negations and modifiers around the sentiment keywords make particular contributions to sentiment; 4) Song lyric is usually very short. To address these problems, the sentiment vector space model (s-VSM) is proposed to represent song lyric document. The preliminary experiments prove that the s-VSM model outperforms the VSM model in the lyric-based song sentiment classification task.

1 Introduction

Song sentiment classification nowadays becomes a hot research topic due largely to the increasing demand of ubiquitous song access, especially via mobile phone. In their music phone *W910i*, Sony and Ericsson provide *Sense Me* component to catch owner's mood and play songs accordingly. Song sentiment classification is the key technology for song recommendation. Many research works have been reported to achieve this goal using audio signal (Knees et al., 2007). But research efforts on lyric-based song classification are very few.

Preliminary experiments show that VSM-based text classification method (Joachims, 2002) is ineffective in song sentiment classification (see Section 5) due to the following four reasons. Firstly, the VSM model considers all content words within song lyric as features in text classification. But in fact many words in song lyric actually make little contribution to sentiment expressing. Using all content words as features, the VSM-based classification methods perform poorly in song sentiment classification. Secondly, observation on lyrics of thousands of Chinese pop songs reveals that sentiment-related nouns and verbs usually carry multiple senses. Unfortunately, the ambiguity is not appropriately handled in the VSM model. Thirdly, negations and modifiers are constantly found around the sentiment words in song lyric to inverse, to strengthen or to weaken the sentiments that the sentences carry. But the VSM model is not capable of reflecting these functions. Lastly, song lyric is usually very short, namely 50 words on average in length, rendering serious sparse data problem in VSM-based classification.

To address the aforementioned problems of the VSM model, the sentiment vector space model (s-VSM) is proposed in this work. We adopt the s-VSM model to extract sentiment features from song lyrics and implement the *SVM-light* (Joachims, 2002) classification algorithm to assign sentiment labels to given songs.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 133–136,
Columbus, Ohio, USA, June 2008.

2 Related Works

Song sentiment classification has been investigated since 1990s in audio signal processing community and research works are mostly found relying on audio signal to make a decision using machine learning algorithms (Li and Ogihara, 2006; Lu et al., 2006). Typically, the sentiment classes are defined based on the Thayer's arousal-valence emotion plane (Thayer, 1989). Instead of assigning songs one of the four typical sentiment labels, Lu et al. (2006) propose the hierarchical framework to perform song sentiment classification with two steps. In the first step the *energy level* is detected with intensity features and the *stress level* is determined in the second step with timbre and rhythm features. It is proved difficult to detect *stress level* using audio as classification proof.

Song sentiment classification using lyric as proof is recently investigated by Chen et al. (2006). They adopt the hierarchical framework and make use of song lyric to detect *stress level* in the second step. In fact, many literatures have been produced to address the sentiment analysis problem in natural language processing research. Three approaches are dominating, i.e. knowledge-based approach (Kim and Hovy, 2004), information retrieval-based approach (Turney and Littman, 2003) and machine learning approach (Pang et al., 2002), in which the last approach is found very popular. Pang et al. (2002) adopt the VSM model to represent product reviews and apply text classification algorithms such as Naïve Bayes, maximum entropy and support vector machines to predict sentiment polarity of given product review.

Chen et al. (2006) also apply the VSM model in lyric-based song sentiment classification. However, our experiments show that song sentiment classification with the VSM model delivers disappointing quality (see Section 5). Error analysis reveals that the VSM model is problematic in representing song lyric. It is necessary to design a new lyric representation model for song sentiment classification.

3 Sentiment Vector Space Model

We propose the sentiment vector space model (s-VSM) for song sentiment classification. Principles of the s-VSM model are listed as follows.

(1) Only sentiment-related words are used to produce sentiment features for the s-VSM model.
(2) The sentiment words are appropriately disambiguated with the neighboring negations and modifiers.
(3) Negations and modifiers are included in the s-VSM model to reflect the functions of inversing, strengthening and weakening.

Sentiment unit is found the appropriate element complying with the above principles.

To be general, we first present the notation for sentiment lexicon as follows.

$$L = \{C, N, M\};\ C = \{c_i\}, i = 1,..., I$$
$$N = \{n_j\},\ j = 1,..., J$$
$$M = \{m_l\}, l = 1,..., L$$

in which L represents sentiment lexicon, C sentiment word set, N negation set and M modifier set. These words can be automatically extracted from a semantic dictionary and each sentiment word is assigned a sentiment label, namely *light-hearted* or *heavy-hearted* according to its lexical definition.

Given a piece of song lyric, denoted as follows,

$$W = \{w_h\}, h = 1,..., H$$

in which W denotes a set of words that appear in the song lyric, the semantic lexicon is in turn used to locate sentiment units denoted as follows.

$$U = \{u_v\} = \{c_{i,v}, n_{j,v}, m_{l,v}\}$$
$$c_{i,v}, \in W \cap C;\ n_{j,v} \in W \cap N;\ m_{l,v} \in W \cap M$$

Note that sentiment units are unambiguous sentiment expressions, each of which contains one sentiment word and possibly one modifier and one negation. Negations and modifiers are helpful to determine the unique meaning of the sentiment words within certain context window, e.g. 3 preceding words and 3 succeeding words in our case. Then, the s-VSM model is presented as follows.

$$V_S = (f_1(U), f_2(U),..., f_T(U)).$$

in which V_S represents the sentiment vector for the given song lyric and $f_i(U)$ sentiment features which are usually certain statistics on sentiment units that appear in lyric.

We classify the sentiment units according to occurrence of sentiment words, negations and modifiers. If the sentiment word is mandatory for any sentiment unit, eight kinds of sentiment units are obtained. Let f_{PSW} denote count of positive senti-

ment words (PSW), f_{NSW} count of negative sentiment words (NSW), f_{NEG} count of negations (NEG) and f_{MOD} count of modifiers (MOD). Eight sentiment features are defined in Table 1.

f_i	Number of sentiment units satisfying …
f_1	$f_{PSW}>0$, $f_{NSW}=f_{NEG}=f_{MOD}=0$
f_2	$f_{PSW}=0$, $f_{NSW}>0$, $f_{NEG}=f_{MOD}=0$
f_3	$f_{PSW}>0$, $f_{NSW}=0$, $f_{NEG}>0$, $f_{MOD}=0$
f_4	$f_{PSW}=0$, $f_{NSW}>0$, $f_{NEG}>0$, $f_{MOD}=0$
f_5	$f_{PSW}>0$, $f_{NSW}=0$, $f_{NEG}=0$, $f_{MOD}>0$
f_6	$f_{PSW}=0$, $f_{NSW}>0$, $f_{NEG}=0$, $f_{MOD}>0$
f_7	$f_{PSW}>0$, $f_{NSW}=0$, $f_{NEG}>0$, $f_{MOD}>0$
f_8	$f_{PSW}=0$, $f_{NSW}>0$, $f_{NEG}>0$, $f_{MOD}>0$

Table 1. Definition of sentiment features. Note that one sentiment unit contains only one sentiment word. Thus it is not possible that f_{PSW} and f_{NSW} are both bigger than zero.

Obviously, sparse data problem can be well addressed using statistics on sentiment units rather than on individual words or sentiment units.

4 Lyric-based Song Sentiment Classification

Song sentiment classification based on lyric can be viewed as a text classification task thus can be handled by some standard classification algorithms. In this work, the *SVM-light* algorithm is implemented to accomplish this task due to its excellence in text classification.

Note that song sentiment classification differs from the traditional text classification in feature extraction. In our case, sentiment units are first detected and the sentiment features are then generated based on sentiment units. As the sentiment units carry unambiguous sentiments, it is deemed that the s-VSM is model is promising to carry out the song sentiment classification task effectively.

5 Evaluation

To evaluate the s-VSM model, a song corpus, i.e. 5SONGS, is created manually. It covers 2,653 Chinese pop songs, in which 1,632 are assigned label of *light-hearted* (positive class) and 1,021 assigned *heavy-hearted* (negative class). We randomly select 2,001 songs (around 75%) for training and the rest for testing. We adopt the standard evaluation criteria in text classification, namely precision (*p*), recall (*r*), f-1 measure (*f*) and accuracy (*a*) (Yang and Liu, 1999).

In our experiments, three approaches are implemented in song sentiment classification, i.e. audio-based (AB) approach, knowledge-based (KB) approach and machine learning (ML) approach, in which the latter two approaches are also referred to as text-based (TB) approach. The intentions are 1) to compare AB approach against the two TB approaches, 2) to compare the ML approach against the KB approach, and 3) to compare the VSM-based ML approach against the s-VSM-based one.

Audio-based (AB) Approach

We extract 10 timbre features and 2 rhythm features (Lu et al., 2006) from audio data of each song. Thus each song is represented by a 12-dimension vector. We run *SVM-light* algorithm to learn on the training samples and classify test ones.

Knowledge-based (KB) Approach

We make use of HowNet (Dong and dong, 2006), to detect sentiment words, to recognize the neighboring negations and modifiers, and finally to locate sentiment units within song lyric. Sentiment (SM) of the sentiment unit (SU) is determined considering sentiment words (SW), negation (NEG) and modifiers (MOD) using the following rule.

(1) *SM*(SU) = *label*(SW);
(2) *SM*(SU) = - *SM*(SU) iff SU contains NEG;
(3) *SM*(SU) = *degree*(MOD)**SM*(SU) iff SU contains MOD.

In the above rule, *label*(x) is the function to read sentiment label($\in\{1, -1\}$) of given word in the sentiment lexicon and *degree*(x) to read its modification degree($\in\{1/2, 2\}$). As the sentiment labels are integer numbers, the following formula is adopted to obtain label of the given song lyric.

$$label = sign\left(\sum_i SM(SU_i)\right)$$

Machine Learning (ML) Approach

The ML approach adopts text classification algorithms to predict sentiment label of given song lyric. The *SVM-light* algorithm is implemented based on VSM model and s-VSM model, respectively. For the VSM model, we apply (CHI) algorithm (Yang and Pedersen, 1997) to select effective sentiment word features. For the s-VSM model, we adopt HowNet as the sentiment lexicon to create sentiment vectors.

Experimental results are presented Table 2.

	p	*R*	*f-1*	*a*
Audio-based	0.504	0.701	0.586	0.504
Knowledge-based	0.726	0.584	0.647	0.714
VSM-based	0.587	**1.000**	0.740	0.587
s-VSM-based	**0.783**	0.750	**0.766**	**0.732**

Table 2. Experimental results

Table 2 shows that the text-based methods outperform the audio-based method. This justifies our claim that lyric is better than audio in song sentiment detection. The second observation is that machine learning approach outperforms the knowledge-based approach. The third observation is that s-VSM-based method outperforms VSM-based method on *f-1* score. Besides, we surprisingly find that VSM-based method assigns all test samples *light-hearted* label thus *recall* reaches 100%. This makes results of VSM-based method unreliable. We look into the model file created by the *SVM-light* algorithm and find that 1,868 of 2,001 VSM training vectors are selected as support vectors while 1,222 s-VSM support vectors are selected. This indicates that the VSM model indeed suffers the problems mentioned in Section 1 in lyric-based song sentiment classification. As a comparison, the s-VSM model produces more discriminative support vectors for the SVM classifier thus yields reliable predictions.

6 Conclusions and Future Works

The s-VSM model is presented in this paper as a document representation model to address the problems encountered in song sentiment classification. This model considers sentiment units in feature definition and produces more discriminative support vectors for song sentiment classification. Some conclusions can be drawn from the preliminary experiments on song sentiment classification. Firstly, text-based methods are more effective than the audio-based method. Secondly, the machine learning approach outperforms the knowledge-based approach. Thirdly, s-VSM model is more reliable and more accurate than the VSM model. We are thus encouraged to carry out more research to further refine the s-VSM model in sentiment classification. In the future, we will incorporate some linguistic rules to improve performance of sentiment unit detection. Meanwhile, sentiment features in the s-VSM model are currently equally weighted. We will adopt some estimation techniques to assess their contributions for the s-VSM model. Finally, we will also explore how the s-VSM model improves quality of polarity classification in opinion mining.

Acknowledgement

Research work in this paper is partially supported by NSFC (No. 60703051) and Tsinghua University under the Basic Research Foundation (No. JC2007049).

References

R.H. Chen, Z.L. Xu, Z.X. Zhang and F.Z. Luo. *Content Based Music Emotion Analysis and Recognition*. Proc. of 2006 International Workshop on Computer Music and Audio Technology, pp.68-75. 2006.

Z. Dong and Q. Dong. *HowNet and the Computation of Meaning*. World Scientific Publishing. 2006.

T. Joachims. *Learning to Classify Text Using Support Vector Machines, Methods, Theory, and Algorithms*. Kluwer (2002).

S.-M. Kim and E. Hovy. *Determining the Sentiment of Opinions*. Proc. COLING'04, pp. 1367-1373. 2004.

P. Knees, T. Pohle, M. Schedl and G. Widmer. *A Music Search Engine Built upon Audio-based and Web-based Similarity Measures*. Proc. of SIGIR'07, pp.47-454. 2007

T. Li and M. Ogihara. *Content-based music similarity search and emotion detection*. Proc. IEEE Int. Conf. Acoustic, Speech, and Signal Processing, pp. 17–21. 2006.

L. Lu, D. Liu and H. Zhang. *Automatic mood detection and tracking of music audio signals*. IEEE Transactions on Audio, Speech & Language Processing 14(1): 5-18 (2006).

B. Pang, L. Lee and S. Vaithyanathan. *Thumbs up? Sentiment Classification using Machine Learning Techniques*. Proc. of EMNLP-02, pp.79-86. 2002.

R. E. Thayer, *The Biopsychology of Mood and Arousal*, New York, Oxford University Press. 1989.

P. D. Turney and M. L. Littman. *Measuring praise and criticism: Inference of semantic orientation from association.* ACM Trans. on Information Systems, 21(4):315–346. 2003.

Y. Yang and X. Liu. *A Re-Examination of Text Categorization Methods*. Proc. of SIGIR'99, pp. 42-49. 1999.

Y. Yang and J. O. Pedersen. *A comparative study on feature selection in text categorization*. Proc. ICML'97, pp.412-420. 1997.

Mining Wikipedia Revision Histories for Improving Sentence Compression

Elif Yamangil **Rani Nelken**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, USA
{elif,nelken}@eecs.harvard.edu

Abstract

A well-recognized limitation of research on supervised sentence compression is the dearth of available training data. We propose a new and bountiful resource for such training data, which we obtain by mining the revision history of Wikipedia for sentence compressions and expansions. Using only a fraction of the available Wikipedia data, we have collected a training corpus of over 380,000 sentence pairs, two orders of magnitude larger than the standardly used Ziff-Davis corpus. Using this newfound data, we propose a novel lexicalized noisy channel model for sentence compression, achieving improved results in grammaticality and compression rate criteria with a slight decrease in importance.

1 Introduction

With the increasing success of machine translation (MT) in recent years, several researchers have suggested transferring similar methods for monolingual text rewriting tasks. In particular, Knight and Marcu (2000) (KM) applied a channel model to the task of *sentence compression* – dropping words from an individual sentence while retaining its important information, and without sacrificing its grammaticality. Compressed sentences can be useful either on their own, e.g., for subtitles, or as part of a larger summarization or MT system. A well-recognized problem of this approach, however, is data sparsity. While bilingual parallel corpora are abundantly available, monolingual parallel corpora, and especially collections of sentence compressions are vanishingly rare. Indeed, most work on sentence compression has used the Ziff-Davis corpus (Knight and Marcu, 2000), which consists of a mere 1067 sentence pairs. While data sparsity is a common problem of many NLP tasks, it is much more severe for sentence compression, leading Turner and Charniak (2005) to question the applicability of the channel model for this task altogether.

Our contribution in this paper is twofold. First, we solve the data sparsity issue by showing that abundant sentence compressions can be extracted from Wikipedia's revision history. Second, we use this data to validate the channel model approach for text compression, and improve upon it by creating a novel fully lexicalized compression model. Our model improves grammaticality and compression rate with only a slight decrease in importance.

2 Data: Wikipedia revision histories as a source of sentence compressions

Many researchers are increasingly turning to Wikipedia as a large-scale data source for training NLP systems. The vast majority of this work uses only the most recent version of the articles. In fact, Wikipedia conveniently provides not only the latest version, but the entire revision history of each of its articles, as dramatically visualized by Viégas et al. (2004). Through Wikipedia's collaborative editing process, articles are iteratively amended and refined by multiple Web users. Users can usually change any aspect of the document's structure and content, but for our purposes here, we focus only on sentence-level edits that add or drop words.

We have downloaded the July snapshot of the

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 137–140,
Columbus, Ohio, USA, June 2008.

English Wikipedia, consisting of 1.4 million articles, and mined a subset of them for such compressions/expansions. We make the simplifying assumption that *all* such edits also retain the core meaning of the sentence, and are therefore valid training data for our purposes. This assumption is of course patently naïve, as there are many cases in which such revisions reverse sentence meaning, add or drop essential information, are part of a flame war, etc. Classifying these edits is an interesting task which we relegate to future work.[1]

From about one-third of the snapshot, we extracted over 380,000 sentence pairs, which is 2 orders of magnitude more than the Ziff-Davis corpus.[2] Wikipedia currently has 2.3 million articles and is constantly expanding. We can therefore expect an increase of another order of magnitude. We thus can afford to be extremely selective of the sentence pairs we use. To handle a dataset of such size (hundreds of GBs), we split it into smaller chunks, and distribute all the processing.

More technically, for each article, we first extract all revisions, and split each revision into a list of its sentences. We run an edit-distance comparison between each such pair, treating each sentence as an atomic "letter". We look for all replacements of one sentence by another and check whether one sentence is a compression of the other.[3] We then run Collins' parser (1997), using just the sentence pairs where parsing succeeds with a negative log likelihood below 200.

3 Noisy channel model

We follow KM in modeling the problem using a generative noisy channel model, but use the new-found training data to lexicalize the model. Sentences start their life in short form, s, are ranked by a source language model, $p(s)$, and then probabilistically expanded to form the long sentence, $p(l|s)$. During decoding, given a long sentence, we seek the most likely short sentence that could have generated it. Using Bayes' rule, this is equivalent to seeking the short sentence s that maximizes $p(s) \cdot p(l|s)$.

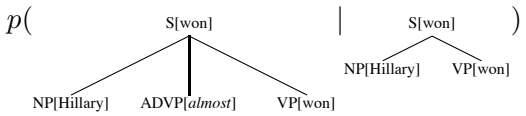
3.1 Lexicalized channel model

KM's original model was purely syntax-based. Daume et al. (2002) used a lexicalized PCFG to rerank the compressions, showing that the addition of lexical information helps eliminate improbable compressions. Here, we propose to enhance lexicalization by including lexical information within the channel model, allowing us to better model which compressions are likely and which are not. A minimal example pair illustrating the utility of lexicalization is the following.

(1) Hillary *barely* won the primaries.

(2) Hillary *almost* won the primaries.

The validity of dropping the adverbial here clearly depends on the lexical value of the adverb. It is more acceptable to drop the adverb in Sentence 1, since dropping it in Sentence 2 reverses the meaning. We learn probabilities of the form:



Our model has the power of making compression decisions based on lexical dependencies between the compressed and retained parts of the parse tree.

Note that Daume et al.'s reranking model cannot achieve this type of distinction, since it is based on reranking the compressed version, at which point the adverb is no longer available.

Since we are interested not only in learning how to compress, but also when to compress, we also include in this procedure unchanged CFG rule pairs that are attested in the corpus. Thus, different ways of expanding a CFG rule compete with each other as well as the possibility of not doing any expansion.

3.2 Smoothing

In order to smooth our estimates we use Witten-Bell discounting (1991) with 6 levels of back-off. This method enables us to tune the confidence parameter associated with an estimate inversely proportionally with the diversity of the context of the estimate. The different levels are illustrated in Table 1. Level 1,

[1] For instance, compressions are more likely to signal optional information than expansions; the lexical items added are likely to be indicative of the type of edit, etc.

[2] The sentence pair corpus is available by contacting the authors.

[3] We ignore word reorderings or replacements that are beyond word addition or deletion.

the most specific level, is fully lexicalized. Transitioning to levels 2 to 4, we lose the lexical information about the subtrees that are not dropped, the head child bearing subtree, and the dropped subtrees, respectively. At level 4, we end up with the non-lexicalized estimates that are equivalent to KM's model. In subsequent back off levels, we abstract away from the CFG rules. In particular, level 5 estimates the probability of dropping subtrees in the context of a certain parent and head child, and level 6 estimates the probability of the same outcome in the coarser context of a parent only.

3.3 Source model

In addition to the lexicalized channel model, we also use a lexicalized probabilistic syntax-based source model, which we train from the parser's output on the short sentences of each pair.

3.4 Decoding

We implemented the forest-based statistical sentence generation method of Langkilde (2000). KM tailored this method to sentence compression, compactly encoding all compressions of a sentence in a forest structure. The forest ranking algorithm which extracts compressed parse trees, optimized the model scores as well as an additional bigram score. Since our model is lexicalized, the bigram scores become less relevant, which was confirmed by experimentation during development. Therefore in our implementation we exclude the bigram scores and other related aspects of the algorithm such as pruning of bigram-suboptimal phrases.

4 Evaluation

We evaluated our system using the same method as KM, using the same 32 sentences taken from the Ziff-Davis corpus. We solicited judgments of *importance* (the value of the retained information), and *grammaticality* for our compression, the KM results, and human compressions from 8 judges, on a scale of 1 (worst) to 5 (best). Mean and standard deviation are shown in Table 2. Our model improves grammaticality and compression rate criteria with only a slight decrease in importance. Here are some illustrative examples, with the deleted material shown in brackets:

(3) The chemical etching process [used for glare protection] is effective and will help if your office has the fluorescent-light overkill [that 's typical in offices].

(4) Prices range from $5,000 [for a microvax 2000] to $179,000 [for the vax 8000 or higher series].

We suspect that the decrease in importance stems from our indiscriminative usage of compressions and expansions to train our system. We hypothesize that in Wikipedia, expansions often add more useful information, as opposed to compressions which are more likely to drop superfluous or erroneous information.[4] Further work is required to classify sentence modifications.

Since one of our model's back-off levels simulates KM's model, we plan to perform an additional comparative evaluation of both models trained on the same data.

5 Discussion and future work

Turner and Charniak (2005) question the viability of a noisy channel model for the sentence compression task. Briefly put, in the typically sparse data setting, there is no way to distinguish between the probability of a sentence as a short sentence and its probability as a regular sentence of English. Furthermore, the channel model is likely to prefer to leave sentences intact, since that is the most prevalent pattern in the training data. Thus, they argue, the channel model is not really compressing, and it is only by virtue of the length penalty that anything gets shortened at all. Our hope here is that by using a far richer source of short sentences, as well as a huge source of compressions, we can overcome this problem. The noisy channel model posits a virtual competition on each word of coming either from the source model (in which case it is retained in the compression) or from the channel model (in which case it is dropped). By having access to a large data set for the first time, we hope to be able to learn which parts of the sentence are more likely to come from

[4] For instance, here is an expansion seen in the data, where the added information (italicized) is important: "In 1952 and 1953 he was stationed in Sendai, Japan during the Korean War *and was shot*." It would be undesirable to drop this added phrase.

Back-off level	expanded	short
1	S[won] → NP[Hillary] ADVP[almost] VP[won]	S[won] → NP[Hillary] VP[won]
2	S[won] → NP ADVP[almost] VP[won]	S[won] → NP VP[won]
3	S → NP ADVP[almost] VP	S → NP VP
4	S → NP ADVP VP	S → NP VP
5	parent = S, head-child = VP, child = ADVP	parent = S, head-child = VP
6	parent = S, child = ADVP	parent = S

Table 1: Back off levels

	KM	Our model	Humans
Compression	72.91%	67.38%	53.33%
Grammaticality	4.02±1.03	4.31±0.78	4.78±0.17
Importance	3.86±1.09	3.65±1.07	3.90±0.58

Table 2: Evaluation results

which of the two parts of the model. Further work is required in order to clarify this point.

Naturally, discriminative models such as McDonald (2006) are also likely to improve by using the added data. We leave the exploration of this topic for future work.

Finally, we believe that the Wikipedia revision history offers a wonderful resource for many additional NLP tasks, which we have begun exploring.

Acknowledgments

This work was partially supported by a Google research award, "Mining Wikipedia's Revision History". We thank Stuart Shieber for his comments on an early draft of this paper, Kevin Knight and Daniel Marcu for sharing the Ziff-Davis dataset with us, and the volunteers for rating sentences. Yamangil thanks Michael Collins for his feedback on the project idea.

References

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Somerset, New Jersey. Association for Computational Linguistics.

H. Daume, Kevin Knight, I Langkilde-Geary, Daniel Marcu, and K Yamada. 2002. The importance of lexicalized syntax models for natural language generation tasks. Proceedings of the Second International Conference on Natural Language Generation. Arden House, NJ, July 1-3.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 170–177, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ryan T. McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, April 3-7, 2006, Trento, Italy*, pages 297–304.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 290–297, Morristown, NJ, USA. Association for Computational Linguistics.

Fernanda B. Viégas, Martin Wattenberg, and Kushal Dave. 2004. Studying cooperation and conflict between authors with *istory flow* visualizations. In Elizabeth Dykstra-Erickson and Manfred Tscheligi, editors, *CHI*, pages 575–582. ACM.

I. Witten and T. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4).

Smoothing a Tera-word Language Model

Deniz Yuret
Koç University
dyuret@ku.edu.tr

Abstract

Frequency counts from very large corpora, such as the Web 1T dataset, have recently become available for language modeling. Omission of low frequency n-gram counts is a practical necessity for datasets of this size. Naive implementations of standard smoothing methods do not realize the full potential of such large datasets with missing counts. In this paper I present a new smoothing algorithm that combines the Dirichlet prior form of (Mackay and Peto, 1995) with the modified back-off estimates of (Kneser and Ney, 1995) that leads to a 31% perplexity reduction on the Brown corpus compared to a baseline implementation of Kneser-Ney discounting.

1 Introduction

Language models, i.e. models that assign probabilities to sequences of words, have been proven useful in a variety of applications including speech recognition and machine translation (Bahl et al., 1983; Brown et al., 1990). More recently, good results on lexical substitution and word sense disambiguation using language models have also been reported (Yuret, 2007).

The recently introduced Web 1T 5-gram dataset (Brants and Franz, 2006) contains the counts of word sequences up to length five in a 10^{12} word corpus derived from publicly accessible Web pages. As this corpus is several orders of magnitude larger than the ones used in previous language modeling studies, it holds the promise to provide more accurate domain independent probability estimates. However, naive application of the well-known smoothing methods do not realize the full potential of this dataset.

In this paper I present experiments with modifications and combinations of various smoothing methods using the Web 1T dataset for model building and the Brown corpus for evaluation. I describe a new smoothing method, Dirichlet-Kneser-Ney (DKN), that combines the Bayesian intuition of MacKay and Peto (1995) and the improved back-off estimation of Kneser and Ney (1995) and gives significantly better results than the baseline Kneser-Ney discounting.

The next section describes the general structure of n-gram models and smoothing. Section 3 describes the data sets and the experimental methodology used. Section 4 presents experiments with adaptations of various smoothing methods. Section 5 describes the new algorithm.

2 N-gram Models and Smoothing

N-gram models are the most commonly used language modeling tools. They estimate the probability of each word using the context made up of the previous $n-1$ words. Let abc represent an n-gram where a is the first word, c is the last word, and b represents ***zero or more words*** in between. One way to estimate $\Pr(c|ab)$ is to look at the number of times word c has followed the previous $n-1$ words ab,

$$\Pr(c|ab) = \frac{C(abc)}{C(ab)} \quad (1)$$

where $C(x)$ denotes the number of times x has been observed in the training corpus. This is the maximum likelihood (ML) estimate. Unfortunately it

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 141–144,
Columbus, Ohio, USA, June 2008.

does not work very well because it assigns zero probability to n-grams that have not been observed in the training corpus. To avoid the zero probabilities, we take some probability mass from the observed n-grams and distribute it to unobserved n-grams. Such redistribution is known as smoothing or discounting.

Most existing smoothing methods can be expressed in one of the following two forms:

$$\Pr(c|ab) = \alpha(c|ab) + \gamma(ab)\Pr(c|b) \quad (2)$$

$$\Pr(c|ab) = \begin{cases} \beta(c|ab) & \text{if } C(abc) > 0 \\ \gamma(ab)\Pr(c|b) & \text{otherwise} \end{cases} \quad (3)$$

Equation 2 describes the so-called interpolated models and Equation 3 describes the back-off models. The highest order distributions $\alpha(c|ab)$ and $\beta(c|ab)$ are typically discounted to be less than the ML estimate so we have some leftover probability for the c words unseen in the context ab. Different methods mainly differ on how they discount the ML estimate. The back-off weights $\gamma(ab)$ are computed to make sure the probabilities are normalized. The interpolated models always incorporate the lower order distribution $\Pr(c|b)$ whereas the back-off models consider it only when the n-gram abc has not been observed in the training data.

3 Data and Method

All the models in this paper are interpolated models built using the counts obtained from the Web 1T dataset and evaluated on the million word Brown corpus using cross entropy (bits per token). The lowest order model is taken to be the word frequencies in the Web 1T corpus. The Brown corpus was retokenized to match the tokenization style of the Web 1T dataset resulting in 1,186,262 tokens in 52,108 sentences. The Web 1T dataset has a 13 million word vocabulary consisting of words that appear 100 times or more in its corpus. 769 sentences in Brown that contained words outside this vocabulary were eliminated leaving 1,162,052 tokens in 51,339 sentences. Capitalization and punctuation were left intact. The n-gram patterns of the Brown corpus were extracted and the necessary counts were collected from the Web 1T dataset in one pass. The end-of-sentence tags were not included in the entropy calculation. For parameter optimization, numerical optimization was performed on a 1,000 sentence random sample of Brown.

4 Experiments

In this section, I describe several smoothing methods and give their performance on the Brown corpus. Each subsection describes a single idea and its impact on the performance. All methods use interpolated models expressed by $\alpha(c|ab)$ and $\gamma(ab)$ based on Equation 2. The Web 1T dataset does not include n-grams with counts less than 40, and I note the specific implementation decisions due to the missing counts where appropriate.

4.1 Absolute Discounting

Absolute discounting subtracts a fixed constant D from each nonzero count to allocate probability for unseen words. A different D constant is chosen for each n-gram order. Note that in the original study, D is taken to be between 0 and 1, but because the Web 1T dataset does not include n-grams with counts less than 40, the optimized D constants in our case range from 0 to 40. The interpolated form is:

$$\alpha(c|ab) = \frac{\max(0, C(abc) - D)}{C(ab*)} \quad (4)$$

$$\gamma(ab) = \frac{N(ab*)D}{C(ab*)}$$

The $*$ represents a wildcard matching any word and $C(ab*)$ is the total count of n-grams that start with the $n-1$ words ab. If we had complete counts, we would have $C(ab*) = \sum_c C(abc) = C(ab)$. However because of the missing counts in general $C(ab*) \leq C(ab)$ and we need to use the former for proper normalization. $N(ab*)$ denotes the number of distinct words following ab in the training data. Absolute discounting achieves its best performance with a 3-gram model and gives 8.53 bits of cross entropy on the Brown corpus.

4.2 Kneser-Ney

Kneser-Ney discounting (Kneser and Ney, 1995) has been reported as the best performing smoothing method in several comparative studies (Chen and Goodman, 1999; Goodman, 2001). The $\alpha(c|ab)$ and $\gamma(ab)$ expressions are identical to absolute discounting (Equation 4) for the highest order n-grams.

However, a modified estimate is used for lower order n-grams used for back-off. The interpolated form is:

$$\begin{aligned} \Pr(c|ab) &= \alpha(c|ab) + \gamma(ab)\Pr'(c|b) \quad (5) \\ \Pr'(c|ab) &= \alpha'(c|ab) + \gamma'(ab)\Pr'(c|b) \end{aligned}$$

Specifically, the modified estimate $\Pr'(c|b)$ for a lower order n-gram is taken to be proportional to the number of unique words that precede the n-gram in the training data. The α' and γ' expressions for the modified lower order distributions are:

$$\begin{aligned} \alpha'(c|b) &= \frac{\max(0, N(*bc) - D)}{N(*b*)} \quad (6) \\ \gamma'(b) &= \frac{R(*b*)D}{N(*b*)} \end{aligned}$$

where $R(*b*) = |c : N(*bc) > 0|$ denotes the number of distinct words observed on the right hand side of the $*b*$ pattern. A different D constant is chosen for each n-gram order. The lowest order model is taken to be $\Pr(c) = N(*c)/N(**)$. The best results for Kneser-Ney are achieved with a 4-gram model and its performance on Brown is 8.40 bits.

4.3 Correcting for Missing Counts

Kneser-Ney takes the back-off probability of a lower order n-gram to be proportional to the number of unique words that precede the n-gram in the training data. Unfortunately this number is not exactly equal to the $N(*bc)$ value given in the Web 1T dataset because the dataset does not include low count abc n-grams. To correct for the missing counts I used the following modified estimates:

$$\begin{aligned} N'(*bc) &= N(*bc) + \delta(C(bc) - C(*bc)) \\ N'(*b*) &= N(*b*) + \delta(C(b*) - C(*b*)) \end{aligned}$$

The difference between $C(bc)$ and $C(*bc)$ is due to the words preceding bc less than 40 times. We can estimate their number to be a fraction of this difference. δ is an estimate of the type token ratio of these low count words. Its valid range is between 1/40 and 1, and it can be optimized along with the other parameters. The reader can confirm that $\sum_c N'(*bc) = N'(*b*)$ and $|c : N'(*bc) > 0| = N(b*)$. The expression for the Kneser-Ney back-off estimate becomes

$$\begin{aligned} \alpha'(c|b) &= \frac{\max(0, N'(*bc) - D)}{N'(*b*)} \quad (7) \\ \gamma'(b) &= \frac{N(b*)D}{N'(*b*)} \end{aligned}$$

Using the corrected N' counts instead of the plain N counts achieves its best performance with a 4-gram model and gives 8.23 bits on Brown.

4.4 Dirichlet Form

MacKay and Peto (1995) show that based on Dirichlet priors a reasonable form for a smoothed distribution can be expressed as

$$\begin{aligned} \alpha(c|ab) &= \frac{C(abc)}{C(ab*) + A} \quad (8) \\ \gamma(ab) &= \frac{A}{C(ab*) + A} \end{aligned}$$

The parameter A can be interpreted as the extra counts added to the given distribution and these extra counts are distributed as the lower order model. Chen and Goodman (1996) suggest that these extra counts should be proportional to the number of words with exactly one count in the given context based on the Good-Turing estimate. The Web 1T dataset does not include one-count n-grams. A reasonable alternative is to take A to be proportional to the missing count due to low-count n-grams: $C(ab) - C(ab*)$.

$$A(ab) = \max(1, K(C(ab) - C(ab*)))$$

A different K constant is chosen for each n-gram order. Using this formulation as an interpolated 5-gram language model gives a cross entropy of 8.05 bits on Brown.

4.5 Dirichlet with KN Back-Off

Using a modified back-off distribution for lower order n-grams gave us a big boost in the baseline results from 8.53 bits for absolute discounting to 8.23 bits for Kneser-Ney. The same idea can be applied to the missing-count estimate. We can use Equation 8 for the highest order n-grams and Equation 7 for lower order n-grams used for back-off. Such a 5-gram model gives a cross entropy of 7.96 bits on the Brown corpus.

5 A New Smoothing Method: DKN

In this section, I describe a new smoothing method that combines the Dirichlet form of MacKay and

Peto (1995) and the modified back-off distribution of Kneser and Ney (1995). We will call this new method Dirichlet-Kneser-Ney, or DKN for short. The important idea in Kneser-Ney is to let the probability of a back-off n-gram be proportional to the number of unique words that precede it. However we do not need to use the absolute discount form for the estimates. We can use the Dirichlet prior form for the lower order back-off distributions as well as the highest order distribution. The extra counts A in the Dirichlet form are taken to be proportional to the missing counts, and the coefficient of proportionality K is optimized for each n-gram order. Where complete counts are available, A should be taken to be proportional to the number of one-count n-grams instead. This smoothing method with a 5-gram model gives a cross entropy of 7.86 bits on the Brown corpus achieving a perplexity reduction of 31% compared to the naive implementation of Kneser-Ney.

The relevant equations are repeated below for the reader's convenience.

$$\begin{aligned}
\Pr(c|ab) &= \alpha(c|ab) + \gamma(ab)\Pr'(c|b) \\
\Pr'(c|ab) &= \alpha'(c|ab) + \gamma'(ab)\Pr'(c|b) \\
\alpha(c|b) &= \frac{C(bc)}{C(b*) + A(b)} \\
\gamma(b) &= \frac{A(b)}{C(b*) + A(b)} \\
\alpha'(c|b) &= \frac{N'(*bc)}{N'(*b*) + A(b)} \\
\gamma'(b) &= \frac{A(b)}{N'(*b*) + A(b)} \\
A(b) &= \max(1, K(C(b) - C(b*))) \\
&\text{or} \quad \max(1, K|c : C(bc) = 1|)
\end{aligned}$$

6 Summary and Discussion

Frequency counts based on very large corpora can provide accurate domain independent probability estimates for language modeling. I presented adaptations of several smoothing methods that can properly handle the missing counts that may exist in such datasets. I described a new smoothing method, DKN, combining the Bayesian intuition of MacKay and Peto (1995) and the modified back-off distribution of Kneser and Ney (1995) which achieves a significant perplexity reduction compared to a naive implementation of Kneser-Ney smoothing. This is a surprizing result because Chen and Goodman (1999) partly attribute the performance of Kneser-Ney to the use of absolute discounting. The relationship between Kneser-Ney smoothing to the Bayesian approach have been explored in (Goldwater et al., 2006; Teh, 2006) using Pitman-Yor processes. These models still suggest discount-based interpolation with type frequencies whereas DKN uses Dirichlet smoothing throughout. The conditions under which the Dirichlet form is superior is a topic for future research.

References

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Linguistic Data Consortium, Philadelphia. LDC2006T13.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the ACL*.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*.

S. Goldwater, T.L. Griffiths, and M. Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Joshua Goodman. 2001. A bit of progress in language modeling. *Computer Speech and Language*.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*.

David J. C. Mackay and Linda C. Bauman Peto. 1995. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19.

Y.W. Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the ACL*, pages 985–992.

Deniz Yuret. 2007. KU: Word sense disambiguation by substitution. In *SemEval-2007: 4th International Workshop on Semantic Evaluations*.

Event Matching Using the Transitive Closure of Dependency Relations

Daniel M. Bikel and Vittorio Castelli
IBM T. J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
{dbikel,vittorio}@us.ibm.com

Abstract

This paper describes a novel event-matching strategy using features obtained from the transitive closure of dependency relations. The method yields a model capable of matching events with an F-measure of 66.5%.

1 Introduction

Question answering systems are evolving from their roots as factoid or definitional answering systems to systems capable of answering much more open-ended questions. For example, it is one thing to ask for the birthplace of a person, but it is quite another to ask for all locations visited by a person over a specific period of time.

Queries may contain several types of arguments: person, organization, country, location, etc. By far, however, the most challenging of the argument types are the event or topic arguments, where the argument text can be a noun phrase, a participial verb phrase or an entire indicative clause. For example, the following are all possible event arguments:

- the U.S. invasion of Iraq
- Red Cross admitting Israeli and Palestinian groups
- GM offers buyouts to union employees

In this paper, we describe a method to match an event query argument to the sentences that mention that event. That is, we seek to model $p(s \text{ contains } e \mid s, e)$, where e is a textual description of an event (such as an event argument for a GALE distillation query) and where s is an arbitrary sentence. In the first example above, "the U.S. invasion of Iraq", such a model should produce a very high score for that event description and the sentence "The U.S. invaded Iraq in 2003."

2 Low-level features

As the foregoing implies, we are interested in training a binary classifier, and so we represent each training and test instance in a feature space. Conceptually, our features are of three different varieties. This section describes the first two kinds, which we call "low-level" features, in that they attempt to capture how much of the basic information of an event e is present in a sentence s.

2.1 Lexical features

We employ several types of simple lexical-matching features. These are similar to the "bag-of-words" features common to many IR and question-answering systems. Specifically, we compute the value $\text{overlap}(s, e) = \frac{\mathbf{w}_s \cdot \mathbf{w}_e}{|\mathbf{w}_e|_1}$, where $\mathbf{w}_e$ (resp: $\mathbf{w}_s$) is the $\{0,1\}$-valued word-feature vector for the event (resp: sentence). This value is simply the fraction of distinct words in e that are present in s. We then quantize this fraction into the bins $[0, 0]$, $(0, 0.33]$, $(0.33, 0.66]$, $(0.66, 0.99]$, $(0.99, 1]$, to produce one of five, binary-valued features to indicate whether none, few, some, many or all of the words match.[1]

2.2 Argument analysis and submodels

Since an event or topic most often involves entities of various kinds, we need a method to recognize those entity mentions. For example, in the event "Abdul Halim Khaddam resigns as Vice President of Syria", we have a PERSON mention, an OCCUPATION mention and a GPE (geopolitical entity) mention. We use an information extraction toolkit (Florian et al., 2004) to analyze each event argument. The toolkit performs the following steps: tokenization, part-of-speech tagging, parsing, mention detection, within-document coreference resolution and cross-document coreference resolution. We also apply the toolkit to our entire search corpus.

After determining the entities in an event description, we rely on lower-level binary classifiers, each of which has been trained to match a specific type

[1] Other binnings did not significantly alter the performance of the models we trained, and so we used the above binning strategy for all experiments reported in this paper.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 145–148,
Columbus, Ohio, USA, June 2008.

of entity. For example, we use a PERSON-matching model to determine if, say, "Abdul Halim Khaddam" from an event description is mentioned in a sentence.[2] We build binary-valued feature functions from the output of our four lower-level classifiers.

3 Dependency relation features

Employing syntactic or dependency relations to aid question answering systems is by no means new (Attardi et al., 2001; Cui et al., 2005; Shen and Klakow, 2006). These approaches all involved various degrees of loose matching of the relations in a query relative to sentences. More recently, Wang et al. (2007) explored the use a formalism called *quasi-synchronous grammar* (Smith and Eisner, 2006) in order to find a more explicit model for matching the set of dependencies, and yet still allow for looseness in the matching.

3.1 The dependency relation

In contrast to previous work using relations, we do not seek to model explicitly a process that transforms one dependency tree to another, nor do we seek to come up with *ad hoc* correlation measures or path similarity measures. Rather, we propose to use features based on the transitive closure of the dependency relation of the event and that of the dependency relation of the sentence. Our aim was to achieve a balance between the specificity of dependency paths and the generality of dependency pairs.

In its most basic form, a *dependency tree* for a sentence $\mathbf{w} = \langle \omega_1, \omega_w, \ldots, \omega_k \rangle$ is a rooted tree $\tau = \langle V, E, r \rangle$, where $V = \{1, \ldots, k\}$, $E = \left\{(i, j) : \omega_i \text{ is the child of } \omega_j\right\}$ and $r \in \{1, \ldots, k\}$: ω_r is the root word. Each element ω_i of our word sequence, rather than being a simple lexical item drawn from a finite vocabulary, will be a complex structure. With each word w_i we associate a part-of-speech tag t_i, a morph (or stem) m_i (which is w_i itself if w_i has no variant), a set of nonterminal labels N_i, a set of synonyms S_i for that word and a canonical mention $cm(i)$. Formally, we let each sequence element be a sextuple $\omega_i = \langle w_i, t_i, m_i, N_i, S_i, cm(i) \rangle$.

[2]This is not as trivial as it might sound: the model must deal with name variants (parts of names, alternate spellings, nicknames) and with metonymic uses of titles ("Mr. President" referring to Bill Clinton or George W. Bush).

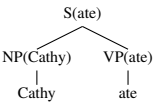


Figure 1: Simple lexicalized tree.

We derive dependency trees from head-lexicalized syntactic parse trees. The set of nonterminal labels associated with each word is the set of labels of the nodes for which that word was the head. For example, in the lexicalized tree in Figure 1, the head word "ate" would be associated with both the nonterminals `S` and `VP`. Also, if a head word is part of an entity mention, then the "canonical" version of that mention is associated with the word, where canonical essentially means the best version of that mention in its coreference chain (produced by our information extraction toolkit), denoted $cm(i)$. In Figure 1, the first word w_1 = `Cathy` would probably be recognized as a PERSON mention, and if the coreference resolver found it to be coreferent with a mention earlier in the same document, say, `Cathy Smith`, then $cm(1)$ = `Cathy Smith`.

3.2 Matching on the transitive closure

Since E represents the child-of dependency relation, let us now consider the transitive closure, E', which is then the descendant-of relation.[3] Our features are computed by examining the overlap between E'_e and E'_s, the descendant-of relation of the event description e and the sentence s, respectively. We use the following, two-tiered strategy.

Let d_e, d_s be elements of E'_e and E'_s, with $d_x.d$ denoting the index of the word that is the descendant in d_x and $d_x.a$ denoting the ancestor. We define the following matching function to match the pair of descendants (or ancestors):

$$\text{match}_d(d_e, d_s) = \\ (m_{d_e.d} = m_{d_s.d}) \vee (cm(d_e.d) = cm(d_s.d)) \quad (1)$$

where match_a is defined analogously for ancestors. That is, $\text{match}_d(d_e, d_s)$ returns `true` if the morph of the descendant of d_e is the same as the morph of the descendant of d_s, *or* if both descendants have canonical mentions with an exact string match; the

[3]We remove all edges (i, j) from E' where either w_i or w_j is a stop word.

function returns `false` otherwise, and match_a is defined analogously for the pair of ancestors. Thus, the pair of functions $\text{match}_d, \text{match}_a$ are "morph or mention" matchers. We can now define our main matching function in terms of match_d and match_a:

$$\text{match}(d_e, d_s) = \text{match}_d(d_e, d_s) \wedge \text{match}_a(d_e, d_s). \quad (2)$$

Informally, $\text{match}(d_e, d_s)$ returns `true` if the pair of descendants have a "morph-or-mention" match *and* if the pair of ancestors have a "morph-or-mention" match. When $\text{match}(d_e, d_s) =$ `true`, we use "morph-or-mention" matching features.

If $\text{match}(d_e, d_s) =$ `false` we then attempt to perform matching based on synonyms of the words involved in the two dependencies (the "second tier" of our two-tiered strategy). Recall that $S_{d_e.d}$ is the set of synonyms for the word at index $d_e.d$. Since we do not perform word sense disambiguation, $S_{d_e.d}$ is the union of all possible synsets for $w_{d_e.d}$. We then define the following function for determining if two dependency pairs match at the synonym level:

$$\text{synmatch}(d_e, d_s) = \quad (3)$$
$$(S_{d_e.d} \cap S_{d_s.d} \neq \emptyset) \wedge (S_{d_e.a} \cap S_{d_s.a} \neq \emptyset).$$

This function returns `true` iff the pair of descendants share at least one synonym and the pair of ancestors share at least one synonym. If there is a synonym match, we use synonym-matching features.

3.3 Dependency matching features

The same sorts of features are produced whether there is a "morph-or-mention" match or a synonym match; however, we still distinguish the two types of features, so that the model may learn different weights according to what type of matching happened. The two matching situations each produce four types of features. Figure 2 shows these four types of features using the event of "Abdul Halim Khaddam resigns as Vice President of Syria" and the sentence "The resignation of Khaddam was abrupt" as an example. In particular, the "depth" features attempt to capture the "importance" the dependency match, as measured by the depth of the ancestor in the event dependency tree.

We have one additional type of feature: we compute the following kernel function on the two sets of dependencies E'_e and E'_s and create features based on quantizing the value:

$$K(E'_e, E'_s) = \quad (4)$$
$$\sum_{(d_e,d_s)\in E'_e \times E'_s \,:\, \text{match}(d_e,d_s)} (\Delta(d_e) \cdot \Delta(d_s))^{-1},$$

$\Delta((i, j))$ being the path distance in τ from node i to j.

4 Data and experiments

We created 159 queries to test this model framework. We adapted a publicly-available search engine (citation omitted) to retrieve documents automatically from the GALE corpus likely to be relevant to the event queries, and then used a set of simple heuristics—a subset of the low-level features described in §2—to retrieve sentences that were more likely than not to be relevant. We then had our most experienced annotator annotate sentences with five possible tags: `relevant`, `irrelevant`, `relevant-in-context`, `irrelevant-in-context` and `garbage` (to deal with sentences that were unintelligible "word salad").[4] Crucially, the annotation guidelines for this task were that an event had to be explicitly mentioned in a sentence in order for that sentence to be tagged `relevant`.

We separated the data roughly into an 80/10/10 split for training, devtest and test. We then trained our event-matching model solely on the examples marked `relevant` or `irrelevant`, of which there were 3546 instances. For all the experiments reported, we tested on our development test set, which comprised 465 instances that had been marked `relevant` or `irrelevant`.

We trained the kernel version of an averaged perceptron model (Freund and Schapire, 1999), using a polynomial kernel with degree 4 and additive term 1. As a baseline, we trained and tested a model using only the lexical-matching features. We then trained and tested models using only the low-level features and all features. Figure 3 shows the performance statistics of all three models, and Figure 4 shows the ROC curves of these models. Clearly, the dependency features help; at our normal operating point of 0, F-measure rises from 62.2 to 66.5. Looking solely

[4] The `*-in-context` tags were to be able to re-use the data for an upstream system capable of handling the GALE distillation query type "list facts about [event]".

Feature type	Example	Comment
Morph bigram	`x-resign-Khaddam`	Sparse, but helpful.
Tag bigram	`x-VBZ-NNP`	
Nonterminal	`x-VP-NP`	All pairs from $N_i \times N_j$ for $(i, j) \in E'_e$.
Depth	`x-eventArgHeadDepth=0`	Depth is 0 because "resigns" is root of event.

Figure 2: Types of dependency features. Example features are for e = "Abdul Halim Khaddam resigns as Vice President of Syria" and s = "The resignation of Khaddam was abrupt." In example features, `x` $\in$ {`m`, `s`}, depending on whether the dependency match was due to "morph-or-mention" matching or synonym matching.

Model	R	P	F
lex	36.6	76.3	49.5
low-level	63.9	60.5	62.2
all	**69.1**	**64.1**	**66.5**

Figure 3: Performance of models.

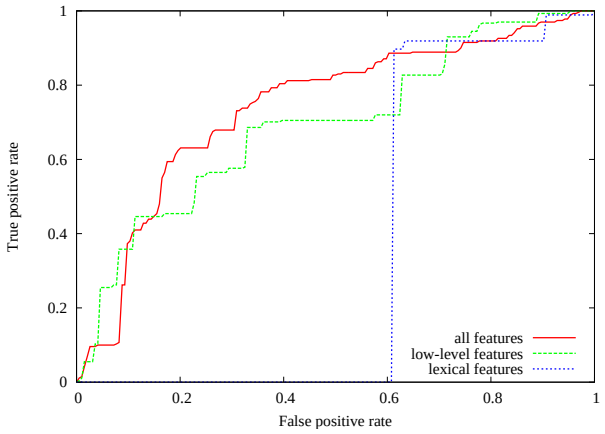


Figure 4: ROC curves of model with only low-level features vs. model with all features.

at pairs of predictions, McNemar's test reveals differences ($p \ll 0.05$) between the predictions of the baseline model and the other two models, but not between those of the low-level model and the model trained with all features.

5 Discussion

There have been several efforts to incorporate dependency information into a question-answering system. These have attempted to define either *ad hoc* similarity measures or a tree transformation process, whose parameters must be learned. By using the transitive closure of the dependency relation, we believe that—especially in the face of a small data set—we have struck a balance between the representative power of dependencies and the need to remain agnostic with respect to similarity measures or formalisms; we merely let the features speak for themselves and have the training procedure of a robust classifier learn the appropriate weights.

Acknowledgements

This work supported by DARPA grant HR0011-06-02-0001. Special thanks to Radu Florian and Jeffrey Sorensen for their helpful comments.

References

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, Alessandro Tommasi, Ellen M. Voorhees, and D. K. Harman. 2001. Selectively using relations to improve precision in question answering. In *TREC-10*, Gaithersburg, Maryland.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *SIGIR 2005*, Salvador, Brazil, August.

Radu Florian, Hani Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicholas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *HLT-NAACL 2004*, pages 1–8.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *COLING-ACL 2006*, Sydney, Australia.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007*, pages 22–32.

A Linguistically Annotated Reordering Model for BTG-based Statistical Machine Translation

Deyi Xiong, Min Zhang, Aiti Aw and Haizhou Li
Human Language Technology
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
{dyxiong, mzhang, aaiti, hli}@i2r.a-star.edu.sg

Abstract

In this paper, we propose a linguistically annotated reordering model for BTG-based statistical machine translation. The model incorporates linguistic knowledge to predict orders for both syntactic and non-syntactic phrases. The linguistic knowledge is automatically learned from source-side parse trees through an annotation algorithm. We empirically demonstrate that the proposed model leads to a significant improvement of 1.55% in the BLEU score over the baseline reordering model on the NIST MT-05 Chinese-to-English translation task.

1 Introduction

In recent years, Bracketing Transduction Grammar (BTG) proposed by (Wu, 1997) has been widely used in statistical machine translation (SMT). However, the original BTG does not provide an effective mechanism to predict the most appropriate orders between two neighboring phrases. To address this problem, Xiong et al. (2006) enhance the BTG with a maximum entropy (MaxEnt) based reordering model which uses boundary words of bilingual phrases as features. Although this model outperforms previous unlexicalized models, it does not utilize any linguistically syntactic features, which have proven useful for phrase reordering (Wang et al., 2007). Zhang et al. (2007) integrates source-side syntactic knowledge into a phrase reordering model based on BTG-style rules. However, one limitation of this method is that it only reorders syntactic phrases because linguistic knowledge from parse trees is only carried by syntactic phrases as far as reordering is concerned, while non-syntactic phrases are combined monotonously with a flat reordering score.

In this paper, we propose a linguistically annotated reordering model for BTG-based SMT, which is a significant extension to the work mentioned above. The new model annotates each BTG node with linguistic knowledge by projecting source-side parse trees onto the corresponding binary trees generated by BTG so that syntactic features can be used for phrase reordering. Different from (Zhang et al., 2007), our annotation algorithm is able to label both syntactic and non-syntactic phrases. This enables our model to reorder any phrases, not limited to syntactic phrases. In addition, other linguistic information such as head words, is also used to improve reordering.

The rest of the paper is organized as follows. Section 2 briefly describes our baseline system while Section 3 introduces the linguistically annotated reordering model. Section 4 reports the experiments on a Chinese-to-English translation task. We conclude in Section 5.

2 Baseline SMT System

The baseline system is a phrase-based system which uses the BTG lexical rules ($A \rightarrow x/y$) to translate source phrase x into target phrase y and the BTG merging rules ($A \rightarrow [A, A]|\langle A, A\rangle$) to combine two neighboring phrases with a straight or inverted order. The BTG lexical rules are weighted with several features, such as phrase translation, word penalty and language models, in a log-linear form. For the merging rules, a MaxEnt-based reordering model using boundary words of neighboring phrases as features is used to predict the merging order, similar to (Xiong et al., 2006). We call this reordering model

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 149–152,
Columbus, Ohio, USA, June 2008.

boundary words based reordering model (BWR). In this paper, we propose to incorporate a linguistically annotated reordering model into the log-linear translation model, so as to strengthen the BWR's phrase reordering ability. We train all the model scaling factors on the development set to maximize the BLEU score. A CKY-style decoder is developed to generate the best BTG binary tree for each input sentence, which yields the best translation.

3 Linguistically Annotated Reordering Model

The linguistically annotated reordering model (LAR) is a MaxEnt-based classification model which predicts the phrase order $o \in \{inverted, straight\}$ during the application of merging rules to combine their left and right neighboring phrases A_l and A_r into a larger phrase A. [1] The model can be formulated as

$$LAR = \frac{exp(\sum_i \theta_i h_i(o, A_l, A_r, A))}{\sum_{o'} exp(\sum_i \theta_i h_i(o', A_l, A_r, A))} \quad (1)$$

where the functions $h_i \in \{0, 1\}$ are reordering features and θ_i are weights of these features. We define the features as linguistic elements which are annotated for each BTG node through an annotation algorithm, which comprise (1) head word hw, (2) the part-of-speech (POS) tag ht of head word and (3) syntactic label sl.

Each merging rule involves 3 nodes (A, A_l, A_r) and each node has 3 linguistic elements (hw, ht, sl). Therefore, the model has 9 features in total. Taking the left node A_l as an example, the model could use its head word w as feature as follows

$$h_i(o, A, A_l, A_r) = \begin{cases} 1, & A_l.hw = w, o = straight \\ 0, & otherwise \end{cases}$$

3.1 Annotation Algorithm

There are two steps to annotate a phrase or a BTG node using source-side parse tree information: (1) determining the span on the source side which is exactly covered by the node or the phrase, then (2) annotating the span according to the source-side parse tree. If the span is exactly covered by a single subtree in the source-side parse tree, it is called **syntactic span**, otherwise it is **non-syntactic span**. One of the challenges in this annotation algorithm is that phrases (BTG nodes) are not always covering syntactic span, in other words, they are not always aligned to all constituent nodes in the source-side tree. To solve this problem, we use heuristic rules to generate pseudo head word and **composite label** which consists of syntactic labels of three relevant constituents for the non-syntactic span. In this way, our annotation algorithm is capable of labelling both syntactic and non-syntactic phrases and therefore providing linguistic information for any phrase reordering.

1: Annotator (span $s = \langle i, j \rangle$, source-side parse tree t)
2: **if** s is a syntactic span **then**
3: Find the subtree c in t which exactly covers s
4: $s.\{\ \} := \{c.hw, c.ht, c.sl\}$
5: **else**
6: Find the smallest subtree c^* subsuming s in t
7: **if** $c^*.hw \in s$ **then**
8: $s.hw := c^*.hw$ and $s.ht := c^*.ht$
9: **else**
10: Find the word $w \in s$ which is nearest to $c^*.hw$
11: $s.hw := w$ and $s.ht := w.t$ /*$w.t$ is the POS tag of w*/
12: **end if**
13: Find the left boundary node ln of s in c^*
14: Find the right boundary node rn of s in c^*
15: $s.sl := ln.sl\text{-}c^*.sl\text{-}rn.sl$
16: **end if**

Figure 1: The Annotation Algorithm.

The annotation algorithm is shown in Fig. 1. For a syntactic span, the annotation is trivial. Annotation elements directly come from the subtree that covers the span exactly. For a non-syntactic span, the process is much complicated. Firstly, we need to locate the smallest subtree c^* subsuming the span (line 6). Secondly, we try to identify the head word/tag of the span (line 7-12) by using its head word directly if it is within the span. Otherwise, the word within the span which is nearest to hw will be assigned as the head word of the span. Finally, we determine the composite label of the span (line 13-15), which is formulated as L-C-R. L/R means the syntactic label of the left/right **boundary node** of s which is the highest leftmost/rightmost sub-node of c^* not overlapping the span. If there is no such boundary node

[1] Each phrase is also a node in the BTG tree generated by the decoder.

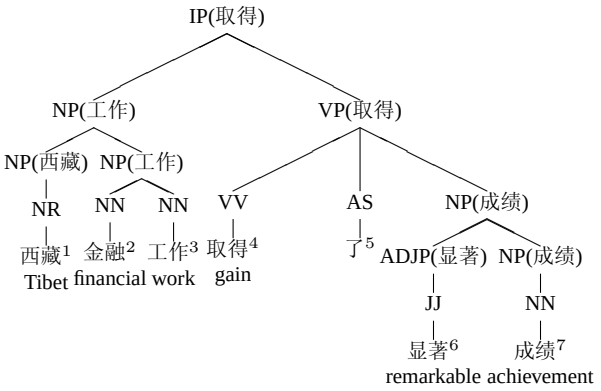


Figure 2: A syntactic parse tree with head word annotated for each internal node. The superscripts of leaf nodes denote their surface positions from left to right.

$span$	hw	ht	sl
$\langle 1, 2\rangle$	金融	NN	NULL-NP-NN
$\langle 2, 3\rangle$	工作	NN	NP
$\langle 2, 4\rangle$	取得	VV	NP-IP-NP
$\langle 3, 4\rangle$	取得	VV	NP-IP-NP

Table 1: Annotation samples according to the tree shown in Fig. 2. hw/ht represents the head word/tag, respectively. sl means the syntactic label.

(the span s is exactly aligned to the left/right boundary of c^*), L/R will be set to NULL. C is the label of c^*. L, R and C together define the external syntactic context of s.

Fig. 2 shows a syntactic parse tree for a Chinese sentence, with head word annotated for each internal node. Some sample annotations are given in Table 1.

3.2 Training and Decoding

Training an LAR model takes three steps. Firstly, we extract annotated reordering examples from source-side parsed, word-aligned bilingual data using the annotation algorithm and the reordering example extraction algorithm of (Xiong et al., 2006). We then generate features using linguistic elements of these examples and finally estimate feature weights. This training process flexibly learns rich syntactic reordering information without explicitly constructing BTG tree or forest for each sentence pair.

During decoding, each input source sentence is firstly parsed to obtain its syntactic tree. Then the CKY-style decoder tries to generate the best BTG tree using the lexical and merging rules. When two neighboring nodes are merged in a specific order, the two embedded reordering models, BWR and LAR, evaluate this merging independently with individual scores. The former uses boundary words as features while the latter uses the linguistic elements as features, annotated on the BTG nodes through the annotation algorithm according to the source-side parse tree.

4 Experiments

All experiments in this section were carried out on the Chinese-to-English translation task of the NIST MT-05. The baseline system and the new system with the LAR model were trained on the FBIS corpus. We removed 15,250 sentences, for which the Chinese parser (Xiong et al., 2005) failed to produce syntactic parse trees. The parser was trained on the Penn Chinese Treebank with a F1 score of 79.4%. The remaining FBIS corpus (224,165 sentence pairs) was used to obtain standard bilingual phrases for the systems.

We extracted 2.8M reordering examples from these sentences. From these examples, we generated 114.8K reordering features for the BWR model using the right boundary words of phrases and 85K features for the LAR model using linguistic annotations. We ran the MaxEnt toolkit (Zhang, 2004) to tune reordering feature weights with iteration number being set to 100 and Gaussian prior to 1 to avoid overfitting.

We built our four-gram language model using Xinhua section of the English Gigaword corpus (181.1M words) with the SRILM toolkit (Stolcke, 2002). For the efficiency of minimum-error-rate training (Och, 2003), we built our development set (580 sentences) using sentences not exceeding 50 characters from the NIST MT-02 evaluation test data.

4.1 Results

We compared various reordering configurations in the baseline system and new system. The baseline system only has BWR as the reordering model, while the new system employs two reordering models: BWR and LAR. For the linguistically annotated reordering model LAR, we augment its feature pool incrementally: firstly using only single labels

[2](SL) as features (132 features in total), then constructing composite labels for non-syntactic phrases (+BNL) (6.7K features), and finally introducing head words and their POS tags into the feature pool (+BNL+HWT) (85K features). This series of experiments demonstrate the impact and degree of contribution made by each feature for reordering. We also conducted experiments to investigate the effect of restricting reordering to syntactic phrases in the new system using the best reordering feature set (SL+BNL+HWT) for LAR. The experimental results (case-sensitive BLEU scores together with confidence intervals) are presented in Table 2, from which we have the following observations:

(1) The LAR model improves the performance statistically significantly. Even we only use the baseline feature set SL with only 132 features for the LAR, the BLEU score improves from 0.2497 to 0.2588. This is because most of the frequent reordering patterns between Chinese and English have been captured using syntactic labels. For example, the pre-verbal modifier PP in Chinese is translated into post-verbal counterpart in English. This reordering can be described by a rule with an inverted order: $VP \rightarrow \langle PP, VP \rangle$, and captured by our syntactic reordering features.

(2) Context information, provided by labels of boundary nodes (BNL) and head word/tag pairs (HWT), also improves phrase reordering. Producing composite labels for non-syntactic BTG nodes (+BNL) and integrating head word/tag pairs into the LAR as reordering features (+BNL+HWT) are both effective, indicating that context information complements syntactic label for capturing reordering patterns.

(3) Restricting phrase reordering to syntactic phrases is harmful. The BLEU score plummets from 0.2652 to 0.2512.

Reordering Configuration	BLEU (%)
BWR	24.97 ± 0.90
BWR + LAR (SL)	25.88 ± 0.95
BWR + LAR (+BNL)	26.27 ± 0.98
BWR + LAR (+BNL+HWT)	26.52 ± 0.96
Only allowed SPs reordering	25.12 ± 0.87

Table 2: The effect of the linguistically annotated reordering model. BWR denotes the boundary word based reordering model while LAR denotes the linguistically annotated reordering model. (SL) is the baseline feature set, (+BNL) and (+BNL+HWT) are extended feature sets for the LAR. SP means *syntactic phrase*.

5 Conclusion

In this paper, we have presented a linguistically annotated reordering model to effectively integrate linguistic knowledge into phrase reordering by merging source-side parse trees with BTG binary trees. Our experimental results show that, on the NIST MT-05 task of Chinese-to-English translation, the proposed reordering model leads to BLEU improvement of 1.55%. We believe that our linguistically annotated reordering model can be further improved by using better annotation which transfers more knowledge (morphological, syntactic or semantic) to the model.

[2]For non-syntactic node, we only use the single label C, without constructing composite label L-C-R.

References

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL 2003*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 901-904.

Chao Wang, Michael Collins and Philipp Koehn. 2007. Chinese Syntactic Reordering for Statistical Machine Translation. In *Proceedings of EMNLP-CoNLL 2007*.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377-403.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, Yueliang Qian. 2005. Parsing the Penn Chinese Treebank with Semantic Knowledge. In *Proceedings of IJCNLP*, Jeju Island, Korea.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of ACL-COLING 2006*.

Dongdong Zhang, Mu Li, Chi-Ho Li and Ming Zhou. 2007. Phrase Reordering Model Integrating Syntactic Knowledge for SMT. In *Proceedings of EMNLP-CoNLL 2007*.

Le Zhang. 2004. Maximum Entropy Modeling Tooklkit for Python and C++. Available at http://homepages.inf.ed.ac.uk/s0450736 /maxent_toolkit.html.

Segmentation for English-to-Arabic Statistical Machine Translation

Ibrahim Badr **Rabih Zbib** **James Glass**
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{iab02, rabih, glass}@csail.mit.edu

Abstract

In this paper, we report on a set of initial results for English-to-Arabic Statistical Machine Translation (SMT). We show that morphological decomposition of the Arabic source is beneficial, especially for smaller-size corpora, and investigate different recombination techniques. We also report on the use of Factored Translation Models for English-to-Arabic translation.

1 Introduction

Arabic has a complex morphology compared to English. Words are inflected for gender, number, and sometimes grammatical case, and various clitics can attach to word stems. An Arabic corpus will therefore have more surface forms than an English corpus of the same size, and will also be more sparsely populated. These factors adversely affect the performance of Arabic↔English Statistical Machine Translation (SMT). In prior work (Lee, 2004; Habash and Sadat, 2006), it has been shown that morphological segmentation of the Arabic source benefits the performance of Arabic-to-English SMT. The use of similar techniques for English-to-Arabic SMT requires recombination of the target side into valid surface forms, which is not a trivial task.

In this paper, we present an initial set of experiments on English-to-Arabic SMT. We report results from two domains: text news, trained on a large corpus, and spoken travel conversation, trained on a significantly smaller corpus. We show that segmenting the Arabic target in training and decoding improves performance. We propose various schemes for recombining the segmented Arabic, and compare their effect on translation. We also report on applying Factored Translation Models (Koehn and Hoang, 2007) for English-to-Arabic translation.

2 Previous Work

The only previous work on English-to-Arabic SMT that we are aware of is by Sarikaya and Deng (2007). It uses shallow segmentation, and does not make use of contextual information. The emphasis of that work is on using Joint Morphological-Lexical Language Models to rerank the output.

Most of the related work, though, is on Arabic-to-English SMT. Lee (2004) uses a trigram language model to segment Arabic words. She then proceeds to deleting or merging some of the segmented morphemes in order to make the segmented Arabic source align better with the English target. Habash and Sadat (2006) use the Arabic morphological analyzer MADA (Habash and Rambow, 2005) to segment the Arabic source; they propose various segmentation schemes. Both works show that the improvements obtained from segmentation decrease as the corpus size increases. As will be shown later, we observe the same trend, which is due to the fact that the model becomes less sparse with more training data.

There has been work on translating from English to other morphologically complex languages. Koehn and Hoang (2007) present Factored Translation Models as an extension to phrase-based statistical machine translation models. Factored models allow the integration of additional morphological fea-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 153–156,
Columbus, Ohio, USA, June 2008.

tures, such as POS, gender, number, etc. at the word level on both source and target sides. The tighter integration of such features was claimed to allow more explicit modeling of the morphology, and is better than using pre-processing and post-processing techniques. Factored Models demonstrate improvements when used to translate English to German or Czech.

3 Arabic Segmentation and Recombination

As mentioned in Section 1, Arabic has a relatively rich morphology. In addition to being inflected for gender, number, voice and case, words attach to various clitics for conjunction (*w+ 'and'*)[1], the definite article (*Al+ 'the'*), prepositions (e.g. *b+ 'by/with', l+ 'for', k+ 'as'*), possessive pronouns and object pronouns (e.g. *+ny 'me/my', +hm 'their/them'*). For example, the verbal form *wsnsAEdhm* and the nominal form *wbsyAratnA* can be decomposed as follows:

(1) a. w+ s+ n+ sAEd +hm
and+ will+ we+ help +them

b. w+ b+ syAr +At +nA
and+ with+ car +PL +our

Also, Arabic is usually written without the diacritics that denote the short vowels, and different sources write a few characters inconsistently. These issues create word-level ambiguity.

3.1 Arabic Pre-processing

Due to the word-level ambiguity mentioned above, but more generally, because a certain string of characters can, in principle, be either an affixed morpheme or part of the base word, morphological decomposition requires both word-level linguistic information and context analysis; simple pattern matching is not sufficient to detect affixed morphemes. To perform pre-translation morphological decomposition of the Arabic source, we use the morphological analyzer MADA. MADA uses SVM-based classifiers for features (such as POS, number and gender, etc.) to choose among the different analyses of a given word in context.

We first normalize the Arabic by changing final *'Y'* to *'y'* and the various forms of *Alif hamza* to bare *Alif*. We also remove diacritics wherever they occur. We then apply one of two morphological decomposition schemes before aligning the training data:

1. **S1**: Decliticization by splitting off each conjunction clitic, particle, definite article and pronominal clitic separately. Note that plural and subject pronoun morphemes are not split.

2. **S2**: Same as S1, except that the split clitics are glued into one prefix and one suffix, such that any given word is split into at most three parts: *prefix+ stem +suffix.*

For example the word *wlAwlAdh* (*'and for his kids'*) is segmented to *w+ l+ AwlAd +P:3MS* according to S1, and to *wl+ AwlAd +P:3MS* according to S2.

3.2 Arabic Post-processing

As mentioned above, both training and decoding use segmented Arabic. The final output of the decoder must therefore be recombined into a surface form. This proves to be a non-trivial challenge for a number of reasons:

1. Morpho-phonological Rules: For example, the feminine marker *'p'* at the end of a word changes to *'t'* when a suffix is attached to the word. So *syArp +P:1S* recombines to *syArty ('my car')*

2. Letter Ambiguity: The character *'Y'* (*Alf mqSwrp*) is normalized to *'y'*. In the recombination step we need to be able to decide whether a final *'y'* was originally a *'Y'*. For example, *mdy +P:3MS* recombines to *mdAh 'its extent'*, since the *'y'* is actually a *Y*; but *fy +P:3MS* recombines to *fyh 'in it'*.

3. Word Ambiguity: In some cases, a word can recombine into 2 grammatically correct forms. One example is the optional insertion of *nwn AlwqAyp* (*protective 'n'*), so the segmented word *lkn +O:1S* can recombine to either *lknny* or *lkny*, both grammatically correct.

To address these issues, we propose two recombination techniques:

1. **R**: Recombination rules defined manually. To resolve word ambiguity we pick the grammatical form that appears more frequently in the

[1] In this paper, Arabic text is written using Buckwalter transliteration

training data. To resolve letter ambiguity we use a unigram language model trained on data where the character *'Y'* had not been normalized. We decide on the non-normalized from of the *'y'* by comparing the unigram probability of the word with *'y'* to its probability with *'Y'*.

2. **T**: Uses a table derived from the training set that maps the segmented form of the word to its original form. If a segmented word has more than one original form, one of them is picked at random. The table is useful in recombining words that are split erroneously. For example, *qrDAy*, a proper noun, gets incorrectly segmented to *qrDAn +P:1S* which makes its recombination without the table difficult.

3.3 Factored Models

For the Factored Translation Models experiment, the factors on the English side are the POS tags and the surface word. On the Arabic side, we use the surface word, the stem and the POS tag concatenated to the segmented clitics. For example, for the word *wlAwlAdh ('and for his kids')*, the factored words are *AwlAd* and *w+l+N+P:3MS*. We use two language models: a trigram for surface words and a 7-gram for the POS+clitic factor. We also use a generation model to generate the surface form from the stem and POS+clitic, a translation table from POS to POS+clitics and from the English surface word to the Arabic stem. If the Arabic surface word cannot be generated from the stem and POS+clitic, we back off to translating it from the English surface word.

4 Experiments

The English source is aligned to the segmented Arabic target using GIZA++ (Och and Ney, 2000), and the decoding is done using the phrase-based SMT system MOSES (MOSES, 2007). We use a maximum phrase length of 15 to account for the increase in length of the segmented Arabic. Tuning is done using Och's algorithm (Och, 2003) to optimize weights for the distortion model, language model, phrase translation model and word penalty over the BLEU metric (Papineni et al., 2001). For our baseline system the tuning reference was non-segmented Arabic. For the segmented Arabic experiments we experiment with 2 tuning schemes: **T1** uses segmented Arabic for reference, and **T2** tunes on non-segmented Arabic. The Factored Translation Models experiments uses the MOSES system.

Scheme	Training Set	Tuning Set
Baseline	34.6%	36.8%
R	4.04%	4.65%
T	N/A	22.1%
T + R	N/A	1.9%

Table 1: Recombination Results. Percentage of sentences with mis-combined words.

4.1 Data Used

We experiment with two domains: text news and spoken dialogue from the travel domain. For the news training data we used corpora from LDC[2]. After filtering out sentences that were too long to be processed by GIZA ($>$ 85 words) and duplicate sentences, we randomly picked 2000 development sentences for tuning and 2000 sentences for testing. In addition to training on the full set of 3 million words, we also experimented with subsets of 1.6 million and 600K words. For the language model, we used 20 million words from the LDC Arabic Gigaword corpus plus 3 million words from the training data. After experimenting with different language model orders, we used 4-grams for the baseline system and 6-grams for the segmented Arabic. The English source is downcased and the punctuations are separated. The average sentence length is 33 for English, 25 for non-segmented Arabic and 36 for segmented Arabic.

For the spoken language domain, we use the IWSLT 2007 Arabic-English (Fordyce, 2007) corpus which consists of a 200,000 word training set, a 500 sentence tuning set and a 500 sentence test set. We use the Arabic side of the training data to train the language model and use trigrams for the baseline system and a 4-grams for segmented Arabic. The average sentence length is 9 for English, 8 for Arabic, and 10 for segmented Arabic.

[2]Since most of the data was originally intended for Arabic-to-English translation our test and tuning sets have only one reference

4.2 Recombination Results

To test the different recombination schemes described in Section 3.2, we run these schemes on the training and development sets of the news data, and calculate the percentage of sentences with recombination errors (Note that, on average, there is one mis-combined word per mis-combined sentence). The scores are presented in Table 1. The baseline approach consists of gluing the prefix and suffix without processing the stem. **T + R** means that the words seen in the training set were recombined using scheme **T** and the remainder were recombined using scheme **R**. In the remaining experiments we use the scheme **T + R**.

4.3 Translation Results

The 1-reference BLEU score results for the news corpus are presented in Table 2; those for IWSLT are in Table 3. We first note that the scores are generally lower than those of comparable Arabic-to-English systems. This is expected, since only one reference was used to evaluate translation quality and since translating to a more morphologically complex language is a more difficult task, where there is a higher chance of translating word inflections incorrectly. For the news corpus, the segmentation of Arabic helps but the gain diminishes as the training data size increases, since the model becomes less sparse. This is consistent with the larger gain obtained from segmentation for IWSLT. The segmentation scheme **S2** performs slightly better than **S1**. The tuning scheme **T2** performs better for the news corpus, while **T1** is better for the IWSLT corpus. It is worth noting that tuning without segmentation hurts the score for IWSLT, possibly because of the small size of the training data. Factored models perform better than our approach with the large training corpus, although at a significantly higher cost in terms of time and required resources.

	Large	**Medium**	**Small**
Training Size	3M	1.6M	0.6M
Baseline	26.44	20.51	17.93
S1 + T1 tuning	26.46	21.94	20.59
S1 + T2 tuning	26.81	21.93	20.87
S2 + T1 tuning	26.86	21.99	20.44
S2 + T2 tuning	27.02	22.21	20.98
Factored Models + tuning	27.30	21.55	19.80

Table 2: BLEU (1-reference) scores for the News data.

	No Tuning	**T1**	**T2**
Baseline	26.39	24.67	
S1	29.07	29.82	
S2	29.11	30.10	28.94

Table 3: BLEU (1-reference) scores for the IWSLT data.

5 Conclusion

In this paper, we showed that making the Arabic match better to the English through segmentation, or by using additional translation model factors that model grammatical information is beneficial, especially for smaller domains. We also presented several methods for recombining the segmented Arabic target. Our results suggest that more sophisticated techniques, such as syntactic reordering, should be attempted.

Acknowledgments

We would like to thank Ali Mohammad, Michael Collins and Stephanie Seneff for their valuable comments.

References

Cameron S. Fordyce 2007. *Overview of the 2007 IWSLT Evaluation Campaign* . In Proc. of IWSLT 2007.

Nizar Habash and Owen Rambow, 2005. *Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop*. In Proc. of ACL.

Nizar Habash and Fatiha Sadat, 2006. *Arabic Preprocessing Schemes for Statistical Machine Translation*. In Proc. of HLT.

Philipp Koehn and Hieu Hoang, 2007. *Factored Translation Models*. In Proc. of EMNLP/CNLL.

Young-Suk Lee, 2004. *Morphological Analysis for Statistical Machine Translation*. In Proc. of EMNLP.

MOSES, 2007. *A Factored Phrase-based Beam-search Decoder for Machine Translation*. URL: http://www.statmt.org/moses/.

Franz Och, 2003. *Minimum Error Rate Training in Statistical Machine Translation*. In Proc. of ACL.

Franz Och and Hermann Ney, 2000. *Improved Statistical Alignment Models*. In Proc. of ACL.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, 2001. *Bleu: a Method for Automatic Evaluation of Machine Translation*. In Proc. of ACL.

Ruhi Sarikaya and Yonggang Deng 2007. *Joint Morphological-Lexical Language Modeling for Machine Translation*. In Proc. of NAACL HLT.

Exploiting N-best Hypotheses for SMT Self-Enhancement

Boxing Chen, Min Zhang, Aiti Aw and **Haizhou Li**
Department of Human Language Technology
Institute for Infocomm Research
21 Heng Mui Keng Terrace, 119613, Singapore
{bxchen, mzhang, aaiti, hli}@i2r.a-star.edu.sg

Abstract

Word and n-gram posterior probabilities estimated on N-best hypotheses have been used to improve the performance of statistical machine translation (SMT) in a rescoring framework. In this paper, we extend the idea to estimate the posterior probabilities on N-best hypotheses for translation phrase-pairs, target language n-grams, and source word reorderings. The SMT system is self-enhanced with the posterior knowledge learned from N-best hypotheses in a re-decoding framework. Experiments on NIST Chinese-to-English task show performance improvements for all the strategies. Moreover, the combination of the three strategies achieves further improvements and outperforms the baseline by 0.67 BLEU score on NIST-2003 set, and 0.64 on NIST-2005 set, respectively.

1 Introduction

State-of-the-art Statistical Machine Translation (SMT) systems usually adopt a two-pass search strategy. In the first pass, a decoding algorithm is applied to generate an N-best list of translation hypotheses; while in the second pass, the final translation is selected by rescoring and re-ranking the N-best hypotheses through additional feature functions. In this framework, the N-best hypotheses serve as the candidates for the final translation selection in the second pass.

These N-best hypotheses can also provide useful feedback to the MT system as the first decoding has discarded many undesirable translation candidates. Thus, the knowledge captured in the N-best hypotheses, such as *posterior probabilities* for words, n-grams, phrase-pairs, and source word reorderings, etc. is more compatible with the source sentences and thus could potentially be used to improve the translation performance.

Word posterior probabilities estimated from the N-best hypotheses have been widely used for confidence measure in automatic speech recognition (Wessel, 2002) and have also been adopted into machine translation. Blatz et al. (2003) and Ueffing et al. (2003) used word posterior probabilities to estimate the confidence of machine translation. Chen et al. (2005), Zens and Ney (2006) reported performance improvements by computing target n-grams posterior probabilities estimated on the N-best hypotheses in a rescoring framework. Transductive learning method (Ueffing et al., 2007) which repeatedly re-trains the generated source-target N-best hypotheses with the original training data again showed translation performance improvement and demonstrated that the translation model can be reinforced from N-best hypotheses.

In this paper, we further exploit the potential of the N-best hypotheses and propose several schemes to derive the posterior knowledge from the N-best hypotheses, in an effort to enhance the language model, translation model, and source word reordering under a re-decoding framework of any phrase-based SMT system.

2 Self-Enhancement with Posterior Knowledge

The self-enhancement system structure is shown in Figure 1. Our baseline system is set up using Moses (Koehn et al., 2007), a state-of-the-art phrase-base SMT open source package. In the followings, we detail the approaches to exploiting the three different kinds of posterior knowledge, namely, language model, translation model and word reordering.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 157–160,
Columbus, Ohio, USA, June 2008.

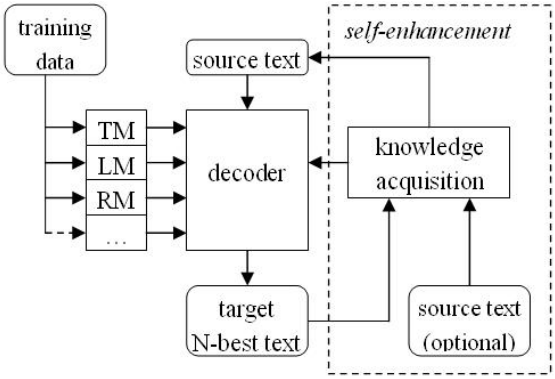


Figure 1: Self-enhancement system structure, where TM is translation model, LM is language model, and RM is reordering model.

2.1 Language Model

We consider self-enhancement of language model as a language model adaptation problem similar to (Nakajima et al., 2002). The original monolingual target training data is regarded as general-domain data while the test data as a domain-specific data. Obviously, the real domain-specific target data (test data) is unavailable for training. In this work, the N-best hypotheses of the test set are used as a quasi-corpus to train a language model. This new language model trained on the quasi-corpus is then used together with the language model trained on the general-domain data (original training data) to produce a new list of N-best hypotheses under our self-enhancement framework. The feature function of the language model $h_{LM}(f_1^J, e_1^I)$ is a mixture model of the two language models as in Equation 1.

$$h_{LM}(f_1^J, e_1^I) = \lambda_1 h_{TLM}(e_1^I) + \lambda_2 h_{QLM}(e_1^I) \quad (1)$$

where f_1^J is the source language words string, e_1^I is the target language words string, *TLM* is the language model trained on target training data, and *QLM* is on the quasi-corpus of N-best hypotheses.

The mixture model exploits multiple language models with weights λ_1 and λ_2 being optimized together with other feature functions. The procedure for self-enhancement of the language model is as follows.

1. Run decoding and extract N-best hypotheses.
2. Train a new language model (*QLM*) on the N-best hypotheses.
3. Optimize the weights of the decoder which uses both original LM (*TLM*) and the new LM (*QLM*).
4. Repeat step 1-3 for a fixed number of iterations.

2.2 Translation Model

In general, we can safely assume that for a given source input, phrase-pairs that appeared in the N-best hypotheses are better than those that did not. We call the former "good phrase-pairs" and the later "bad phrase-pairs" for the given source input. Hypothetically, we can reinforce the translation model by appending the "good phrase-pairs" to the original phrase table and changing the probability space of the translation model, as phrase-based translation probabilities are estimated using relative frequencies. The new direct phrase-based translation probabilities are computed as follows:

$$p(\tilde{e} \mid \tilde{f}) = \frac{N_{train}(\tilde{f}, \tilde{e}) + N_{nbest}(\tilde{f}, \tilde{e})}{N_{train}(\tilde{f}) + N_{nbest}(\tilde{f})} \quad (2)$$

where $\tilde{f}$ is the source language phrase, $\tilde{e}$ is the target language phrase, $N_{train}(.)$ is the frequencies observed in the training data, and $N_{nbest}(.)$ is the frequencies observed in the N-best hypotheses. For those phrase-pairs that did not appear in the N-best hypotheses list ("bad phrase-pairs"), $N_{nbest}(\tilde{f}, \tilde{e})$ equals 0, but the marginal count of $\tilde{f}$ is increased by $N_{nbest}(\tilde{f})$, in this way the phrase-based translation probabilities of "bad phrase-pairs" degraded when compared with the corresponding probabilities in the original translation model, and that of "good phrase-pairs" increased, hence improve the translation model.

The procedure for translation model self-enhancement can be summarized as follows.

1. Run decoding and extract N-best hypotheses.
2. Extract "good phrase-pairs" according to the hypotheses' phrase-alignment information and append them to the original phrase table to generate a new phrase table.
3. Score the new phrase table to create a new translation model.
4. Optimize the weights of the decoder with the above new translation model.
5. Repeat step 1-4 for a fixed number of iterations.

2.3 Word Reordering

Some previous work (Costa-jussà and Fonollosa, 2006; Li et al., 2007) have shown that reordering a source sentence to match the word order in its cor-

responding target sentence can produce better translations for a phrase-based SMT system. We bring this idea forward to our word reordering self-enhancement framework, which similarly translates a source sentence (*S*) to target sentence (*T*) in two stages: $S \rightarrow S' \rightarrow T$, where S' is the reordered source sentence.

The phrase-alignment information in each hypothesis indicates the word reordering for source sentence. We select the word reordering with the highest posterior probability as the best word reordering for a given source sentence. Word reorderings from different phrase segmentation but with same word surface order are merged. The posterior probabilities of the word re-orderings are computed as in Equation 3.

$$p(r_1^J \mid f_1^J) = \frac{N(r_1^J)}{N_{hyp}} \quad (3)$$

where $N(r_1^J)$ is the count of word reordering r_1^J, and N_{hyp} is the number of N-best hypotheses.

The words of the source sentence are then reordered according to their indices in the best selected word reordering r_1^J. The procedure for self-enhancement of word reordering is as follows.

1. Run decoding and extract N-best hypotheses.
2. Select the best word re-orderings according to the phrase-alignment information.
3. Reorder the source sentences according to the selected word reordering.
4. Optimize the weights of the decoder with the reordered source sentences.
5. Repeat step 1-4 for a fixed number of iterations.

3 Experiments and Results

Experiments on Chinese-to-English NIST translation tasks were carried out on the FBIS[1] corpus. We used NIST 2002 MT evaluation test set as our development set, and the NIST 2003, 2005 test sets as our test sets as shown in Table 1.

We determine the number of iteration empirically by setting it to 10. We then observe the BLEU score on the development set for each iteration. The iteration number which achieved the best BLEU score on development set is selected as the iteration number of iterations for the test set.

[1] LDC2003E14

Data set	type	#Running words	
		Chinese	English
train	parallel	7.0M	8.9M
	monolingual	-	61.5M
NIST 02	dev	23.2K	108.6K
NIST 03	test	25.8K	116.5K
NIST 05	test	30.5K	141.9K

Table 1: Statistics of training, dev and test sets. Evaluation sets of NIST campaigns include 4 references: total numbers of running words are provided in the table.

System	#iter.	NIST 02	NIST 03	NIST 05
Base	-	27.67	26.68	24.82
TM	4	27.87	26.95	25.05
LM	6	27.96	27.06	25.07
WR	6	27.99	27.04	25.11
Comb	7	**28.45**	**27.35**	**25.46**

Table 2: BLEU% scores of five systems: decoder (Base), self-enhancement on translation model (TM), language model (LM), word reordering (WR) and the combination of TM, LM and WR (Comb).

Further experiments also suggested that, in this experiment scenario, setting the size of N-best list to 3,000 arrives at the greatest performance improvements. Our evaluation metric is BLEU (Papineni et al., 2002). The translation performance is reported in Table 2, where the column "#iter." refers to the iteration number where the system achieved the best BLEU score on development set.

Compared with the baseline ("Base" in Table 2), all three self-enhancement methods ("TM", "LM", and "WR" in Table 2) consistently improved the performance. In general, absolute gains of 0.23-0.38 BLEU score were obtained for each method on two test sets. While comparing the performance among all three methods, we can see that they achieved very similar improvement. Combining the three methods showed further gains in BLEU score. Totally, the combined system outperformed the baseline by 0.67 BLEU score on NIST'03, and 0.64 on NIST'05 test set, respectively.

4 Discussion

As posterior knowledge applied in our models are *posterior probabilities*, the main difference between our work and all previous work is the use of knowledge source, where we derive knowledge from the N-best hypotheses generated from previous iteration.

Comparing the work of (Nakajima et al., 2002), there is a slight difference between the two models. Nakajima et al. used only 1-best hypothesis, while we use N-best hypotheses of test set as the quasi-corpus to train the language model.

In the work of (Costa-jussà and Fonollosa, 2006; Li et al., 2007) which similarly translates a source sentence (*S*) to target sentence (*T*) in two stages: $S \rightarrow S' \rightarrow T$, they derive S' from training data; while we obtain S' based on the occurrence frequency, i.e. posterior probability of each source word reordering in the N-best hypotheses list.

An alternative solution for enhancing the translation model is through self-training (Ueffing, 2006; Ueffing et al., 2007) which re-trains the source-target N-best hypotheses together with the original training data, and thus differs from ours in the way of new phrase pairs extraction. We only supplement those phrase-pairs appeared in the N-best hypotheses to the original phrase table. Further experiment showed that improvement obtained by self-training method is not as consistent on both development and test sets as that by our method. One possible reason is that in self-training, the entire translation model is adjusted with the addition of new phrase-pairs extracted from the source-target N-best hypotheses, and hence the effect is less predictable.

5 Conclusions

To take advantage of the N-best hypotheses, we proposed schemes in a re-decoding framework and made use of the posterior knowledge learned from the N-best hypotheses to improve a phrase-based SMT system. The posterior knowledge include posterior probabilities for target n-grams, translation phrase-pairs and source word re-orderings, which in turn improve the language model, translation model, and word reordering respectively.

Experiments were based on the state-of-the-art phrase-based decoder and carried out on NIST Chinese-to-English task. It has been shown that all three methods improved the performance. Moreover, the combination of all three strategies outperforms each individual method and significantly outperforms the baseline. We demonstrated that the SMT system can be self-enhanced by exploiting useful feedback from the N-best hypotheses which are generated by itself.

References

J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. 2003. Confidence estimation for machine translation. *Final report, JHU/CLSP Summer Workshop.*

B. Chen, R. Cattoni, N. Bertoldi, M. Cettolo and M. Federico. 2005. The ITC-irst SMT System for IWSLT-2005. In *Proceeding of IWSLT-2005*, pp.98-104, Pittsburgh, USA, October.

M. R. Costa-jussà, J. A. R. Fonollosa. 2006. Statistical Machine Reordering. In *Proceeding of EMNLP 2006*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL-2007*, pp. 177-180, Prague, Czech Republic.

C.-H. Li, M. Li, D. Zhang, M. Li, M. Zhou and Y. Guan. 2007. A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation. In *Proceedings of ACL-2007*. Prague, Czech Republic.

H. Nakajima, H. Yamamoto, T. Watanabe. 2002. Language model adaptation with additional text generated by machine translation. In *Proceedings of COLING-2002*. Volume 1, Pages: 1-7. Taipei.

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

N. Ueffing. 2006. Using Monolingual Source-Language Data to Improve MT Performance. In *Proceedings of IWSLT 2006*. Kyoto, Japan. November 27-28.

N. Ueffing, K. Macherey, and H. Ney. 2003. Confidence Measures for Statistical Machine Translation. In *Proceeding of MT Summit IX*, pages 394–401, New Orleans, LA, September.

N. Ueffing, G. Haffari, A. Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of ACL-2007*, Prague.

F. Wessel. 2002. Word Posterior Probabilities for Large Vocabulary Continuous Speech Recognition. Ph.D. thesis, RWTH Aachen University. Aachen, Germany, January.

R. Zens and H. Ney. 2006. N-gram Posterior Probabilities for Statistical Machine Translation. In *Proceedings of the HLT-NAACL Workshop on SMT*, pp. 72-77, NY.

Partial Matching Strategy for Phrase-based Statistical Machine Translation

Zhongjun He[1,2] and **Qun Liu**[1] and **Shouxun Lin**[1]
[1]Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, 100190, China
[2]Graduate University of Chinese Academy of Sciences
Beijing, 100049, China
{zjhe,liuqun,sxlin}@ict.ac.cn

Abstract

This paper presents a partial matching strategy for phrase-based statistical machine translation (PBSMT). Source phrases which do not appear in the training corpus can be translated by word substitution according to partially matched phrases. The advantage of this method is that it can alleviate the data sparseness problem if the amount of bilingual corpus is limited. We incorporate our approach into the state-of-the-art PBSMT system Moses and achieve statistically significant improvements on both small and large corpora.

1 Introduction

Currently, most of the phrase-based statistical machine translation (PBSMT) models (Marcu and Wong, 2002; Koehn et al., 2003) adopt full matching strategy for phrase translation, which means that a phrase pair $(\widetilde{f}, \widetilde{e})$ can be used for translating a source phrase $\bar{f}$, only if $\widetilde{f} = \bar{f}$. Due to lack of generalization ability, the full matching strategy has some limitations. On one hand, the data sparseness problem is serious, especially when the amount of the bilingual data is limited. On the other hand, for a certain source text, the phrase table is redundant since most of the bilingual phrases cannot be fully matched.

In this paper, we address the problem of translation of *unseen phrases*, the source phrases that are not observed in the training corpus. The alignment template model (Och and Ney, 2004) enhanced phrasal generalizations by using words classes rather than the words themselves. But the phrases are overly generalized. The hierarchical phrase-based model (Chiang, 2005) used hierarchical phrase pairs to strengthen the generalization ability of phrases and allow long distance reorderings. However, the huge grammar table greatly increases computational complexity. Callison-Burch et al. (2006) used paraphrases of the trainig corpus for translating unseen phrases. But they only found and used the semantically similar phrases. Another method is to use multi-parallel corpora (Cohn and Lapata, 2007; Utiyama and Isahara, 2007) to improve phrase coverage and translation quality.

This paper presents a partial matching strategy for translating unseen phrases. When encountering unseen phrases in a source sentence, we search partially matched phrase pairs from the phrase table. Then we keep the translations of the matched part and translate the unmatched part by word substitution. The advantage of our approach is that we alleviate the data sparseness problem without increasing the amount of bilingual corpus. Moreover, the partially matched phrases are not necessarily synonymous. We incorporate the partial matching method into the state-of-the-art PBSMT system, Moses. Experiments show that, our approach achieves statistically significant improvements not only on small corpus, but also on large corpus.

2 Partial Matching for PBSMT

2.1 Partial Matching

We use *matching similarity* to measure how well the source phrases match each other. Given two source phrases $\widetilde{f}_1^J$ and $\widetilde{f'}_1^J$, the matching similarity is computed as:

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 161–164,
Columbus, Ohio, USA, June 2008.

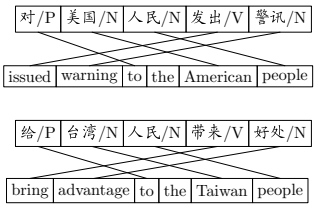


Figure 1: An example of partially matched phrases with the same POS sequence and word alignment.

$$SIM(\widetilde{f}_1^J, \widetilde{f'}_1^J) = \frac{\sum_{j=1}^{J} \delta(f_j, f'_j)}{J} \quad (1)$$

where,

$$\delta(f, f') = \begin{cases} 1 & \text{if } f = f' \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Therefore, partial matching takes full matching ($SIM(\widetilde{f}, \bar{f}) = 1.0$) as a special case. Note that in order to improve search efficiency, we only consider the partially matched phrases with the same length.

In our experiments, we use a matching threshold α to tune the precision of partial matching. Low threshold indicates high coverage of unseen phrases, but will suffer from much noise. In order to alleviate this problem, we search partially matched phrases under the constraint that they must have the same parts-of-speech (POS) sequence. See Figure 1 for illustration. Although the matching similarity of the two phrases is only 0.2, as they have the same POS sequence, the word alignments are the same. Therefore, the lower source phrase can be translated according to the upper phrase pair with correct word reordering. Furthermore, this constraint can sharply decrease the computational complexity since there is no need to search the whole phrase table.

2.2 Translating Unseen Phrases

We translate an unseen phrase f_1^J according to the partially matched phrase pair ($f'^J_1, e'^I_1, \widetilde{a}$) as follows:

1. Compare each word between f_1^J and f'^J_1 to get the position set of the different words: $P = \{j | f_j \neq f'_j, j = 1, 2, \ldots, J\}$;
2. Remove f'_j from f'^J_1 and e'_{a_j} from e'^I_1, where $j \in P$;
3. Find the translation e for $f_j (j \in P)$ from the phrase table and put it into the position a_j in e'^I_1 according to the word alignment $\widetilde{a}$.

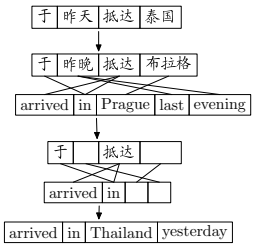


Figure 2: An example of phrase translation.

Figure 2 shows an example. In fact, we create a translation template dynamically in step 2:

$$\langle \text{于 X}_1 \text{ 抵达 X}_2, \text{arrived in X}_2 \text{ X}_1 \rangle \quad (3)$$

Here, on the source side, each of the non-terminal X corresponds to a single source word. In addition, the removed sub-phrase pairs should be consistent with the word alignment matrix.

Following conventional PBSMT models, we use 4 features to measure phrase translation quality: the translation weights $p(\widetilde{f}|\widetilde{e})$ and $p(\widetilde{e}|\widetilde{f})$, the lexical weights $p_w(\widetilde{f}|\widetilde{e})$ and $p_w(\widetilde{e}|\widetilde{f})$. The new constructed phrase pairs keep the translation weights of their "parent" phrase pair. The lexical weights are computed by word substitution. Suppose $S\{(f', e')\}$ is the pair set in ($\widetilde{f'}$,$\widetilde{e'}$,$\widetilde{a}$) which replaced by $S\{(f, e)\}$ to create the new phrase pair ($\widetilde{f}$,$\widetilde{e}$,$\widetilde{a}$), the lexical weight is computed as:

$$p_w(\widetilde{f}|\widetilde{e}, \widetilde{a}) = \frac{p_w(\widetilde{f'}|\widetilde{e'}, \widetilde{a}) \times \prod_{(f,e) \in S\{(f,e)\}} p_w(f|e)}{\prod_{(f',e') \in S\{(f',e')\}} p_w(f'|e')} \quad (4)$$

Therefore, the newly constructed phrase pairs can be used for decoding as they have already existed in the phrase table.

2.3 Incorporating Partial Matching into the PBSMT Model

In this paper, we incorporate the partial matching strategy into the state-of-the-art PBSMT system, Moses[1]. Given a source sentence, Moses firstly uses the full matching strategy to search all possible *translation options* from the phrase table, and then uses a beam-search algorithm for decoding.

[1]http://www.statmt.org/moses/

Therefore, we do incorporation by performing partial matching for phrase translation before decoding. The advantage is that the main search algorithm need not be changed.

For a source phrase $\widetilde{f}$, we search partially matched phrase pair ($\widetilde{f}'$, $\widetilde{e}'$, $\widetilde{a}$) from the phrase table. If $SIM(\widetilde{f}, \widetilde{f}')$=1.0, which means $\widetilde{f}$ is observed in the training corpus, thus $\widetilde{e}'$ can be directly stored as a translation option. However, if $\alpha \leq SIM(\widetilde{f}, \widetilde{f}') < 1.0$, we construct translations for $\widetilde{f}$ according to Section 2.2. Then the newly constructed translations are stored as translation options.

Moses uses translation weights and lexical weights to measure the quality of a phrase translation pair. For partial matching, besides these features, we add matching similarity $SIM(\widetilde{f}, \widetilde{f}')$ as a new feature. For a source phrase, we select top N translations for decoding. In Moses, N is set by the pruning parameter *ttable-limit*.

3 Experiments

We carry out experiments on Chinese-to-English translation on two tasks: **Small-scale task**, the training corpus consists of 30k sentence pairs (840K + 950K words); **Large-scale task**, the training corpus consists of 2.54M sentence pairs (68M + 74M words). The 2002 NIST MT evaluation test data is used as the development set and the 2005 NIST MT test data is the test set. The baseline system we used for comparison is the state-of-the-art PBSMT system, Moses.

We use the ICTCLAS toolkit[2] to perform Chinese word segmentation and POS tagging. The training script of Moses is used to train the bilingual corpus. We set the maximum length of the source phrase to 7, and record word alignment information in the phrase table. For the language model, we use the SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram model on the Xinhua portion of the Gigaword corpus.

To run the decoder, we set ttable-limit=20, distortion-limit=6, stack=100. The translation quality is evaluated by BLEU-4 (case-sensitive). We perform minimum-error-rate training (Och, 2003) to tune the feature weights of the translation model to maximize the BLEU score on development set.

[2]http://www.nlp.org.cn/project/project.php?proj_id=6

α	1.0	0.7	0.5	0.3	0.1
BLEU	24.44	24.43	24.86	**25.31**	25.13

Table 1: Effect of matching threshold on BLEU score.

3.1 Small-scale Task

Table 1 shows the effect of matching threshold on translation quality. The baseline uses full matching (α=1.0) for phrase translation and achieves a BLEU score of 24.44. With the decrease of the matching threshold, the BLEU scores increase. when α=0.3, the system obtains the highest BLEU score of 25.31, which achieves an absolute improvement of 0.87 over the baseline. However, if the threshold continue decreasing, the BLEU score decreases. The reason is that low threshold increases noise for partial matching.

The effect of matching threshold on the coverage of n-gram phrases is shown in Figure 3. When using full matching (α=1.0), long phrases (length$\geq$3) face a serious data sparseness problem. With the decrease of the threshold, the coverage increases.

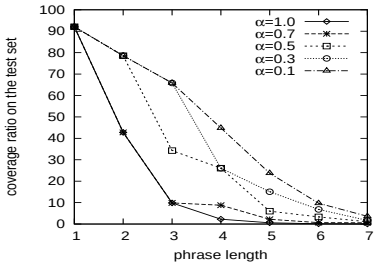


Figure 3: Effect of matching threshold on the coverage of n-gram phrases.

Table 2 shows the phrase number of 1-best output under α=1.0 and α=0.3. When α=1.0, the long phrases (length$\geq$3) only account for 2.9% of the total phrases. When α=0.3, the number increases to 10.7%. Moreover, the total phrase of α=0.3 is less than that of α=1.0, since source text is segmented into more long phrases under partial matching, and most of the long phrases are translated from partially matched phrases (the row *0.3*$\leq SIM <$*1.0*).

3.2 Large-scale Task

For this task, the BLEU score of the baseline is 30.45. However, for partial matching method with

Phrase Length		1	2	3	4	5	6	7	total
α=1.0		19485	4416	615	87	12	2	1	24618
α=0.3	SIM=1.0	14750	2977	387	48	10	1	0	21195
	0.3$\leq SIM <$1.0	0	1196	1398	306	93	17	12	

Table 2: Phrase number of 1-best output. α=1.0 means full matching. For α=0.3, SIM=1.0 means full matching, $0.3 \leq SIM < 1.0$ means partial matching.

α=0.5[3], the BLEU score is 30.96, achieving an absolute improvement of 0.51. Using Zhang's significant tester (Zhang et al., 2004), both the improvements on the two tasks are statistically significant at $p < 0.05$.

The improvement on large-scale task is less than that on small-scale task since larger corpus relieves data sparseness. However, the partial matching approach can also improve translation quality by using long phrases. For example, the segmentation and translation for the Chinese sentence "但是 经济 产出 的 长期 趋势 将" are as follows:

Full matching:
长期 | 经济 产出 | 但是 | 的 | 趋势 | 将
long term | *economic output* | , but | *the* | *trend* | will

Partial matching:
但是 | 经济 产出 的 长期 趋势 | 将
but | *the long-term trend of economic output* | will

Here the source phrase "经济 产出 的 长期 趋势" cannot be fully matched. Thus the decoder breaks it into 4 short phrases, but performs an incorrect reordering. Using partial matching, the long phrase is translated correctly since it can partially matched the phrase pair "*经济 发展 的 必然 趋势, the inevitable trend of economic development*".

3.3 Conclusion

This paper presents a partial matching strategy for phrase-based statistical machine translation. Phrases which are not observed in the training corpus can be translated according to partially matched phrases by word substitution. Our method can relieve data sparseness problem without increasing the amount of the corpus. Experiments show that our approach achieves statistically significant improvements over the state-of-the-art PBSMT system Moses.

In future, we will study sophisticated partial matching methods, since current constraints are excessively strict. Moreover, we will study the effect of word alignment on partial matching, which may affect word substitution and reordering.

[3]Due to time limit, we do not tune the threshold for large-scale task.

Acknowledgments

We would like to thank Yajuan Lv and Yang Liu for their valuable suggestions. This work was supported by the National Natural Science Foundation of China (NO. 60573188 and 60736014), and the High Technology Research and Development Program of China (NO. 2006AA010108).

References

C. Callison-Burch, P. Koehn, and M. Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proc. of NAACL06*, pages 17–24.

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

T. Cohn and M. Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proc. of ACL07*, pages 728–735.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL03*, pages 127–133.

D. Marcu and W. Wong. 2002. A phrasebased joint probabilitymodel for statistical machine translation. In *Proc. of EMNLP02*, pages 133–139.

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

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

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

M. Utiyama and H. Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Proc. of NAACL-HLT07*, pages 484–491.

Y. Zhang, S. Vogel, and A. Waibel. 2004. Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *Proc. of LREC04*, pages 2051–2054.

Unsupervised Learning of Acoustic Sub-word Units

Balakrishnan Varadarajan* and **Sanjeev Khudanpur*** **Emmanuel Dupoux**
Center for Language and Speech Processing Laboratoire de Science Cognitive
Johns Hopkins University et Psycholinguistique
Baltimore, MD 21218 75005, Paris, France
{bvarada2 , khudanpur}@jhu.edu emmanuel.dupoux@gmail.com

Abstract

Accurate unsupervised learning of phonemes of a language directly from speech is demonstrated via an algorithm for joint unsupervised learning of the topology and parameters of a hidden Markov model (HMM); states and short state-sequences through this HMM correspond to the learnt sub-word units. The algorithm, originally proposed for unsupervised learning of allophonic variations within a given phoneme set, has been adapted to learn without any knowledge of the phonemes. An evaluation methodology is also proposed, whereby the state-sequence that aligns to a test utterance is transduced in an automatic manner to a phoneme-sequence and compared to its manual transcription. Over 85% phoneme recognition accuracy is demonstrated for speaker-dependent learning from fluent, large-vocabulary speech.

1 Automatic Discovery of Phone(me)s

Statistical models learnt from data are extensively used in modern automatic speech recognition (ASR) systems. Transcribed speech is used to estimate conditional models of the acoustics given a phoneme-sequence. The phonemic pronunciation of words and the *phonemes* of the language, however, are derived almost entirely from linguistic knowledge. In this paper, we investigate whether the phonemes may be learnt automatically from the speech signal.

Automatic learning of phoneme-like units has significant implications for theories of language acquisition in babies, but our considerations here are somewhat more technological. We are interested in developing ASR systems for languages or dialects for which such linguistic knowledge is scarce or nonexistent, *and* in extending ASR techniques to recognition of signals other than speech, such as manipulative gestures in endoscopic surgery. Hence an algorithm for automatically learning an inventory of intermediate symbolic units—intermediate relative to the acoustic or kinematic signal on one end and the word-sequence or surgical act on the other—is very desirable.

Except for some early work on isolated word/digit recognition (Paliwal and Kulkarni, 1987; Wilpon et al., 1987, etc), not much attention has been paid to automatic derivation of sub-word units from speech, perhaps because pronunciation lexicons are now available[1] in languages of immediate interest. What *has* been investigated is automatically learning allophonic variations of each phoneme due to co-articulation or contextual effects (Takami and Sagayama, 1992; Fukada et al., 1996); the phoneme inventory is usually assumed to be known.

The general idea in allophone learning is to begin with an inventory of only one allophone per phoneme, and incrementally refine the inventory to better fit the speech signal. Typically, each phoneme is modeled by a separate HMM. In early stages of refinement, when very few allophones are available, it is hoped that "similar" allophones of a phoneme will be modeled by shared HMM states, and that subsequent refinement will result in distinct states for different allophones. The key therefore is to devise a scheme for successive refinement of a model shared by many allophones. In the HMM setting, this amounts to simultaneously refining the *topology* and the model *parameters*. A successive state splitting (SSS) algorithm to achieve this was proposed by Takami and Sagayama (1992), and en-

* This work was partially supported by National Science Foundation Grants No IIS-0534359 and OISE-0530118.

[1] See http://www.ldc.upenn.edu/Catalog/byType.jsp

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 165–168,
Columbus, Ohio, USA, June 2008.

hanced by Singer and Ostendorf (1996). Improvements in phoneme recognition accuracy using these derived allophonic models over phonemic models were obtained.

In this paper, we investigate directly learning the allophone inventory of a language from speech without recourse to its phoneme set. We begin with a one-state HMM for all speech sounds and modify the SSS algorithm to successively learn the topology and parameters of HMMs with even larger numbers of states. States sequences through this HMM are expected to correspond to allophones. The most likely *state-sequence* for a speech segment is interpreted as an "allophonic labeling" of that speech by the learnt model. Performance is measured by mapping the resultant state-sequence to phonemes.

One contribution of this paper is a significant improvement in the efficacy of the SSS algorithm as described in Section 2. It is based on observing that the improvement in the goodness of fit by up to two consecutive splits of any of the current HMM states can be evaluated *concurrently and efficiently*. Choosing the best subset of splits from among these is then cast as a constrained knapsack problem, to which an efficient solution is devised. Another contribution of this paper is a method to evaluate the accuracy of the resulting "allophonic labeling," as described in Section 3. It is demonstrated that if a small amount of phonetically transcribed speech is used to learn a Markov (bigram) model of state-sequences that arise from each phone, an evaluation tool results with which we may measure phone recognition accuracy, even though the HMM labels the speech signal not with phonemes but merely a state-sequence. Section 4 presents experimental results, where the performance accuracies with different learning setups are tabulated. We also see how as little as 5 minutes of speech is adequate for learning the acoustic units.

2 An Improved and Fast SSS Algorithm

The improvement of the SSS algorithm of Takami and Sagayama (1992), renamed ML-SSS by Singer and Ostendorf (1996), proceeds roughly as follows.

1. Model all the speech[2] using a 1-state HMM with a *diagonal-covariance* Gaussian. (N=1.)

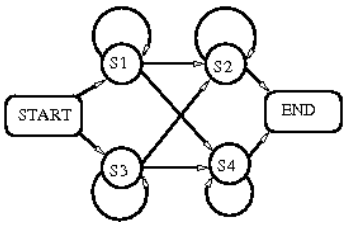


Figure 1: Modified four-way split of a state s.

2. For each HMM state s, compute the gain in log-likelihood (LL) of the speech by either a contextual or a temporal split of s into two states s_1 and s_2. Among the N states, select and and split the one that yields the most gain in LL.
3. If the gain is above a threshold, retain the split and set $N = N + 1$; furthermore, if N is less than desired, re-estimate all parameters of the new HMM, and go to Step 2.

Note that the key computational steps are the for-loop of Step 2 and the re-estimation of Step 3.

Modifications to the ML-SSS Algorithm: We made the following modifications that are favorable in terms of greater speed and larger search space, thereby yielding a gain in likelihood that is potentially greater than the original ML-SSS.

1. Model all the speech using a 1-state HMM with a *full-covariance* Gaussian density. Set $N = 1$.
2. Simultaneously replace each state s of the HMM with the 4-state topology shown in Figure 1, yielding a $4N$-state HMM. If the state s had parameters $(\mu_s, \mathbf{\Sigma}_s)$, then means of its 4-state replacement are $\mu_{s_1} = \mu_s - \delta = \mu_{s_4}$ and $\mu_{s_2} = \mu_s + \delta = \mu_{s_3}$, with $\delta = \epsilon\lambda^* v^*$, where λ^* and v^* are the principal eigenvalue and eigenvector of Σ_s and $0 < \epsilon \ll 1$ is typically 0.2.
3. Re-estimate all parameters of this (overgrown) HMM. Gather the Gaussian sufficient statistics for each of the $4N$ states from the last pass of re-estimation: the state occupancy π_{s_i}. The sample mean μ_{s_i}, and sample covariance Σ_{s_i}.
4. Each quartet of states (see Figure 1) that resulted from the same original state s can be

[2]Note that the original application of SSS was for learning the allophonic variations of a phoneme; hence the phrase "all the speech" meant all the speech corresponding *separately* to each phoneme. Here it really means all the speech.

merged back in different ways to produce 3, 2 or 1 HMM states. There are 6 ways to end up with 3 states, and 7 to end up with 2 states. Retain for further consideration the 4 state split of s, the best merge back to 3 states among the 6 ways, the best merge back to 2 states among the 7 ways, and the merge back to 1 state.

5. Reduce the number of states from $4N$ to $N+\Delta$ by *optimally*[3] merging back quartets that cause the least loss in log-likelihood of the speech.
6. Set $N = N + \Delta$. If N is less than the desired HMM size, retrain the HMM and go to Step 2.

Observe that the 4-state split of Figure 1 permits a slight look-ahead in our scheme in the sense that the goodness of a contextual or temporal split of two different states can be compared in the same iteration with two consecutive splits of a single state. Also, the split/merge statistics for a state are gathered in our modified SSS assuming that the other states have already been split, which facilitates consideration of concurrent state splitting. If $s_1, \ldots, s_m$ are merged into $\tilde{s}$, the loss of log-likelihood in Step 4 is:

$$\frac{d}{2}\sum_{i=1}^{m}\pi_{s_i}\log|\Sigma_{\tilde{s}}| - \frac{d}{2}\sum_{i=1}^{m}\pi_{s_i}\log|\Sigma_{s_i}|\,, \qquad (1)$$

where $\Sigma_{\tilde{s}} = \dfrac{\sum_{i=1}^{m}\pi_{s_i}\left(\Sigma_{s_i}+\mu_{s_i}\mu'_{s_i}\right)}{\sum_{i=1}^{m}\pi_{s_i}} - \mu_{\tilde{s}}\mu'_{\tilde{s}}$.

Finally, in selecting the best Δ states to add to the HMM, we consider many more ways of splitting the N original states than SSS does. E.g. going up from $N = 6$ to $N+\Delta = 9$ HMM states could be achieved by a 4-way split of a single state, a 3-way split of one state and 2-way of another, or a 2-way split of three distinct states; all of them are explored in the process of merging from $4N = 24$ down to 9 states. Yet, like SSS, no original state s is permitted to merge with another original state s'. This latter restriction leads to an $O(N^5)$ algorithm for finding the best states to merge down[4]. Details of the algorithm are ommited for the sake of brevity.

In summary, our modified ML-SSS algorithm can leap-frog by Δ states at a time, e.g. $\Delta = \alpha N$, compared to the standard algorithm, and it has the benefit of some lookahead to avoid greediness.

[3] This entails solving a constrained knapsack problem.

[4] This is a restricted version of the 0-1 knapsack problem.

3 Evaluating the Goodness of the Labels

The HMM learnt in Section 2 is capable of assigning state-labels to speech via the Viterbi algorithm. Evaluating whether these labels are linguistically meaningful requires *interpreting* the labels in terms of phonemes. We do so as follows.

Some phonetically transcribed speech is labeled with the learnt HMM, and the label sequences corresponding to each phone segment are extracted. Since the HMM was learnt from unlabeled speech, the labels and short label-sequences usually correspond to allophones, not phonemes. Therefore, for each *triphone*, i.e. each phone tagged with its left- and right-phone context, a simple *bigram* model of label sequences is estimated. An unweighted "phone loop" that accepts all phone sequences is created, and composed with these bigram models to create a label-to-phone transducer capable of mapping HMM label sequences to phone sequences.

Finally, the test speech (not used for HMM learning, nor for estimating the bigram model) is treated as having been "generated" by a source-channel model in which the label-to-phone transducer is the source—generating an HMM state-sequence—and the Gaussian densities of the learnt HMM states constitute the channel—taking the HMM state-sequence as the channel input and generating the observed speech signal as the output. Standard Viterbi decoding determines the most likely phone sequence for the test speech, and phone accuracy is measured by comparison with the manual phonetic transcription.

4 Experimental Results

4.1 Impact of the Modified State Splitting

The ML-SSS procedure estimates $2N$ different $N{+}1$-state HMMs to grow from N to $N{+}1$ states. Our procedure estimates *one* $4N$ state HMM to grow to $N{+}\Delta$, making it hugely faster for large N.

Table 1 compares the log-likelihood of the training speech for ML-SSS and our procedure. The results validate our modifications, demonstrating that at least in the regimes feasible for ML-SSS, there is no loss (in fact a tiny gain) in fitting the speech data, and a big gain in computational effort[5].

[5] ML-SSS with $\Delta{=}1$ was impractical beyond $N{=}22$.

# of states	SSS ($\Delta = 1$)	$\Delta = 3$	$\Delta = N$
8	-7.14	-7.13	-7.13
10	-7.08	-7.06	-7.06
22	-6.78	-6.76	N/A
40	N/A	-6.23	-6.20

Table 1: Aggressive state splitting does not cause any degradation in log-likelihood relative to ML-SSS.

4.2 Unsupervised Learning of Sub-word Units

We used about 30 minutes of phonetically transcribed Japanese speech from *one* speaker[6] provided by Maekawa (2003) for our unsupervised learning experiments. The speech was segmented via silence detection into 800 utterances, which were further partitioned into a 24-minute training set (80%) and 6-minute test set (20%).

Our first experiment was to learn an HMM from the training speech using our modified ML-SSS procedure; we tried $N = 22$, 70 and 376. For each N, we then labeled the training speech using the learnt HMM, used the phonetic transcription of the training speech to estimate label-bigram models for each triphone, and built the label-to-phone transducer as described in Section 3. We also investigated (i) using only 5 minutes of training speech to learn the HMM, but still labeling and using all 24 minutes to build the label-to-phone transducer, and (ii) setting aside 5 minutes of training speech to learn the transducer and using the rest to learn the HMM. For each learnt HMM+transducer pair, we phonetically labeled the test speech.

The results in the first column of Table 2 suggest that the sub-word units learnt by the HMM are indeed interpretable as phones. The second column suggests that a small amount of speech (5 minutes) may be adequate to learn these units consistently. The third column indicates that learning how to map the learnt (allophonic) units to phones requires relatively more transcribed speech.

4.3 Inspecting the Learnt Sub-word Units

The most frequent 3-, 4- and 5-state sequences in the automatically labeled speech consistently matched particular phones in specific articulatory contexts, as shown below, i.e. the HMM learns allophones.

[6]We heeded advice from the literature indicating that automatic methods model gross channel- and speaker-differences before capturing differences between speech sounds.

HMM	24 min	5 min	19 min
label-to-phone	24 min	24 min	5 min
27 states	71.4%	70.9%	60.2%
70 states	84.4%	84.7%	75.8%
376 states	87.2%	86.8%	76.6%

Table 2: Phone recognition accuracy for different HMM sizes (N), and with different amounts of speech used to learn the HMM labeler and the label-to-phone transducer.

HMM labels	L-contxt	Phone	R-contxt
11, 28, 32	*vowel*	t	[e|a|o]
15, 17, 2	[g|k]	[u|o]	[⋆]
3, 17, 2	[k|t|g|d]	a	[k|t|g|d]
31, 5, 13, 5	*vowel*	[s|sj|sy]	*vowel*
17, 2, 31, 11	[g|t|k|d]	[a|o]	[t|k]
3, 30, 22, 34	[⋆]	a	*silence*
6, 24, 8, 15, 22	[⋆]	o	*silence*
4, 3, 17, 2, 21	[k|t]	a	[k|t]
4, 17, 24, 2, 31	[s|sy|z]	o	[t|d]
	[t|d]	o	[s|sy|z]

For instance, the label sequence $3, 17, 2$, corresponds to an "a" surrounded by stop consonants {t, d, k, g}; further restricting the sequence to $4, 3, 17, 2, 21$, results in restricting the context to the unvoiced stops {t, k}. That such clusters are learnt without knowledge of phones is remarkable.

References

T. Fukada, M. Bacchiani, K. K. Paliwal, and Y. Sagisaka. 1996. Speech recognition based on acoustically derived segment units. In *ICSLP*, pages 1077–1080.

K. Maekawa. 2003. Corpus of spontaneous japanese: its design and evaluation. In *ISCA/IEEE Workshop on Spontaneous Speech Processing and Recognition*.

K. K. Paliwal and A. M. Kulkarni. 1987. Segmentation and labeling using vector quantization and its application in isolated word recognition. *Journal of the Acoustical Society of India*, 15:102–110.

H. Singer and M. Ostendorf. 1996. Maximum likelihood successive state splitting. In *ICASSP*, pages 601–604.

J. Takami and S. Sagayama. 1992. A successive state splitting algorithm for efficient allophone modeling. In *ICASSP*, pages 573–576.

J. G. Wilpon, B. H. Juang, and L. R. Rabiner. 1987. An investigation on the use of acoustic sub-word units for automatic speech recognition. In *ICASSP*, pages 821–824.

High Frequency Word Entrainment in Spoken Dialogue

Ani Nenkova
Dept. of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
nenkova@seas.upenn.edu

Agustín Gravano
Dept. of Computer Science
Columbia University
New York, NY 10027, USA
agus@cs.columbia.edu

Julia Hirschberg
Dept. of Computer Science
Columbia University
New York, NY 10027, USA
julia@cs.columbia.edu

Abstract

Cognitive theories of dialogue hold that entrainment, the automatic alignment between dialogue partners at many levels of linguistic representation, is key to facilitating both production and comprehension in dialogue. In this paper we examine novel types of entrainment in two corpora—Switchboard and the Columbia Games corpus. We examine entrainment in use of *high-frequency words* (the most common words in the corpus), and its association with dialogue naturalness and flow, as well as with task success. Our results show that such entrainment is predictive of the perceived naturalness of dialogues and is significantly correlated with task success; in overall interaction flow, higher degrees of entrainment are associated with more overlaps and fewer interruptions.

1 Introduction

When people engage in conversation, they adapt the way they speak to their conversational partner. For example, they often adopt a certain way of describing something based upon the way their conversational partner describes it, negotiating a common description, particularly for items that may be unfamiliar to them (Brennan, 1996). They also alter their amplitude, if the person they are speaking with speaks louder than they do (Coulston et al., 2002), or reuse syntactic constructions employed earlier in the conversation (Reitter et al., 2006). This phenomenon is known in the literature as entrainment, accommodation, adaptation, or alignment.

There is a considerable body of literature which posits that entrainment may be crucial to human perception of dialogue success and overall quality, as well as to participants' evaluation of their conversational partners. Pickering and Garrod (2004) propose that the automatic alignment at many levels of linguistic representation (lexical, syntactic and semantic) is key for both production and comprehension in dialogue, and facilitates interaction. Goleman (2006) also claims that a key to successful communication is human ability to synchronize their communicative behavior with that of their conversational partner. For example, in laboratory studies of non-verbal entrainment (mimicry of mannerisms and facial expressions between subjects and a confederate), Chartrand and Bargh (1999) found not only that subjects displayed a strong unintentional entrainment, but also that greater entrainment/mimicry led subjects to feel that they liked the confederate more and that the overall interaction was progressing more smoothly. People who had a high inclination for empathy (understanding the point of view of the other) entrained to a greater extent than others. Reitter et al. (2007) also found that degree of entrainment in lexical and syntactic repetitions that occurred in only the first five minutes of each dialogue significantly predicted task success in studies of the HCRC Map Task Corpus.

In this paper we examine a novel dimension of entrainment between conversation partners: the use of *high-frequency words*, the most frequent words in the dialogue or corpus. In Section 2 we describe experiments on high-frequency word entrainment and perceived dialogue naturalness in Switchboard dia-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 169–172,
Columbus, Ohio, USA, June 2008.

logues. The degree of high-frequency word entrainment predicts naturalness with an accuracy of 67% over a 50% baseline. In Section 3 we discuss experiments on the association of high-frequency word entrainment with task success and turn-taking. Results show that degree of high-frequency word entrainment is positively and significantly correlated with task success and proportion of overlaps in these dialogues, and negatively and significantly correlated with proportion of interruptions.

2 Predicting perceived naturalness

2.1 The Switchboard Corpus

The Switchboard Corpus (Godfrey et al., 1992) is a collection of recordings of spontaneous telephone conversations between speakers of many varieties of American English who were asked to discuss a preassigned topic from a set including favorite types of music or the new roles of women in society. The corpus consists of 2430 conversations with an average duration of 6 minutes, for a total of 240 hours and three million words. The corpus has been orthographically transcribed and annotated for degree of naturalness on Likert scales from 1 (very natural) to 5 (not natural at all).

2.2 Entrainment and perceived naturalness

Previous studies (Niederhoffer and Pennebaker, 2002) have suggested that adaptation in overall word count as well as words of particular parts of speech, or words associated with emotion or with various cognitive states, can predict the degree of coordination and engagement of conversational partners. Here, we examine conversational partners' similarity in high-frequency word usage in the Switchboard corpus as a predictor of the hand-annotated naturalness scores for their conversation. Using entrainment over the most frequent words in the entire corpus has the advantage of avoiding sparsity problems; we hypothesize that it will be more general and robust than attempting to measure lexical entrainment over the high-frequency words that occur in a particular conversation.

Our measure of entrainment $entr(w)$ is defined as the negated absolute value of the difference between the fraction of times a particular word w is used by the two speakers S_1 and S_2. More formally,

$$entr(w) = -\left|\frac{count_{S_1}(w)}{ALL_{S_1}} - \frac{count_{S_2}(w)}{ALL_{S_2}}\right|$$

Here, ALL_{S_i} is the number of all words uttered by speaker S_i in the given conversation, and $count_{S_i}(w)$ is the number of times S_i used word w.

The $entr(w)$ statistic was computed for the 100 most common words in the entire Switchboard corpus and feature selection was used to determine the 25 most predictive words used for later classification: *um, how, okay, go, I've, all, very, as, or, up, a, no, more, something, from, this, what, too, got, can, he, in, things, you, and*.

The data for the experiments was a balanced set of 250 conversations rated "1" (very natural) and 250 examples of problematic conversations with ratings of 3, 4 or 5. The accuracy of predicting the binary naturalness (ratings of 1 or 3-5) of each conversation from a logistic regression model is 63.76%, significantly over a 50% random baseline. This result confirms the hypothesis that entrainment in high-frequency word usage is a good indicator of the perceived naturalness of a conversation.

Some of our 25 high-frequency words are in fact *cue phrases*, which are important indicators of dialogue structure. This suggests that a more focused examination of this class of words might be useful.

3 Association with task success and dialogue flow

3.1 The Columbia Games Corpus

The Columbia Games Corpus (Benus et al., 2007) is a collection of 12 spontaneous task-oriented dyadic conversations elicited from native speakers of Standard American English. Subjects played a series of computer games requiring verbal communication between partners to achieve a common goal, either identifying matching cards appearing on each of their screens, or moving an object on one screen to the same location in which it appeared on the other, where each subject could see only their own screen. The games were designed to encourage frequent and natural conversation by engaging the subjects in competitive yet collaborative tasks. For example, players could receive points in the games in a variety of ways and had to negotiate the best strategy

for matching cards; in other games, they received more points if they could place objects in exactly the same location. Subjects were scored on each game and their overall score determined the additional monetary compensation they would receive. A total of 9h 8m ($\sim$73,800 words) of dialogue were recorded. All files in the corpus were orthographically transcribed and words were hand-aligned by trained annotators. A subset of the corpus was also labeled for different types of turn-taking behavior. These include *(i)* **smooth turn exchanges**—speaker S_2 takes the floor after speaker S_1 has completed her turn, with no overlap; *(ii)* **overlaps**—S_2 starts his turn before S_1 has completely finished her turn, but S_1 does complete her turn; *(iii)* **interruptions**—S_2 starts talking before S_1 completes her turn, and as a result S_1 does not complete her utterance. We used these annotations to study the association between entrainment and turn-taking behavior.

3.2 Entrainment and task success

In the Columbia Games Corpus, we hypothesize that the game score achieved by the participants is a good measure of the effectiveness of the dialogue. To determine the extent to which task success is related to the degree of entrainment in high-frequency word usage, we examined 48 dialogues. We computed the correlation coefficient between the game score (normalized by the highest achieved score for the game type) and two different ways of quantifying the degree of entrainment between the speakers (S_1 and S_2) in several word classes. In addition to overall high-frequency words, we looked at two subclasses of words often used in dialogue:

25MF-G The 25 most frequent words in the game.

25MF-C The 25 most frequent words over the entire corpus: *the, a, okay, and, of, I, on, right, is, it, that, have, yeah, like, in, left, it's, uh, so, top, um, bottom, with, you, to*.

ACW Affirmative cue words: *alright, gotcha, huh, mm-hm, okay, right, uh-huh, yeah, yep, yes, yup*. There are 5831 instances in the corpus (7.9% of all words).

FP Filled pauses: *uh, um, mm*. The corpus contains 1845 instances of filled pauses (2.5% of all tokens).

We generalize our measure of word entrainment $entr(w)$ to each of these *classes* of words c:

$$ENTR_1(c) = \sum_{w \in c} entr(w)$$

$ENTR_1$ ranges from 0 to $-\infty$, with 0 meaning perfect match on usage of lexical items in class c. An alternative measure of entrainment that we experimented with is defined as

$$ENTR_2(c) = -\frac{\sum_{w \in c} |count_{S_1}(w) - count_{S_2}(w)|}{\sum_{w \in c} (count_{S_1}(w) + count_{S_2}(w))}$$

The entrainment score defined in this way ranges from 0 to -1, with 0 meaning perfect match on lexical usage and -1 meaning perfect **mis**match.

The correlations between the normalized game score and these measures of entrainment are shown in Table 1. $ENTR_1$ for the 25 most frequent words, both corpus-wide and game-specific, is highly and significantly correlated with task success, with stronger results for game-specific words. For the

	$ENTR_1$		$ENTR_2$	
Word class	*cor*	*p*	*cor*	*p*
25MF-C	**0.341**	**0.018**	0.187	0.202
25MF-G	**0.376**	**0.008**	0.260	0.074
ACW	0.230	0.116	**0.372**	**0.009**
FP	-0.080	0.591	-0.007	0.964

Table 1: Pearson's correlation with game score.

filled pauses class, there is essentially no correlation between entrainment and task success, while for affirmative cue words there is association only under the $ENTR_2$ definition of entrainment. The difference in results between $ENTR_1$ and $ENTR_2$ suggests that the two measures of entrainment capture different aspects of dialogue coordination and that exploring various formulations of entrainment deserves future attention.

3.3 Dialogue coordination

The coordination of turn-taking in dialogue is especially important for successful interaction. Speech overlaps (O), might indicate a lively, highly coordinated conversation, with participants anticipating the end of their interlocutor's speaking turn. Smooth switches of turns (S) with no overlapping speech are also characteristic of good coordination, in cases where these are not accompanied by long pauses between turns. On the other hand, interruptions (I) and long inter-turn **latency** (L)—long simultaneous pauses by the speakers— are generally perceived as a sign of poorly coordinated dialogues.

To determine the relationship between entrainment and dialogue coordination, we examined the correlation between entrainment types and the proportion of interruptions, smooth switches and overlaps, for which we have manual annotations for a subset of 12 dialogues. We also looked at the correlation of entrainment with mean latency in each dialogue. Table 2 summarizes our major findings.

		cor	*p*
$ENTR_1$(25MF-C)	I	$-$**0.612**	**0.035**
$ENTR_1$(25MF-G)	I	$-$0.514	0.087
$ENTR_1$(ACW)	O	**0.636**	**0.026**
$ENTR_2$(ACW)	O	**0.606**	**0.037**
$ENTR_1$(FP)	O	**0.750**	**0.005**
$ENTR_2$(25MF-G)	O	**0.605**	**0.037**
$ENTR_2$(25MF-G)	S	$-$**0.663**	**0.019**
$ENTR_2$(ACW)	L	$-$**0.757**	**0.004**
$ENTR_2$(25MF-G)	L	$-$0.523	0.081

Table 2: Pearson's correlation with proportion of overlaps, interruptions, smooth switches, and mean latency.

The two measures that were significantly correlated with task success—$ENTR_1$(25MF-C) and $ENTR_1$(25MF-G)—also correlated *negatively* with the proportion of interruptions in the dialogue. This finding could have important implications for the development of spoken dialog systems (SDS). For example, a measure of entrainment might be used to anticipate the user's propensity to interrupt the system, signalling the need to change dialogue strategy. It also suggests that if the system entrains *to users* it might help to reduce such interruptions. While our study is of association, not causality, this suggests future areas of investigation.

Our other correlations reveal that turn exchanges characterized by overlaps are reliably associated with entrainment in usage of affirmative cue word, filled pauses and game-specific most frequent words. Long latency is negatively associated with entrainment in affirmative cue words and game-specific most frequent words. Overall, the more entrainment, the more engaged the participants and the better coordination there is between them, with shorter latencies and more overlaps.

Unexpectedly, smooth switches correlate negatively with entrainment in game-specific most frequent words. This result might be confounded by the presence of long latencies in some switches. While smooth switches are desirable, especially in SDS, long latencies between turns can indicate lack of coordination.

4 Conclusion

We present a corpus study relating dialogue naturalness, success and coordination with speaker entrainment on common words: most frequent words overall, most frequent words in a dialogue, filled pauses, and affirmative cue words. We find that degree of entrainment with respect to most frequent words can distinguish dialogues rated most natural from those rated less natural. Entrainment over classes of common words also strongly correlates with task success and highly engaged and coordinated turn-taking behavior. Entrainment over corpus-wide most frequent words significantly correlates with task success and minimal interruptions—important goals of SDS. In future work we will explore the consequences of system entrainment to SDS users in helping systems achieve these goals, and the use of simple measures of entrainment to modify dialogue strategies in order to decrease the occurrence of user interruptions.

Acknowledgments

This work was funded in part by NSF IIS-0307905.

References

S. Benus, A. Gravano, and J. Hirschberg. 2007. The prosody of backchannels in American English. *ICPhS'07*.

S.E. Brennan. 1996. Lexical entrainment in spontaneous dialog. *ISSD'96*.

T. Chartrand and J. Bargh. 1999. The chameleon effect: the perception-behavior link and social interaction. *J. of Personality & Social Psych.*, 76(6):893–910.

R. Coulston, S. Oviatt, and C. Darves. 2002. Amplitude convergence in children's conversational speech with animated personas. *ICSLP'02*.

J. Godfrey, E. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. *ICASSP'92*.

Daniel Goleman. 2006. *Social Intelligence*. Bantam.

K. Niederhoffer and J. Pennebaker. 2002. Linguistic style matching in social interaction.

M. J. Pickering and S. Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27:169–226.

D. Reitter and J. Moore. 2007. Predicting success in dialogue. *ACL'07*.

D. Reitter, F. Keller, and J.D. Moore. 2006. Computational Modelling of Structural Priming in Dialogue. *HLT-NAACL'06*.

Distributed Listening: A Parallel Processing Approach to Automatic Speech Recognition

Yolanda McMillian
3101 Shelby Center
Auburn University
Auburn, AL 36849-5347, USA
mcmilym@auburn.edu

Juan E. Gilbert
3101 Shelby Center
Auburn University
Auburn, AL 36849-5347, USA
gilbert@auburn.edu

Abstract

While speech recognition systems have come a long way in the last thirty years, there is still room for improvement. Although readily available, these systems are sometimes inaccurate and insufficient. The research presented here outlines a technique called Distributed Listening which demonstrates noticeable improvements to existing speech recognition methods. The Distributed Listening architecture introduces the idea of multiple, parallel, yet physically separate automatic speech recognizers called listeners. Distributed Listening also uses a piece of middleware called an interpreter. The interpreter resolves multiple interpretations using the Phrase Resolution Algorithm (PRA). These efforts work together to increase the accuracy of the transcription of spoken utterances.

1 Introduction

Research in the area of natural language processing has been on-going for over thirty years (Natural Language Software Registry, 2004; Jurafsky and Martin, 2000); however, there is still room for improvement with mainstream speech recognition systems (Deng, 2004). Distributed Listening will further research in this area. The concept is based around the idea of multiple speech input sources. Previous research activities involved a single microphone with multiple, separate recognizers that all yielded improvements in accuracy. Distributed Listening uses multiple, parallel speech recognizers, with each recognizer having its own input source (Gilbert, 2005). Each recognizer is a listener. Once input is collected from the listeners, one machine, the interpreter, processes all of the input (see figure 1). To process the input, a phrase resolution algorithm is used.

This approach is analogous to a crime scene with multiple witnesses (the listeners) and a detective (the interpreter) who pieces together the stories of the witnesses using his/her knowledge of crime scenes to form a hypothesis of the actual event. Each witness will have a portion of the story that is the same as the other witnesses. It is up to the detective to fill in the blanks. With Distributed Listening, the process is very similar. Each listener will have common recognition results and the interpreter will use the phrase resolution algorithm to resolve conflicts.

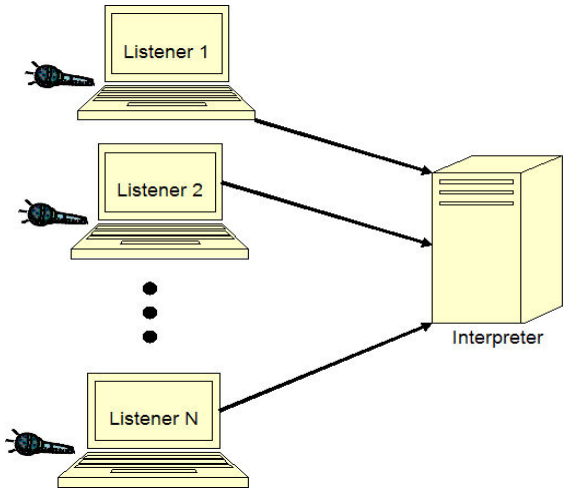


Figure 1. Distributed Listening Architecture

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 173–176,
Columbus, Ohio, USA, June 2008.

2 Background

Automatic speech recognition systems convert a speech signal into a sequence of words, usually based on the Hidden Markov Model (HMM), in which words are constructed from a sequence of states (Baum, 1972; Young et al., 1989; Young 1990; Furui, 2002).

There are several systems that used the HMM along with multiple speech recognizers in an effort to improve speech recognition, as discussed next.

2.1 Enhanced Majority Rules

Barry (et al., 1994) took three different Automatic Speech Recognition (ASR) systems, along with an Enhanced Majority Rules (EMR) software algorithm. Each of the three individual systems received the same input, performed speech recognition, and sent the result to the master system.

The EMR resolved inconsistencies by looking for agreement from the individual systems for the recognized word. If there was no majority agreement, the EMR looked to the second word for agreement before relying on the distance scores. This architecture produced better recognition accuracy than each of the individual systems.

While an improvement was made, the architecture can suffer from distorted input. Since each system receives the same input, if the input signal is not good, then all of the individual systems will receive bad input.

2.2 Virtual Intelligent Codriver

The Virtual Intelligent Codriver (VICO) project also used multiple ASR systems in parallel (Brutti et al., 2004; Cristoforetti et al., 2003). Each ASR received the same input and had its own language model. The resulting interpretations from each ASR are compared to each other using confidence scores. The interpretation with the highest recognition accuracy is selected. While the experiments resulted in noticeable improvements over the individual ASR systems, there are two shortcomings. First, if the input signal is distorted, then each recognizer will receive bad input. Secondly, if each recognizer contains a piece of the optimal interpretation, then this architecture falls short.

2.3 Recognized Output Voting Error Reduction

The Recognizer Output Voting Error Reduction (ROVER) system is a composite of multiple ASR systems that uses a voting process to reconcile differences in the individual ASR system outputs (Fiscus, 1997). Multiple interpretations are passed from each recognition engine to the alignment module. Once aligned, the voting module is called. The voting module scores each word within the alignment vertically and the words with the highest scores are chosen. On average, this composite ASR system produces a lower error rate than any of the individual systems, but suffers from order of combination and ties.

2.4 Modified ROVER

To solve the problem that results from the order of combination and ties of the original ROVER system, Schwenk proposed a modified ROVER system that used a dynamic programming algorithm built on language models (Schwenk and Gauvain, 2000). The modified ROVER system resulted in a reduction in the word error rates over the original ROVER system.

3 Distributed Listening

Distributed Listening builds on the architectures that use multiple speech recognizers and enhances it with the use of multiple input sources.

Distributed Listening is made of three significant parts: Listeners, an Interpreter, and a Phrase Resolution Algorithm.

3.1 Listeners

Distributed Listening uses multiple speech recognizers, working in parallel, to process the spoken input. Each recognizer is called a listener and is equipped with it's own input source. Each listener is a separate, physical computing device with its own memory, processor, and disk space. Each listener collects input from the user. The result of each listener is passed to the interpreter.

3.2 Interpreter

Once input is collected from the listeners, the input is passed to the interpreter. The interpreter will

process all of the input collected from each listener as described next.

3.3 Phrase Resolution Algorithm

To resolve multiple interpretations from the listeners, the Phrase Resolution Algorithm (PRA) is used.

The underlying grammar of the PRA is based on an N-gram language model. An N-gram language model is used by the recognizer to predict word sequences. Distributed Listening uses an N-gram of size 1, also known as a unigram. The grammar consists of known utterances that can be made by the user.

The unigram grammar is stored in a phrase database. The grammar is organized according to individual words and phrases. Each phrase is placed in a table. The phrases are broken down into their individual words and placed in another table. The table of words keeps a count of the number of times each word appears in each phrase, resembling the unigram language model.

To determine the most likely spoken phrase, queries are made against the collection of individual words, also known as the complete word set. The queries try to identify matching phrase(s) based on specified words. The matching phrase(s) with the highest concentrations of words is returned by the query.

The word concentration is determined by comparing the length of the phrase with the number of matching words found in the complete word set. The concentration of the number of words found within each phrase is calculated using all interpretations from the listeners. The phrase(s) with the highest concentration of words is the most likely spoken phrase.

4 System Architecture

There are multiple models for Distributed Listening; Homogeneous, Heterogeneous, and Hybrid. The Homogeneous model uses the same grammar for each listener. Within the Heterogeneous model, each listener uses a different grammar. The Hybrid model contains a combination of the Homogenous and Heterogeneous models.

4.1 Homogeneous

In a homogenous Distributed Listening architecture, each listener has the same grammar or language model. Although all of the listeners are identical in capturing the input, this architecture allows for the different perspectives of the utterances to also be captured.

4.2 Heterogeneous

Heterogeneous architectures use different grammars or language models on each listener. Each listener has its own input source and recognizer and implies a distributed grammar/language model. This allows for flexibility as very large grammars and vocabularies can be distributed across several listeners.

4.3 Hybrid

The hybrid architecture is a homogenous architecture of heterogeneous Distributed Listening nodes, as shown in figure 2. This gives the embedded environment the ability to recognize multiple languages, as well as accommodate translations of inter-mixed spoken language.

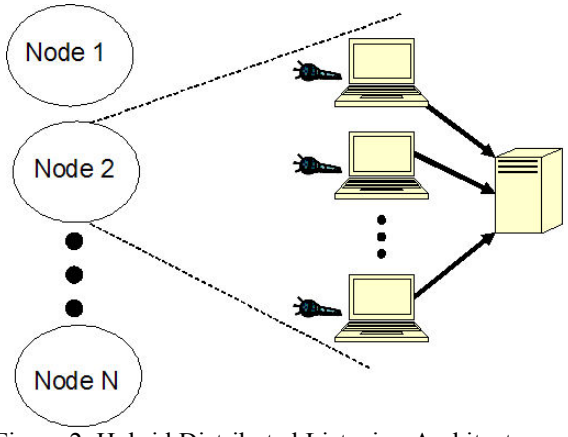


Figure 2. Hybrid Distributed Listening Architecture

5 Conclusion

The goal of Distributed Listening research is to take a unique approach in order to enhance the success of the traditional approaches to speech recognition. The approach of Distributed Listening directly mimics people. The psychology domain has shown that people use a form of Distributed Listening called Dichotic Listening, where people listen to two voices, one in each ear,

at the same time (Bruder, 2004). Distributed Listening is a natural extension of Dichotic Listening, where computers are listening in the same manner as people. Distributed Listening is an attempt to enable computer systems to perform similar to humans while decreasing error rates.

Preliminary studies have shown a decrease in error rates. Early results indicate that Distributed Listening is a viable alternative to current speech recognition systems. Additional studies are being planned that will effectively test the Phrase Resolution Algorithm.

References

Barry, T., Solz, T., Reising, J. & Williamson, D. **The simultaneous use of three machine speech recognition systems to increase recognition accuracy**, In Proceedings of the IEEE 1994 National Aerospace and Electronics Conference, vol.2, pp. 667 - 671, 1994.

Baum, L.E. **An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov process**. Inequalities 3, 1-8, 1972.

Bruder, G.E., Stewart, J.W., McGrath, P.J., Deliyannides, D., Quitkin, F.M. **Dichotic listening tests of functional brain asymmetry predict response to fluoxetine in depressed women and men**. Neuropsychopharmacology, 29(9), pp. 1752-1761, 2004.

Brutti, A., Coletti, P., Cristoforetti, L., Geutner, P., Giacomini, A., Gretter, R., et al. **Use of Multiple Speech Recognition Units in a In-car Assistance Systems**, chapter in "DSP for Vehicle and Mobile Systems", Kluwer Publishers, 2004.

Cristoforetti, L., Matassoni, M., Omologo, M. & Svaizer, P., **Use of parallel recognizers for robust in-car speech interaction**, In Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing [ICASSP 2003], Hong-Kong, 2003.

Deng, L. & Huang, X., **Challenges in adopting speech recognition**, Communications of the ACM, vol. 47, no. 1, pp. 69-75, January 2004.

Fiscus, J. G., **A post-processing system to yield reduced error word rates: Recognizer output voting error reduction (ROVER)**. In IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 347–354, 1997.

Furui, S., **Recent progress in spontaneous speech recognition and understanding**, In Proceedings of the IEEE Workshop on Multimedia Signal Processing, 2002.

Gilbert, J. E. (2005). **Distributed Listening Research.** *In Proceedings of AVIOS Speech Technology Track***,** San Francisco, California, SpeechTEK West, pp. 1 – 10.

Jurafsky, D. & Martin, J., **Speech and Language Processing**, Prentice Hall, 2000.

Natural Language Software Registry, [Online]. Available: http://registry.dfki.de/, 2004.

Schwenk, H. & Gauvain, J., **Improved ROVER using Language Model Information**, In ISCA ITRW Workshop on Automatic Speech Recognition: Challenges for the new Millenium, Paris, pp. 47–52, 2000.

Young, S.R., **Use of dialog, pragmatics and semantics to enhance speech recognition**, Speech Communication, vol. 9, pp. 551-564, 1990.

Young, S.R., Hauptmann, A.G. , Ward, W.H. , Smith, E.T. & Werner, P., **High level knowledge sources in usable speech recognition systems**, Communications of the ACM, vol. 31, no. 2, pp. 183-194, 1989.

Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations

Steven Bethard
Institute for Cognitive Science
Department of Computer Science
University of Colorado
Boulder, CO 80309, USA
steven.bethard@colorado.edu

James H. Martin
Institute for Cognitive Science
Department of Computer Science
University of Colorado
Boulder, CO 80309, USA
james.martin@colorado.edu

Abstract

Finding temporal and causal relations is crucial to understanding the semantic structure of a text. Since existing corpora provide no parallel temporal and causal annotations, we annotated 1000 conjoined event pairs, achieving inter-annotator agreement of 81.2% on temporal relations and 77.8% on causal relations. We trained machine learning models using features derived from WordNet and the Google N-gram corpus, and they outperformed a variety of baselines, achieving an F-measure of 49.0 for temporals and 52.4 for causals. Analysis of these models suggests that additional data will improve performance, and that temporal information is crucial to causal relation identification.

1 Introduction

Working out how events are tied together temporally and causally is a crucial component for successful natural language understanding. Consider the text:

(1) I ate a bad tuna sandwich, got food poisoning and had to have a shot in my shoulder. *wsj_0409*

To understand the semantic structure here, a system must order events along a timeline, recognizing that *getting food poisoning* occurred BEFORE *having a shot*. The system must also identify when an event is not independent of the surrounding events, e.g. *got food poisoning* was CAUSED by *eating a bad sandwich*. Recognizing these temporal and causal relations is crucial for applications like question answering which must face queries like *How did he get food poisoning?* or *What was the treatment?*

Currently, no existing resource has all the necessary pieces for investigating parallel temporal and causal phenomena. The TimeBank (Pustejovsky et al., 2003) links events with BEFORE and AFTER relations, but includes no causal links. PropBank (Kingsbury and Palmer, 2002) identifies ARGM-TMP and ARGM-CAU relations, but arguments may only be temporal or causal, never both. Thus existing corpora are missing some crucial pieces for studying temporal-causal interactions. Our research aims to fill these gaps by building a corpus of parallel temporal and causal relations and exploring machine learning approaches to extracting these relations.

2 Related Work

Much recent work on temporal relations revolved around the TimeBank and TempEval (Verhagen et al., 2007). These works annotated temporal relations between events and times, but low inter-annotator agreement made many TimeBank and TempEval tasks difficult (Boguraev and Ando, 2005; Verhagen et al., 2007). Still, TempEval showed that on a constrained tense identification task, systems could achieve accuracies in the 80s, and Bethard and colleagues (Bethard et al., 2007) showed that temporal relations between a verb and a complement clause could be identified with accuracies of nearly 90%.

Recent work on causal relations has also found that arbitrary relations in text are difficult to annotate and give poor system performance (Reitter, 2003). Girju and colleagues have made progress by selecting constrained pairs of events using web search patterns. Both manually generated Cause-Effect patterns (Girju et al., 2007) and patterns based on nouns

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 177–180,
Columbus, Ohio, USA, June 2008.

	Full	Train	Test
Documents	556	344	212
Event pairs	1000	697	303
BEFORE relations	313	232	81
AFTER relations	16	11	5
CAUSAL relations	271	207	64

Table 1: Contents of the corpus and its train/test sections

Task	Agreement	Kappa	F
Temporals	81.2	0.715	71.9
Causals	77.8	0.556	66.5

Table 2: Inter-annotator agreement by task.

linked causally in WordNet (Girju, 2003) were used to collect examples for annotation, with the resulting corpora allowing machine learning models to achieve performance in the 70s and 80s.

3 Conjoined Events Corpus

Prior work showed that finding temporal and causal relations is more tractable in carefully selected corpora. Thus we chose a simple construction that frequently expressed both temporal and causal relations, and accounted for 10% of all adjacent verbal events: events conjoined by the word *and*.

Our temporal annotation guidelines were based on the guidelines for TimeBank and TempEval, augmented with the guidelines of (Bethard et al., 2008). Annotators used the labels:

BEFORE The first event fully precedes the second
AFTER The second event fully precedes the first
NO-REL Neither event clearly precedes the other

Our causal annotation guidelines were based on paraphrasing rather than the intuitive notions of *cause* used in prior work (Girju, 2003; Girju et al., 2007). Annotators selected the best paraphrase of *"and"* from the following options:

CAUSAL *and as a result, and as a consequence, and enabled by that*
NO-REL *and independently, and for similar reasons*

To build the corpus, we first identified verbs that represented events by running the system of (Bethard and Martin, 2006) on the TreeBank. We then used a set of tree-walking rules to identify conjoined event pairs. 1000 pairs were annotated by two annotators and adjudicated by a third. Table 1

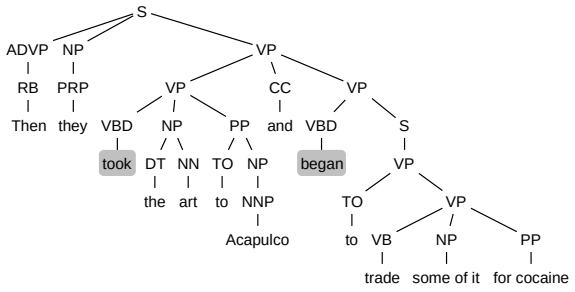


Figure 1: Syntactic tree from *wsj_0450* with events *took* and *began* highlighted.

and Table 2 give statistics for the resulting corpus[1]. The annotators had substantial agreement on temporals (81.2%) and moderate agreement on causals (77.8%). We also report F-measure agreement, since BEFORE, AFTER and CAUSAL relations are more interesting than NO-REL. Annotators had F-measure agreement of 71.9 on temporals and 66.5 causals.

4 Machine Learning Methods

We used our corpus for machine learning experiments where relation identification was viewed as pair-wise classification. Consider the sentence:

(2) The man who had brought it in for an estimate had [EVENT *returned*] to collect it and was [EVENT *waiting*] in the hall. *wsj_0450*

A temporal classifier should label *returned-waiting* with BEFORE since *returned* occurred first, and a causal classifier should label it CAUSAL since this *and* can be paraphrased as *and as a result*.

We identified both syntactic and semantic features for our task. These will be described using the example event pair in Figure 1. Our syntactic features characterized surrounding surface structures:

- The event words, lemmas and part-of-speech tags, e.g. *took*, *take*, `VBD` and *began*, *begin*, `VBD`.
- All words, lemmas and part-of-speech tags in the verb phrases of each event, e.g. *took*, *take*, `VBD` and *began, to, trade*, *begin, trade*, `VBD,TO,VB`.
- The syntactic paths from the first event to the common ancestor to the second event, e.g. `VBD>VP`, `VP` and `VP<VBD`.

[1]Train: *wsj_0416-wsj_0759*. Test: *wsj_0760-wsj_0971*. verbs.colorado.edu/~bethard/treebank-verb-conj-anns.xml

- All words before, between and after the event pair, e.g. *Then, they* plus *the, art, to, Acapulco, and* plus *to, trade, some, of, it, for, cocaine*.

Our semantic features encoded surrounding word meanings. We used WordNet (Fellbaum, 1998) root synsets (*roots*) and lexicographer file names (*lexnames*) to derive the following features:

- All event roots and lexnames, e.g. *take#33, move#1 ... body, change ...* for *took* and *be#0, begin#1 ... change, communication ...* for *began*.
- All lexnames before, between and after the event pair, e.g. *all* plus *artifact, location*, etc. plus *possession, artifact*, etc.
- All roots and lexnames shared by both events, e.g. *took* and *began* were both *act#0, be#0* and *change, communication*, etc.
- The least common ancestor (LCA) senses shared by both events, e.g. *took* and *began* meet only at their roots, so the LCA senses are *act#0* and *be#0*.

We also extracted temporal and causal word associations from the Google N-gram corpus (Brants and Franz, 2006), using `<keyword> <pronoun> <word>` patterns, where *before* and *after* were the keywords for temporals, and *because* was the keyword for causals. Word scores were assigned as:

$$score(w) = log\left(\frac{N_{keyword}(w)}{N(w)}\right)$$

where $N_{keyword}(w)$ is the number of times the word appeared in the keyword's pattern, and $N(w)$ is the number of times the word was in the corpus. The following features were derived from these scores:

- Whether the event score was in at least the Nth percentile, e.g. *took*'s -6.1 *because* score placed it above 84% of the scores, so the feature was true for $N = 70$ and $N = 80$, but false for $N = 90$.
- Whether the first event score was greater than the second by at least N, e.g. *took* and *began* have *after* scores of -6.3 and -6.2 so the feature was true for $N = -1$, but false for $N = 0$ and $N = 1$.

5 Results

We trained SVM^{perf} classifiers (Joachims, 2005) for the temporal and causal relation tasks[2] using the train/test split from Table 1 and the feature sets:

	Temporals			Causals		
Model	P	R	F1	P	R	F1
BEFORE	26.7	94.2	41.6	-	-	-
CAUSAL	-	-	-	21.1	100.0	34.8
1st Event	35.0	24.4	28.8	31.0	20.3	24.5
2nd Event	36.1	30.2	32.9	22.4	17.2	19.5
POS Pair	46.7	8.1	13.9	30.0	4.7	8.1
Syntactic	36.5	53.5	43.4	24.4	79.7	37.4
Semantic	35.8	55.8	43.6	27.2	64.1	38.1
All	43.6	55.8	**49.0**	27.0	59.4	37.1
All+Tmp	-	-	-	46.9	59.4	**52.4**

Table 3: Performance of the temporal relation identification models: (A)ccuracy, (P)recision, (R)ecall and (F1)-measure. The null label is NO-REL.

Syntactic The syntactic features from Section 4.
Semantic The semantic features from Section 4.
All Both syntactic and semantic features.
All+Tmp (Causals Only) Syntactic and semantic features, plus the gold-standard temporal label.

We compared our models against several baselines, using precision, recall and F-measure since the NO-REL labels were uninteresting. Two simple baselines had 0% recall: a lookup table of event word pairs[3], and the majority class (NO-REL) label for causals. We therefore considered the following baselines:

BEFORE Classify all instances as BEFORE, the majority class label for temporals.
CAUSAL Classify all instances as CAUSAL.
1st Event Use a lookup table of 1st words and the labels they were assigned in the training data.
2nd Event As **1st Event**, but using 2nd words.
POS Pair As **1st Event**, but using part of speech tag pairs. POS tags encode tense, so this suggests the performance of a tense-based classifier.

The results on our test data are shown in Table 3. For temporal relations, the F-measures of all SVM models exceeded all baselines, with the combination of syntactic and semantic features performing 5 points better (43.6% precision and 55.8% recall) than either feature set individually. This suggests that our syntactic and semantic features encoded complementary information for the temporal relation task. For

[2]We built multi-class SVMs using the *one-vs-rest* approach and used 5-fold cross-validation on the training data to set parameters. For temporals, C=0.1 (for syntactic-only models), C=1.0 (for all other models), and loss-function=F1 (for all models). For causals, C=0.1 and loss-function=precision/recall break even point (for all models).

[3]Only 3 word pairs from training were seen during testing.

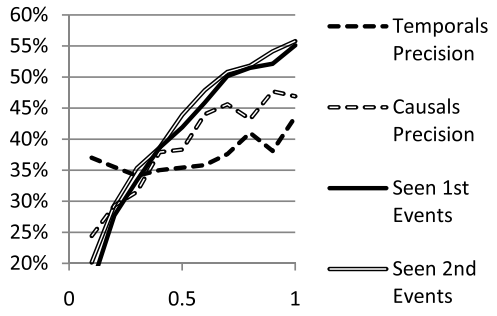


Figure 2: Model precisions (dotted lines) and percent of events in the test data seen during training (solid lines), given increasing fractions of the training data.

causal relations, all SVM models again exceeded all baselines, but combining syntactic features with semantic ones gained little. However, knowing about underlying temporal relations boosted performance to 46.9% precision and 59.4% recall. This shows that progress in causal relation identification will require knowledge of temporal relations.

We examined the effect of corpus size on our models by training them on increasing fractions of the training data and evaluating them on the test data. The precisions of the resulting models are shown as dotted lines in Figure 2. The models improve steadily, and the causals precision can be seen to follow the solid curves which show how event coverage increases with increased training data. A logarithmic trendline fit to these seen-event curves suggests that annotating all 5,013 event pairs in the Penn TreeBank could move event coverage up from the mid 50s to the mid 80s. Thus annotating additional data should provide a substantial benefit to our temporal and causal relation identification systems.

6 Conclusions

Our research fills a gap in existing corpora and NLP systems, examining parallel temporal and causal relations. We annotated 1000 event pairs conjoined by the word *and*, assigning each pair both a temporal and causal relation. Annotators achieved 81.2% agreement on temporal relations and 77.8% agreement on causal relations. Using features based on WordNet and the Google N-gram corpus, we trained support vector machine models that achieved 49.0 F on temporal relations, and 37.1 F on causal relations. Providing temporal information to the causal relations classifier boosted its results to 52.4 F. Future work will investigate increasing the size of the corpus and developing more statistical approaches like the Google N-gram scores to take advantage of large-scale resources to characterize word meaning.

Acknowledgments

This research was performed in part under an appointment to the U.S. Department of Homeland Security (DHS) Scholarship and Fellowship Program.

References

S. Bethard and J. H. Martin. 2006. Identification of event mentions and their semantic class. In *EMNLP-2006*.

S. Bethard, J. H. Martin, and S. Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *ICSC-2007*.

S. Bethard, W. Corvey, S. Klingenstein, and J. H. Martin. 2008. Building a corpus of temporal-causal structure. In *LREC-2008*.

B. Boguraev and R. K. Ando. 2005. Timebank-driven timeml analysis. In *Annotating, Extracting and Reasoning about Time and Events*. IBFI, Schloss Dagstuhl, Germany.

T. Brants and A. Franz. 2006. Web 1t 5-gram version 1. Linguistic Data Consortium, Philadelphia.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press.

R. Girju, P. Nakov, V. Nastase, S. Szpakowicz, P. Turney, and D. Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *SemEval-2007*.

R. Girju. 2003. Automatic detection of causal relations for question answering. In *ACL Workshop on Multilingual Summarization and Question Answering*.

T. Joachims. 2005. A support vector method for multivariate performance measures. In *ICML-2005*.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *LREC-2002*.

J. Pustejovsky, P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. 2003. The timebank corpus. In *Corpus Linguistics*, pages 647–656.

D. Reitter. 2003. Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. *LDV-Forum, GLDV-Journal for Computational Linguistics and Language Technology*, 18(1/2):38–52.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *SemEval-2007*.

Evolving new lexical association measures using genetic programming

Jan Šnajder **Bojana Dalbelo Bašić** **Saša Petrović** **Ivan Sikirić**
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, Croatia
{jan.snajder, bojana.dalbelo, sasa.petrovic, ivan.sikiric}@fer.hr

Abstract

Automatic extraction of collocations from large corpora has been the focus of many research efforts. Most approaches concentrate on improving and combining known lexical association measures. In this paper, we describe a genetic programming approach for evolving new association measures, which is not limited to any specific language, corpus, or type of collocation. Our preliminary experimental results show that the evolved measures outperform three known association measures.

1 Introduction

A *collocation* is an expression consisting of two or more words that correspond to some conventional way of saying things (Manning and Schütze, 1999). Related to the term collocation is the term n-*gram*, which is used to denote any sequence of *n* words. There are many possible applications of collocations: automatic language generation, word sense disambiguation, improving text categorization, information retrieval, etc. As different applications require different types of collocations that are often not found in dictionaries, automatic extraction of collocations from large textual corpora has been the focus of much research in the last decade; see, for example, (Pecina and Schlesinger, 2006; Evert and Krenn, 2005).

Automatic extraction of collocations is usually performed by employing *lexical association measures* (AMs) to indicate how strongly the words comprising an *n*-gram are associated. However, the use of lexical AMs for the purpose of collocation extraction has reached a plateau; recent research in this field has focused on combining the existing AMs in the hope of improving the results (Pecina and Schlesinger, 2006). In this paper, we propose an approach for deriving new AMs for collocation extraction based on genetic programming. A similar approach has been usefully applied in text mining (Atkinson-Abutridy et al., 2004) as well as in information retrieval (Gordon et al., 2006).

Genetic programming is an evolutionary computational technique designed to mimic the process of natural evolution for the purpose of solving complex optimization problems by stochastically searching through the whole space of possible solutions (Koza, 1992). The search begins from an arbitrary seed of possible solutions (the initial population), which are then improved (evolved) through many iterations by employing the operations of selection, crossover, and mutation. The process is repeated until a termination criterion is met, which is generally defined by the goodness of the best solution or the expiration of a time limit.

2 Genetic programming of AMs

2.1 AM representation

In genetic programming, possible solutions (in our case lexical AMs) are mathematical expressions represented by a tree structure (Koza, 1992). The leaves of the tree can be constants, or statistical or linguistic information about an *n*-gram. A constant can be any real number in an arbitrarily chosen interval; our experiments have shown that variation of this interval does not affect the performance. One special constant that we use is the number of words in the corpus. The statistical information about an *n*-gram can be the frequency of any part of the *n*-gram. For ex-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 181–184,
Columbus, Ohio, USA, June 2008.

ample, for a trigram *abc* the statistical information can be the frequency $f(abc)$ of the whole trigram, frequencies $f(ab)$ and $f(bc)$ of the digrams, and the frequencies of individual words $f(a)$, $f(b)$, and $f(c)$. The linguistic information about an *n*-gram is the part-of-speech (POS) of any one of its words.

Inner nodes in the tree are operators. The binary operators are addition, subtraction, multiplication, and division. We also use one unary operator, the natural logarithm, and one ternary operator, the IF-THEN-ELSE operator. The IF-THEN-ELSE node has three descendant nodes: the left descendant is the condition in the form "i-th word of the *n*-gram has a POS tag T," and the other two descendants are operators or constants. If the condition is true, then the subexpression corresponding to the middle descendant is evaluated, otherwise the subexpression corresponding to the right descendant is evaluated.

The postfix expression of an AM can be obtained by traversing its tree representation in postorder. Figure 1 shows the representation of the Dice coefficient using our representation.

2.2 Genetic operators

The crossover operator combines two parent solutions into a new solution. We defined the crossover operator as follows: from each of the two parents, one node is chosen randomly, excluding any nodes that represent the condition of the IF-THEN-ELSE operator. A new solution is obtained by replacing the subtree of the chosen node of the first parent with the subtree of the chosen node of the second parent. This method of defining the crossover operator is the same as the one described in (Gordon et al., 2006).

The mutation operator introduces new "genetic material" into a population by randomly changing a solution. In our case, the mutation operator can do one of two things: either remove a randomly selected inner node (with probability of 25%), or insert an inner node at a random position in the tree (with probability of 75%). If a node is being removed from the tree, one of its descendants (randomly chosen) takes its place. An exception to this rule is the IF-THEN-ELSE operator, which cannot be replaced by its condition node. If a node is being inserted, a randomly created operator node replaces an existing node that then becomes a descendant of the new node. If the inserted node is not a unary operator, the required number of random leaves is created.

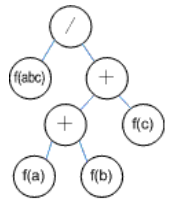
The selection operator is used to copy the best solutions into the next iteration. The goodness of the solution is determined by the *fitness function*, which assigns to each solution a number indicating how good that particular solution actually is. We measure the goodness of an AM in terms of its F_1 score, obtained from the precision and recall computed on a random sample consisting of 100 positive *n*-grams (those considered collocations) and 100 negative *n*-grams (non-collocations). These *n*-grams are ranked according to the AM value assigned to them, after which we compute the precision and recall by considering first n best-ranked *n*-grams as positives and the rest as negatives, repeating this for each n between 1 and 200. The best F_1 score is then taken as the AM's goodness.

Using the previous definition of the fitness function, preliminary experiments showed that solutions soon become very complex in terms of number of nodes in the tree (namely, on the order of tens of thousands). This is a problem both in terms of space and time efficiency; allowing unlimited growth of the tree quickly consumes all computational resources. Also, it is questionable whether the performance benefits from the increased size of the solution. Thus, we modified the fitness function to also take into account the size of the tree (that is, the less nodes a tree has, the better). Favoring shorter solutions at the expense of some loss in performance is known as *parsimony*, and it has already been successfully used in genetic programming (Koza, 1992). Therefore, the final formula for the fitness function we used incorporates the parsimony factor and is given by

$$fitness(j) = F_1(j) + \eta \frac{L_{max} - L(j)}{L_{max}}, \quad (1)$$

where $F_1(j)$ is the F_1 score (ranging from 0 to 1) of the solution j, η is the parsimony factor, L_{max} is the maximal size (measured in number of nodes), and $L(j)$ is the size of solution j. By varying η we can control how much loss of performance we will tolerate in order to get smaller, more elegant solutions.

Genetic programming algorithms usually iterate until a *termination criterion* is met. In our case, the algorithm terminates when a certain number, k, of iterations has passed without an improvement in the



$$Dice(a,b,c) = \frac{f(abc)}{f(a)+f(b)+f(c)}$$

Figure 1: Dice coefficient for digrams represented by tree

results. To prevent the overfitting problem, we measure this improvement on another sample (*validation sample*) that also consists of 100 collocations and 100 non-collocations.

3 Preliminary results

3.1 Experimental setting

We use the previously described genetic programming approach to evolve AMs for extracting collocations consisting of three words from a corpus of 7008 Croatian legislative documents. Prior to this, words from the corpus were lemmatized and POS tagged. Conjunctions, propositions, pronouns, interjections, and particles were treated as stop-words and tagged with a POS tag X. *N*-grams starting or ending with a stopword, or containing a verb, were filtered out. For evaluation purposes we had a human expert annotate 200 collocations and 200 non-collocations, divided into the evaluation and validation sample. We considered an *n*-gram to be a collocation if it is a compound noun, terminological expression, or a proper name. Note that we could have adopted any other definition of a collocation, since this definition is implicit in the samples provided.

In our experiments, we varied a number of genetic programming parameters. The size of the initial population varied between 50 and 50 thousand randomly generated solutions. To examine the effects of including some known AMs on the performance, the following AMs had a 50% chance of being included in the initial population: pointwise mutual information (Church and Hanks, 1990), the Dice coefficient, and the heuristic measure defined in (Petrović et al., 2006):

$$H(a,b,c) = \begin{cases} 2\log\frac{f(abc)}{f(a)f(c)} & \text{if } POS(b) = X, \\ \log\frac{f(abc)}{f(a)f(b)f(c)} & \text{otherwise.} \end{cases}$$

For the selection operator we used the well-known *three-tournament selection*. The probability of mutation was chosen from the interval $[0.0001, 0.3]$, and the parsimony factor η from the interval $[0, 0.05]$, thereby allowing a maximum of 5% loss of F_1 in favor of smaller solutions. The maximal size of the tree in nodes was chosen from the interval $[20, 1000]$. After the F_1 score for the validation sample began dropping, the algorithm would continue for another k iterations before stopping. The parameter k was chosen from the interval $[10^4, 10^7]$. The experiments were run with 800 different random combinations of the aforementioned parameters.

3.2 Results

Around 20% of the evolved measures (that is, the solutions that remained after the algorithm terminated) achieved F_1 scores of over 80% on both the evaluation and validation samples. This proportion was 13% in the case when the initial population did not include any known AMs, and 23% in the case when it did, thus indicating that including known AMs in the initial population is beneficial. The overall best solution had 205 nodes and achieved an F_1 score of 88.4%. In search of more elegant AMs, we singled out solutions that had less than 30 nodes. Among these, a solution that consisted of 13 nodes achieved the highest F_1. This measure is given by

$$M_{13}(a,b,c) = \begin{cases} -0.423\frac{f(a)f(c)}{f^2(abc)} & \text{if } POS(b) = X, \\ 1 - \frac{f(b)}{f(abc)} & \text{otherwise.} \end{cases}$$

The association measure M_{13} is particularly interesting because it takes into account whether the middle word in a trigram is a stopword (denoted by the POS tag X). This supports the claim laid out in (Petrović et al., 2006) that the trigrams containing stopwords (e.g., *cure for cancer*) should be treated differently, in that the frequency of the stopword should be ignored. It is important to note that the aforementioned measure H was not included in the initial population from which M_{13} evolved. It is also worthwhile noting that in such populations, out of 100 best evolved measures, all but four of them featured a condition identical to that of M_{13} ($POS(b) = X$). In other words, the majority of the measures evolved this condition completely independently, without H being included in the initial population.

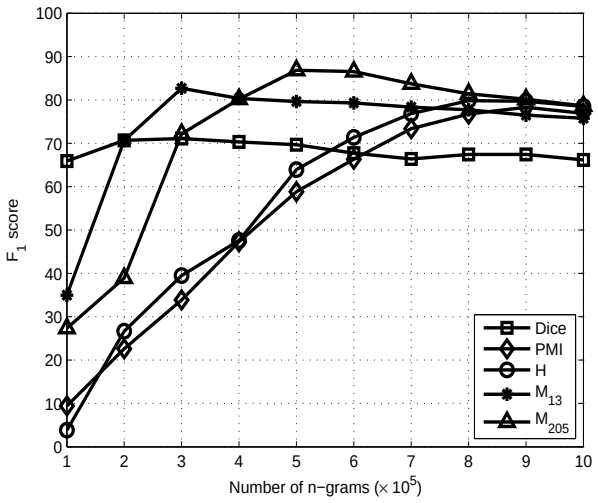


Figure 2: Comparison of association measures on a corpus of 7008 Croatian documents

Figure 2 shows the comparison of AMs in terms of their F_1 score obtained on the corpus of 7008 documents. The x axis shows the number of n best ranked *n*-grams that are considered positives (we show only the range of n in which all the AMs achieve their maximum F_1; all measures tend to perform similarly with increasing n). The maximum F_1 score is achieved if we take 5×10^5 *n*-grams ranked best by the M_{205} measure. From Fig. 2 we can see that the evolved AMs M_{13} and M_{205} outperformed the other three considered AMs. For example, collocations *kosilica za travu* (*lawn mower*) and *digitalna obrada podataka* (*digital data processing*) were ranked at the 22th and 34th percentile according to Dice, whereas they were ranked at the 97th and 87th percentile according to M_{13}.

4 Conclusion

In this paper we described a genetic programming approach for evolving new lexical association measures in order to extract collocations.

The evolved association measure will perform at least as good as any other AM included in the initial population. However, the evolved association measure may be a complex expression that defies interpretation, in which case it may be treated as a black-box suitable for the specific task of collocation extraction. Our approach only requires an evaluation sample, thus it is not limited to any specific type of collocation, language or corpus.

The preliminary experiments, conducted on a corpus of Croatian documents, showed that the best evolved measures outperformed other considered association measures. Also, most of the best evolved association measures took into account the linguistic information about an *n*-gram (the POS of the individual words).

As part of future work, we intend to apply our approach to corpora in other languages and compare the results with existing collocation extraction systems. We also intend to apply our approach to collocations consisting of more than three words, and to experiment with additional linguistic features.

Acknowledgments

This work has been supported by the Government of Republic of Croatia, and Government of Flanders under the grant No. 036-1300646-1986 and KRO/009/06.

References

John Atkinson-Abutridy, Chris Mellish, and Stuart Aitken. 2004. Combining information extraction with genetic algorithms for text mining. *IEEE Intelligent Systems*, 19(3):22–30.

Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Stephan Evert and Brigitte Krenn. 2005. Using small random samples for the manual evaluation of statistical evaluation measures. *Computer Speech and Language*, 19(4):450–466.

Michael Gordon, Weiguo Fan, and Praveen Pathak. 2006. Adaptive web search: Evolving a program that finds information. *IEEE Intelligent Systems*, 21(5):72–77.

John R. Koza. 1992. *Genetic programming: On the programming of computers by means of natural selection*. MIT Press.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Pavel Pecina and Pavel Schlesinger. 2006. Combining association measures for collocation extraction. In *Proc. of the COLING/ACL 2006*, pages 651–658.

Saša Petrović, Jan Šnajder, Bojana Dalbelo Bašić, and Mladen Kolar. 2006. Comparison of collocation extraction measures for document indexing. *J. of Computing and Information Technology*, 14(4):321–327.

Semantic Types of Some Generic Relation Arguments: Detection and Evaluation

Sophia Katrenko
Institute of Informatics
University of Amsterdam
the Netherlands
katrenko@science.uva.nl

Pieter Adriaans
Institute of Informatics
University of Amsterdam
the Netherlands
pietera@science.uva.nl

Abstract

This paper presents an approach to detection of the semantic types of relation arguments employing the WordNet hierarchy. Using the SemEval-2007 data, we show that the method allows to generalize relation arguments with high precision for such generic relations as *Origin-Entity*, *Content-Container*, *Instrument-Agency* and some other.

1 Introduction and Motivation

A common approach to learning relations is composed from two steps, identification of arguments and relation validation. This methodology is widely used in different domains, such as biomedical. For instance, in order to extract instances of a relation of protein interactions, one has to first identify all protein names in text and, second, verify if a relation between them holds.

Clearly, if arguments are already given, accuracy of relation validation is higher compared to the situation when the arguments have to be identified automatically. In either case, this methodology is effective for the domain-dependent relations but is not considered for more generic relation types. If a relation is more generic, such as *Part-Whole*, it is more difficult to identify its arguments because they can be of many different semantic types. An example below contains a causality relation (**virus** causes **flu**). Note that syntactic information is not sufficient to be able to detect such relation mention and the background knowledge is needed.

> A person infected with a particular **flu virus** strain develops antibody against that virus.

In this paper we propose a method to detect semantic types of the generic relation arguments. For the *Part-Whole* relation, it is known that it embraces such subtypes as *Member-Collection* or *Place-Area* while there is not much information on the other relation types. We do not claim semantic typing to be sufficient to recognize relation mentions in text, however, it would be interesting to examine the accuracy of relation extraction when the background knowledge *only* is used. Our aim is therefore to discover precise generalizations per relation type rather than to cover all possible relation mentions.

2 A Method: Making Semantic Types of Arguments Explicit

We propose a method for generalizing relation argument types based on the positive and negative examples of a given relation type. It is also necessary that the arguments of a relation are annotated using some semantic taxonomy, such as WordNet (Fellbaum, 1998). Our hypothesis is as follows: because of the positive and negative examples, it should be possible to restrict semantic types of arguments using negative examples. If negative examples are nearly positive, the results of such generalization should be precise. Or, in machine learning terms, such negative examples are close to the decision boundary and if used during generalization, precision will be boosted. If negative examples are far from the decision boundary, their use will most likely not help to identify semantic types and will result in overgeneralization.

To test this hypothesis, we use an idea borrowed from induction of the deterministic finite automata.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 185–188,
Columbus, Ohio, USA, June 2008.

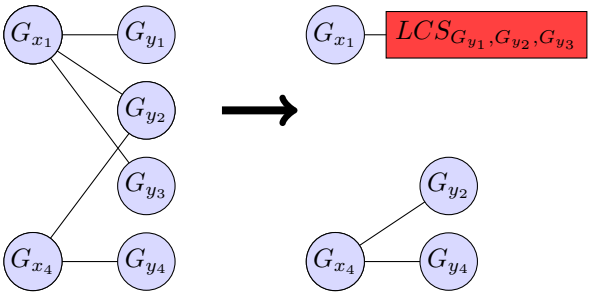


Figure 1: Generalization process.

More precisely, to infer deterministic finite automata (DFA) from positive and negative examples, one first builds the maximal canonical automaton (MCA) (Pernot et al., 2005) with one starting state and a separate sequence of states for each positive example and then uses a merging strategy such that no negative examples are accepted.

Similarly, for a positive example $< x_i, y_i >$ we collect all f hyperonyms $H_{x_i} = h^1_{x_i}, h^2_{x_i}, \ldots, h^f_{x_i}$ for x_i where $h^1_{x_i}$ is an immediate hyperonym and $h^f_{x_i}$ is the most general hyperonym. The same is done for y_i. Next, we use all negative examples to find G_{x_i} and G_{y_i} which are generalization types of the arguments of a given positive example $< x_i, y_i >$. In other words, we perform generalization per relation argument in a form of *one positive example vs. all negative examples*. Because of the multi-inheritance present in WordNet, it is possible to find more hyperonymy paths than one. To take it into account, the most general hyperonym $h^f_{x_i}$ equals to a splitting point/node.

It is reasonable to assume that the presence of a general semantic category of one argument will require a more specific semantic category for the other. Generalization per argument is, on the one hand, useful because none of the arguments share a semantic category with the corresponding arguments of all negative examples. On the other hand, it is too restrictive if one aims at identification of the relation type. To avoid this, we propose to generalize semantic category of one argument by taking into account a semantic category of the other. In particular, one can represent a binary relation as a bipartite graph where the corresponding nodes (relation arguments) are connected. A natural way of generalizing would be to combine the nodes which differ on the basis of their similarity. In case of WordNet, we can use a least common subsumer (LCS) of the nodes. Given the bipartite graph in Figure 1, it can be done as follows. For every vertex G_{x_i} in one part which is connected to several vertices $G_{y_1}, \ldots, G_{y_k}$ in the other, we compute LCS of $G_{y_1}, \ldots, G_{y_k}$. Note that we require the semantic contrains on both arguments to be satisfied in order to validate a given relation. Generalization via LCS is carried out in both directions. This step is described in more detail in Algorithm 1.

Algorithm 1 Generalization via LCS

1: Memory $\mathcal{M} = \varnothing$
2: Direction: $\rightarrow$
3: **for all** $< G_{x_i}, G_{y_i} > \in \mathcal{G}$ **do**
4: Collect all $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ s. t. $G_{x_i} = G_{x_j}$
5: **if** exists $< G_{x_k}, G_{y_j} >$ s. t. $G_{x_i} \neq G_{x_k}$ **then**
6: $\mathcal{G} = \mathcal{G} \cup \{< G_{x_j}, G_{y_j} >\}$
7: **end if**
8: Compute $\mathcal{L} = LCS_{G_{y_0},\ldots,G_{y_l}}$
9: Replace $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ with $< G_{x_j}, \mathcal{L} >$ in $\mathcal{G}$
10: $\mathcal{M} = \mathcal{M} \cup \{< G_{x_j}, \mathcal{L} >\}$
11: **end for**
12: Direction: $\leftarrow$
13: **for all** $< G_{x_i}, G_{y_i} > \in \mathcal{G}$ **do**
14: Collect all $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ s. t. $G_{y_i} = G_{y_j}$ and $< G_{x_j}, G_{y_j} > \notin \mathcal{M}$
15: Compute $\mathcal{L} = LCS_{G_{x_0},\ldots,G_{x_l}}$
16: Replace $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ with $< \mathcal{L}, G_{y_j} >$ in $\mathcal{G}$
17: **end for**
18: **return** $\mathcal{G}$

Example Consider, for instance, two sentences from the SemEval data (Instrument-Agency relation).

> 013 ”The test is made by inserting the end of a <e1>jimmy</e1> or other <e2>burglar</e2>’s tool and endeavouring to produce impressions similar to those which have been found on doors or windows.” WordNet(e1) = ”jimmy%1:06:00::”, WordNet(e2) = ”burglar%1:18:00::”, Instrument-Agency(e1, e2) = ”true”
> 040 ”<e1>Thieves</e1> used a <e2>blowtorch</e2> and bolt cutters to force their way through a fenced area

topped with razor wire." WordNet(e1) = "thief%1:18:00::", WordNet(e2) = "blowtorch%1:06:00::", Instrument-Agency(e2, e1) = "true"

First, we find the sense keys corresponding to the relation arguments, ("jimmy%1:06:00::", "burglar%1:18:00::") = (jimmy#1, burglar#1) and ("blowtorch%1:06:00::", "thief%1:18:00::") = (blowtorch#1, thief#1).By using negative examples, we obtain the following pairs: (apparatus#1, bad_person#1) and (bar#3, bad_person#1). These pairs share the second argument and it makes it possible to apply generalization in the direction $\leftarrow$. LCS of apparatus#1 and bar#3 is instrumentality#3 and hence the generalized pair becomes (instrumentality#3, bad_person#1).

Note that an order in which the directions are chosen in Algorithm 1 does not affect the resulting generalizations. Keeping all generalized pairs in the memory $\mathcal{M}$ ensures that whatever direction ($\rightarrow$ or $\leftarrow$) a user chooses first, the output of the algorithm will be the same.

Until now, we have considered generalization in one step only. It would be natural to extend this approach to the iterative generalization such that it is performed until no further generalization steps can be made (it corresponds either to the two specific argument types or to the situation when the top of the hierarchy is reached). However, such method would most likely result in overgeneralization by boosting recall but drastically decreasing precision. As an alternative we propose to use memory $\mathcal{M}^{\mathcal{I}}$ defined over the iterations. After each iteration step every generalized pair $< G_{x_i}, G_{y_i} >$ is applied to the training set and if it accepts at least one negative example, it is either removed from the set $\mathcal{G}$ (first iteration) or this generalization pair is decomposed back into the pairs it was formed from (all other iterations). By employing backtracking we guarantee that empirical error on the training set $E_{emp} = 0$.

3 Evaluation

Data For semantic type detection, we use 7 binary relations from the training set of the SemEval-2007 competition, all definitions of which share the requirement of the syntactic closeness of the arguments. Further, their definitions have various restrictions on the nature of the arguments. Short description of the relation types we study is given below.

Cause-Effect(X,Y) This relation takes place if, given a sentence S, it is possible to entail that X is the cause of Y. Y is usually not an entity but a nominal denoting occurrence (activity or event).

Instrument-Agency(X,Y) This relation is true if S entails the fact that X is the instrument of Y (Y uses X). Further, X is an entity and Y is an actor or an activity.

Product-Producer(X,Y) X is a product of Y, or Y produces X, where X is any abstract or concrete object.

Origin-Entity(X,Y) X is the origin of Y where X can be spatial or material and Y is the entity derived from the origin.

Theme-Tool(X,Y) The tool Y is intended for X is either its result or something that is acted upon.

Part-Whole(X,Y) X is part of Y and this relation can be one of the following five types: Place-Area, Stuff-Object, Portion-Mass, Member-Collection and Component-Integral object.

Content-Container(X,Y) A sentence S entails the fact that X is stored inside Y. Moreover, X is not a component of Y and can be removed from it.

We hypothesize that *Cause-Effect* and *Part-Whole* are the relation types which may require sentential information to be detected. These two relations allow a greater variety of arguments and the semantic information alone might be not sufficient. Such relation types as *Product-Producer* or *Instrument-Agency* are likely to benefit more from the external knowledge. Our method depends on the positive and negative examples in the training set and on the semantic hierarchy we use. If some parts of the hierarchy are more flat, the resulting patterns may be too general.

As not all examples have been annotated with the information from WordNet, we removed them form the test data while conducting this experiment. *Content-Container* turned out to be the only relation type whose examples are fully annotated. In contrast, *Product-Producer* is a relation type with the most information missing (9 examples removed). There is no reason to treat relation mentions as mutually exclusive, therefore, only negative example provided for a particular relation type are used to determine semantic types of its arguments.

Discussion The entire generalization process results in a zero-error on the training set. It does not, however, guarantee to hold given a new data set. The loss in precision on the unseen exam-

Relation type	P, %	R, %	A, %	B-A, %
Origin-Entity	100	26.5	67.5	55.6
Content-Container	81.8	47.4	67.6	51.4
Cause-Effect	100	2.8	52.7	51.2
Instrument-Agency	78.3	48.7	67.6	51.3
Product-Producer	77.8	38.2	52.4	66.7
Theme-Tool	66.7	8.3	65.2	59.2
Part-Whole	66.7	15.4	66.2	63.9
avg.	**81.6**	26.8	62.7	57.0

Table 1: Performance on the test data

ples can be caused by the generalization pairs where both arguments are generalized to the higher level in the hierarchy than it ought to be. To check how the algorithm behaves, we first evaluate the specialization step on the test data from the SemEval challenge. Among all the relation types, only *Instrument-Agency*, *Part-Whole* and *Content-Container* fail to obtain 100% precision after the specialization step. It means that, already at this stage, there are some false positives and the contextual classification is required to achieve better performance.

The results of the method introduced here are presented in Table 1. Systems which participated in SemEval were categorized depending on the input information they have used. The category *WordNet* implies that WordNet was employed but it does not exclude a possibility of using other resources. Therefore, to estimate how well our method performs, we calculated accuracy and compared it against a baseline that always returns the most frequent class label (B-A). Given the results of the teams participating in the challenge, the organizers mention *Product-Producer* as one of the easiest relations, while *Origin-Entity* and *Theme-Tool* are considered to be ones of the hardest to detect (Girju et al., 2007). Interestingly, *Origin-Entity* obtains the highest precision compared to the other relation types while using our approach.

Table 2 contains some examples of the semantic types we found for each relation. Some of them are quite specific (e.g., *Origin-Entity*), while the other arguments may be very general (e.g., *Cause-Effect*). The examples of the patterns for *Part-Whole* can be divided in several subtypes, such as Member-Collection (person#1, social_group#1), Place-Area (top_side#1, whole#2) or Stuff-Object (germanium#1, mineral#1).

Relation	(G_X, G_Y)
Content-Container	(physical_entity#1, vessel#3)
Instrument-Agency	(instrumentality#3, bad_person#1) (printing_machine#1, employee#1)
Cause-Effect	(cognitive_operation#1, joy#1) (entity#1, harm#2) (cognitive_content#1, communication#2)
Product-Producer	(knowledge#1, social_unit#1) (content#2, individual#1) (instrumentality#3, business_organisation#1)
Origin-Entity	(article#1, section#1) (vegetation#1, plant_part#1) (physical_entity#1, fat#1)
Theme-Tool	(abstract_entity#1, implementation#2) (animal#1, water#6) (nonaccomplishment#1, human_action#1)
Part-Whole	(top_side#1, whole#2) (germanium#1, mineral#1) (person#1, social_group#1)

Table 2: Some examples per relation type.

4 Conclusions

As expected, the semantic types derived for such relations as *Origin-Entity*, *Content-Container* and *Instrument-Agency* provide high precision on the test data. In contrast, precision for *Theme-Tool* is the lowest which has been noted by the participants of the SemEval-2007. In terms of accuracy, *Cause-Effect* seems to obtain 100% precision but low recall and accuracy. An explanation for that might be a fact that causation can be characterized by a great variety of argument types many of which have been absent in the training data. *Origin-Entity* obtains the maximal precision with accuracy much higher than baseline.

References

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

Nicholas Pernot, Antoine Cornuéjols, and Michele Sebag. 2005. Phase transition within grammatical inference. *In Proceedings of IJCAI 2005*.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. *In ACL 2007*.

Mapping between Compositional Semantic Representations and Lexical Semantic Resources: Towards Accurate Deep Semantic Parsing

Sergio Roa†‡, Valia Kordoni† and Yi Zhang†
Dept. of Computational Linguistics, Saarland University, Germany†
German Research Center for Artificial Intelligence (DFKI GmbH)†
Dept. of Computer Science, University of Freiburg, Germany‡
{sergior,kordoni,yzhang}@coli.uni-sb.de

Abstract

This paper introduces a machine learning method based on bayesian networks which is applied to the mapping between deep semantic representations and lexical semantic resources. A probabilistic model comprising Minimal Recursion Semantics (MRS) structures and lexicalist oriented semantic features is acquired. Lexical semantic roles enriching the MRS structures are inferred, which are useful to improve the accuracy of deep semantic parsing. Verb classes inference was also investigated, which, together with lexical semantic information provided by VerbNet and PropBank resources, can be substantially beneficial to the parse disambiguation task.

1 Introduction

Recent studies of natural language parsing have shown a clear and steady shift of focus from pure syntactic analyses to more semantically informed structures. As a result, we have seen an emerging interest in parser evaluation based on more theory-neutral and semantically informed representations, such as dependency structures. Some approaches have even tried to acquire semantic representations without full syntactic analyses. The so-called shallow semantic parsers build basic predicate-argument structures or label semantic roles that reveal the partial meaning of sentences (Carreras and Màrquez, 2005). Manually annotated lexical semantic resources like PropBank (Palmer et al., 2005), VerbNet (Kipper-Schuler, 2005), or FrameNet (Baker et al., 1998) are usually used as gold standards for training and evaluation of such systems. In the meantime, various existing parsing systems are also adapted to provide semantic information in their outputs. The obvious advantage in such an approach is that one can derive more fine-grained representations which are not typically available from shallow semantic parsers (e.g., modality and negation, quantifiers and scopes, etc.). To this effect, various semantic representations have been proposed and used in different parsing systems. Generally speaking, such semantic representations should be capable of embedding shallow semantic information (i.e., predicate-argument or semantic roles). However, it is non-trivial to map even the basic predicate-arguments between different representations. This becomes a barrier to both sides, making the cross-fertilization of systems and resources using different semantic representations very difficult.

In this paper, we present a machine learning approach towards mapping between deep and shallow semantic representations. More specifically, we use Bayesian networks to acquire a statistical model that enriches the Minimal Recursion Semantics structures produced by the English Resource Grammar (ERG) (Flickinger, 2002) with VerbNet-like semantic roles. Evaluation results show that the mapping from MRS to semantic roles is reliable and beneficial to deep parsing.

2 Minimal Recursion Semantics

The semantic representation we are interested in in this paper is the Minimal Recursion Semantics (MRS). Because of its underspecifiability, it has been widely used in many deep and shallow processing systems. The main assumption behind MRS is that the interesting linguistic units for computational semantics are the *elementary predications* (EPs), which are single relations with associated arguments (Copestake et al., 2006). In this paper, the MRS structures are created with the English Resource Grammar (ERG), a HPSG-based broad coverage precision grammar for English. The seman-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 189–192,
Columbus, Ohio, USA, June 2008.

tic predicates and their linguistic behaviour (including the set of semantic roles, indication of optional arguments, and their possible value constraints are specified by the grammar as its semantic interface (SEM-I) (Flickinger et al., 2005).

3 Relating MRS structures to lexical semantic resources

3.1 Feature extraction from linguistic resources

The first set of features used to find corresponding lexical semantic roles for the MRS predicate arguments are taken from Robust MRS (RMRS) structures (Copestake, 2006). The general idea of the process is to traverse the bag of elementary predications looking for the verbs in the parsed sentence. When a verb is found, then its arguments are taken from the **rarg** tags and alternatively from the **in-g** conjunctions related to the verb. So, given the sentence:

(1) Yields on money-market mutual funds continued to slide, amid signs that portfolio managers expect further declines in interest rates.

the obtained features for *expect* are shown in Table 1.

SEM-I roles	Features	Words
ARG1	`manager_n_of`	managers
ARG2	`propositional_m_rel`	further declines

Table 1: RMRS features for the verb *expect*

The SEM-I role labels are based mainly on syntactic characteristics of the verb. We employed the data provided by the PropBank and VerbNet projects to extract lexical semantic information. For PropBank, the argument labels are named ARG1,..., ARGN and additionally ARGM for adjuncts. In the case of VerbNet, 31 different thematic roles are provided, e.g. *Actor, Agent, Patient, Proposition, Predicate, Theme, Topic*. A treebank of RMRS structures and derivations was generated by using the PropBank corpus. The process of RMRS feature extraction was applied and a new verb dependency trees dataset was created.

To obtain a correspondence between the SEM-I role labels and the PropBank (or VerbNet) role labels, a procedure which maps these labellings for each utterance and verb found in the corpus was implemented. Due to the possible semantic roles that subjects and objects in a sentence could bear, the mapping between SEM-I roles and VerbNet role labels is not one-to-one. The general idea of this alignment process is to use the words in a given utterance which are selected by a given role label, both a SEM-I and a PropBank one. With these words, a naive assumption was applied that allows a reasonable comparison and alignment of these two sources of information. The naive assumption considers that if all the words selected by some SEM-I label are found in a given PropBank (VerbNet) role label, then we can deduce that these labels can be aligned. An important constraint is that all the SEM-I labels must be exhausted. An additional constraint is that ARG1, ARG2 or ARG3 SEM-I labels cannot be mapped to ARGM PropBank labels. When an alignment between a SEM-I role and a corresponding lexical semantic role is found, no more mappings for these labels are allowed. For instance, given the example in Table 1, with the following Propbank (VerbNet) labelling:

(2) [$_{\text{Arg}_0\text{(Experiencer)}}$ Portfolio managers] *expect* [$_{\text{Arg}_1\text{(Theme)}}$ further declines in interest rates.]

the alignment shown in Table 2 is obtained.

SEM-I roles	Mapped roles	Features
ARG1	Experiencer	`manager_n_of`
ARG2	Theme	`propositional_m_rel`

Table 2: Alignment instance obtained for the verb *expect*

Since the use of fine-grained features can make the learning process very complex, the WordNet semantic network (Fellbaum, 1998) was also employed to obtain generalisations of nouns. The algorithm described in (Pedersen et al., 2004) was used to disambiguate the sense, given the heads of the verb arguments and the verb itself (by using the mapping from VerbNet senses to WordNet verb senses (Kipper-Schuler, 2005)). Alternatively, a *naive* model has also been proposed, in which these features are simply generalized as nouns. For prepositions, the ontology provided by the SEM-I was used. Other words like adjectives or verbs in arguments were simply generalised as their corresponding type (e.g., *adjectival_rel* or *verbal_rel*).

3.2 Inference of semantic roles with Bayesian Networks

The inference of semantic roles is based on training of BNs by presenting instances of the features extracted, during the learning process. Thus, a training example corresponding to the features shown in Table 2 might be represented as Figure 1 shows, using a first-order approach. After training, the network can infer a proper PropBank (VerbNet) semantic role, given some RMRS role corresponding to some verb. The use of some of these features can be relaxed to test different alternatives.

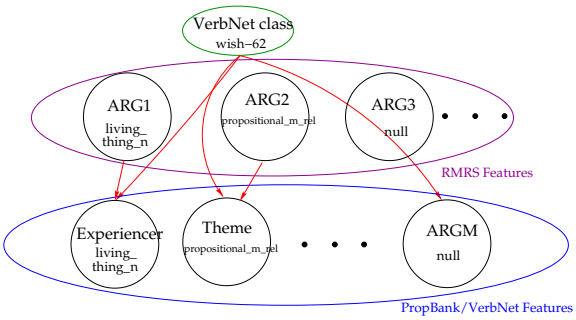


Figure 1: A priori structure of the BN for lexical semantic roles inference.

Two algorithms are used to train the BNs. The Maximum Likelihood (ML) estimation procedure is used when the structure of the model is known. In our experiments, the a priori structure shown in Figure 1 was employed. In the case of the Structural Expectation Maximization (SEM) Algorithm, the initial structure assumed for the ML algorithm serves as an initial state for the network and then the learning phase is executed in order to learn other conditional dependencies and parameters as well. The training procedure is described in Figure 2.

procedure *Train (Model)*
1: **for all** Verbs **do**
2: **for all** Sentences and Parsings which include the current verb **do**
3: Initialize vertices of the network with SEM-I labels and features.
4: Initialize optionally vertices with the corresponding VerbNet class.
5: Initialize edges connecting corresponding features.
6: Append the current features as evidence for the network.
7: **end for**
8: Start Training Model for the current Verb, where Model is ML or SEM.
9: **end for**

Figure 2: Algorithm for training Bayesian Networks for inference of lexical semantic roles

After the training phase, a testing procedure using the Markov Chain Monte Carlo (MCMC) inference engine can be used to infer role labels. Since it is reasonable to think that in some cases the VerbNet class is not known, the presentation of this feature as evidence can be left as optional. Thus, after presenting as evidence the SEM-I related features, a role label with highest probability is obtained after using the MCMC with the current evidence.

4 Experimental results

The experiment uses 10370 sentences from the PropBank corpus which have a mapping to VerbNet (Loper et al., 2007) and are successfully parsed by the ERG (December 2006 version). Up to 10 best parses are recorded for each sentence. The total number of instances, considering that each sentence contains zero or more verbs, is 13589. The algorithm described in section 3.1 managed to find at least one mapping for 10960 of these instances (1020 different verb lexemes). If the number of parsing results is increased to 25 the results are improved (1460 different verb lexemes were found). In the second experiment, the sentences without VerbNet mappings were also included.

The results for the probabilistic models for infering lexical semantic roles are shown in Table 3, where the term *naive* means that no WordNet features were included in the training of the models, but only simple features like *noun_rel* for nouns. On the contrary, when mode is *complete*, WordNet hypernyms up to the 5th level in the hierarchy were used. In this set of experiments the VerbNet class was also included (in the marked cases) during the learning and inference phases.

Corpus	Nr. iter. MCMC	Mode	Model	Verb classes	Accuracy %
PropBank with VerbNet labels	1000	ML	naive		78.41
	10000	ML	naive		84.48
	10000	ML	naive	×	87.92
	1000	ML	complete		84.74
	10000	ML	complete		86.79
	10000	ML	complete	×	87.76
	1000	SEM	naive		84.25
	1000	SEM	complete		87.26
PropBank with PropBank labels	1000	ML	naive		87.46
	1000	SEM	naive		90.27

Table 3: Results of role mapping with probabilistic model

In Table 3, the errors are due to the problems introduced by the alternation behaviour of the verbs, which are not encoded in the SEM-I labelling and

also some contradictory annotations in the mapping between PropBank and VerbNet. Furthermore, the use of the WordNet features may also generate a more complex model or problems derived from the disambiguation process and hence produce errors in the inference phase. In addition, it is reasonable to use the VerbNet class information in the learning and inference phases, which in fact improves slightly the results. The outcomes also show that the use of the SEM algorithm improves accuracy slightly, meaning that the conditional dependency assumptions were reasonable, but still not perfect.

The model can be slightly modified for verb class inference, by adding conditional dependencies between the VerbNet class and SEM-I features, which can potentially improve the parse disambiguation task, in a similar way of thinking to (Fujita et al., 2007). For instance, for the following sentence, we derive an incorrect mapping for the verb *stay* to the VerbNet class EXIST-47.1-1 with the (falsely) favored parse where the PP *"in one place"* is treated as an adjunct/modifier. For the correct reading where the PP is a complement to *stay*, the mapping to the correct VerbNet class LODGE-46 is derived, and the correct LOCATION role is identified for the PP.

(3) Regardless of whether [$_{Theme}$ you] hike from lodge to lodge or **stay**$_{\text{LODGE-46}}$ [$_{Location}$ *in one place*] and take day trips, there are plenty of choices.

5 Conclusions and Future Work

In this paper, we have presented a study of mapping between the HPSG parser semantic outputs in form of MRS structures and lexical semantic resources. The experiment result shows that the Bayesian network reliably maps MRS predicate-argument structures to semantic roles. The automatic mapping enables us to enrich the deep parser output with semantic role information. Preliminary experiments have also shown that verb class inference can potentially improve the parse disambiguation task. Although we have been focusing on improving the deep parsing system with the mapping to annotated semantic resources, it is important to realise that the mapping also enables us to enrich the shallow semantic annotations with more fine-grained analyses from the deep grammars. Such analyses can eventually be helpful for applications like question answering, for instance, and will be investigated in the future.

References

Collin Baker, Charles Fillmore, and John Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics*, pages 86–90, San Francisco, CA.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan.

Ann Copestake, Dan P. Flickinger, and Ivan A. Sag. 2006. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(4):281–332.

Ann Copestake. 2006. Robust minimal recursion semantics. Working Paper.

Christiane D. Fellbaum. 1998. *WordNet – An Electronic Lexical Database*. MIT Press.

Dan Flickinger, Jan T. Lønning, Helge Dyvik, Stephan Oepen, and Francis Bond. 2005. SEM-I rational MT. Enriching deep grammars with a semantic interface for scalable machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 165 – 172, Phuket, Thailand.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications.

Sanae Fujita, Francis Bond, Stephan Oepen, and Takaaki Tanaka. 2007. Exploiting semantic information for hpsg parse selection. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 25–32, Prague, Czech Republic.

Karin Kipper-Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Edward Loper, Szu ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between Propbank and Verbnet. In *Proceedings of the 7th International Workshop on Computational Linguistics*, Tilburg, the Netherlands.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*.

Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion*

Emiel Krahmer
Tilburg University
Tilburg, The Netherlands
E.J.Krahmer@uvt.nl

Erwin Marsi
Tilburg University
Tilburg, The Netherlands
E.C.Marsi@uvt.nl

Paul van Pelt
Tilburg University
Tilburg, The Netherlands
paul.vanpelt@gmail.com

Abstract

We show that question-based sentence fusion is a better defined task than generic sentence fusion (Q-based fusions are shorter, display less variety in length, yield more identical results and have higher normalized Rouge scores). Moreover, we show that in a QA setting, participants strongly prefer Q-based fusions over generic ones, and have a preference for union over intersection fusions.

1 Introduction

Sentence fusion is a text-to-text generation application, which given two related sentences, outputs a single sentence expressing the information shared by the two input sentences (Barzilay and McKeown 2005). Consider, for example, the following pair of sentences:[1]

(1) Posttraumatic stress disorder (PTSD) is a psychological disorder which is classified as an anxiety disorder in the DSM-IV.

(2) Posttraumatic stress disorder (abbrev. PTSD) is a psychological disorder caused by a mental trauma (also called psychotrauma) that can develop after exposure to a terrifying event.

Fusing these two sentences with the strategy described by Barzilay and McKeown (based on aligning and fusing the respective dependency trees) would result in a sentence like (3).

(3) Posttraumatic stress disorder (PTSD) is a psychological disorder.

Barzilay and McKeown (2005) argue convincingly that employing such a fusion strategy in a multi-document summarization system can result in more informative and more coherent summaries.

It should be noted, however, that there are multiple ways to fuse two sentences. Besides fusing the shared information present in both sentences, we can conceivably also fuse them such that *all* information present in either of the sentences is kept, without any redundancies. Marsi and Krahmer (2005) refer to this latter strategy as **union fusion** (as opposed to **intersection fusion**, as in (3)). A possible union fusion of (1) and (2) would be:

(4) Posttraumatic stress disorder (PTSD) is a psychological disorder, which is classified as an anxiety disorder in the DSM-IV, caused by a mental trauma (also called psychotrauma) that can develop after exposure to a terrifying event.

Marsi and Krahmer (2005) propose an algorithm which is capable of producing both fusion types. Which type is more useful is likely to depend on the kind of application and information needs of the user, but this is essentially still an open question.

*Thanks are due to Ed Hovy for discussions on the Rouge metrics and to Carel van Wijk for statistical advice. The dataset described in this paper (2200 fusions of pairs of sentences) is available upon request. This research was carried out within the Deaso project (http://daeso.uvt.nl/).

[1] All examples are English translations of Dutch originals.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 193–196,
Columbus, Ohio, USA, June 2008.

However, there is a complication. Daumé III & Marcu (2004) argue that generic sentence fusion is an ill-defined task. They describe experimental data showing that when participants are given two consecutive sentences from a single document and are asked to fuse them (in the intersection sense), different participants produce very different fusions. Naturally, if human participants cannot reliably perform fusions, evaluating automatic fusion strategies is always going to be a shaky business. The question is *why* different participants come to different fusions. One possibility, which we explore in this paper, is that it is the generic nature of the fusion which causes problems. In particular, we hypothesize that fusing two sentences in the context of a preceding question (the natural setting in QA applications) results in more agreement among humans. A related question is of course what the results would be for union fusion. Will people agree more on the unions than on the intersections? And is the effect of a preceding question the same for both kinds of fusion? In Experiment I, below, we address these questions, by collecting and comparing four different fusions for various pairs of related sentences, both generic and question-based ones, and both intersection and union ones.

While it seems a reasonable hypothesis that question-based fusions will lead to more agreement among humans, the really interesting question is which fusion strategy (if any) is most appreciated by users in a task-based evaluation. Given that Experiment I gives us four different fusions per pair of sentence, an interesting follow-up question is which leads to the best answers in a QA setting. Do participants prefer concise (intersection) or complete (union) answers? And does it matter whether the fusion was question-based or not? In Experiment II, we address these questions via an evaluation experiment using a (simulated) medical question-answering system, in which participants have to rank four answers (resulting from generic and question-based intersection and union fusions) for different medical questions.

2 Experiment I: Data-collection

Method To collect pairs of related sentences to be fused under different conditions, we proceeded as follows. As our starting point we used a set of 100 medical questions compiled as a benchmark for evaluating medical QA systems, where all correct answers were manually retrieved from the available text material. Based on this set, we randomly selected 25 questions for which more than one answer could be found (otherwise there would be nothing to fuse), and where the first two answer sentences shared at least some information (otherwise intersection fusion would be impossible).

Fusion type	**Length M (SD)**	**# Id.**
Generic Intersection	15.6 (2.9)	73
Q-Based Intersection	8.1 (2.5)	189
Generic Union	31.2 (7.8)	109
Q-Based Union	19.2 (4.7)	134

Table 1: Mean sentence length (plus Standard Deviation) and number of identical fusion results as a function of fusion type ($n = 550$ for each type).

Participants were 44 native speakers of Dutch (20 women) with an average age of 30.1 years, none with a background in sentence fusion research. Experiment I has a mixed between-within subjects design. Participants were randomly assigned to either the intersection or the union condition, and within each condition they first had to produce 25 generic and then 25 question-based fusions. In the latter case, participants were given the original question used to retrieve the sentences to be fused.

The experiment was run using a web-based script. Participants were told that the purpose of the experiment was merely to gather data, they were not informed about our interest in generic vs question based fusion. Before participants could start with their task, the concept of sentence fusion (either fusion or intersection, depending on the condition) was explained, using a number of worked examples. After this, the actual experiment started.

Results First consider the descriptive statistics in Table 1. Naturally, intersection fusion leads to shorter sentences on average than union fusion. More interestingly, question (Q)-based fusions lead to significantly shorter sentences than their generic counterparts (intersection $t = 9.1, p < .001$, union: $t = 6.1, p < .001$, two-tailed). Also note that

	Generic Intersection	Q-Based Intersection	Generic Union	Q-Based Union
Rouge-1	.036	.068	.035	.041
Rouge-SU4	.014	.038	.018	.020
Rouge-SU9	.014	.040	.016	.020

Table 2: Average Rouge-1, Rouge-SU4 and Rouge-SU9 (normalized for sentence length) as a function of fusion type.

the variation among participants decreases in the Q-based conditions (lower standard deviations). This suggests that participants in the Q-based conditions indeed show less variety in their fusions than participants in the generic conditions. This is confirmed by the number of identical (i.e., duplicated) fusions, which is indeed higher in the Q-based conditions, although the difference is only significant for intersections ($\chi^2(1) = 51.3, p < .001$).

We also computed average Rouge-1, Rouge-SU4 and Rouge-SU9 scores for each set of fusions, to be able to quantify the overlap between participants in the various conditions. One complication is that these metrics are sensitive to sentence-length (longer sentences are more likely to contain overlapping words than shorter ones), hence in Table 2 we report on Rouge scores that are normalized with respect sentence length. The resulting picture is surprisingly consistent: Q-based fusion on all three metrics results in higher normalized Rouge scores, where the difference is generally small in the case of union, and rather substantial for intersection.

3 Experiment II: Evaluation

The previous experiment indicates that Q-based fusion is indeed a better-defined summarization task than generic fusion, in this experiment we address the question which kind of fusion participants prefer in a QA application.

Method We selected 20 from the 25 questions used in Experiment I, for which we made sure that the fusions in the four categories resulted in sentences with a sufficiently different content. For each question, one representative sentence was selected from the 22 fusions produced by participants in Experiment I, for each of the four categories (Q-based intersection, Q-based union, Generic intersection and Generic union). This representative sentence was the most frequent result for that particular category. When no such sentence was present for a particular task, a random selection was made.

Fusion type	Mean Rank
Q-Based Union	1.888
Q-Based Intersection	2.471
Generic Intersection	2.709
Generic Union	2.932

Table 4: Mean rank from 1 (= "best") to 4 (="worst") as a function of fusion type.

Participants were 38 native speakers of Dutch (17 men), with an average age of 39.4 years. None had participated in Experiment I and none had a background in sentence fusion research. Participants were confronted with the selected 20 questions, one at a time. For each question, participants saw four alternative answers (one from each category). Figure 3 shows one question, with four different fusions derived by participants from example sentences (1) and (2). Naturally, the labels for the 4 fusion strategies were not part of the experiment. Participants were asked to rank the 4 answers from "best" (rank 1) to "worst" (rank 4), via a forced choice paradigm (i.e., they also had to make a choice if they felt that two answers were roughly as good). Experiment II had a within-subjects design, which means that all 38 participants ranked the answers for all 20 questions.

Results Table 4 gives the mean rank for the four fusion types. To test for significance, we performed a repeated measures Analysis of Variance (ANOVA) with fusion type and question as the independent variables and average rank as the dependent variable. A main effect was found of fusion type ($F(3, 111) = 20.938, p < .001, \eta^2 = .361$).

	What is PTSD?
Generic Intersection	Posttraumatic stress disorder (PTSD) is a psychological disorder.
Q-based Intersection	PTSD stands for posttraumatic stress disorder and is a psychological disorder.
Generic Union	Posttraumatic stress disorder (PTSD) is a psychological disorder, which is classified as an anxiety disorder in the DSM-IV, caused by a mental trauma (also called psychotrauma) that can develop after exposure to a terrifying event.
Q-based Union	PTSD (posttraumatic stress disorder) is a psychological disorder caused by a mental trauma (also called psychotrauma) that can develop after exposure to a terrifying event.

Table 3: Example question from Experiment II, with four possible answers, based on different fusions strategies (obtained in Experiment I).

Pairwise comparisons using the Bonferroni method show that all comparisons are statistically significant (at $p < .001$) except for the one between Generic Intersection and Generic Union. Thus, in particular: Q-based union is ranked significantly higher than Q-based intersection, which in turn is ranked significantly higher than both Generic union and intersection (whose respective ranks are not significantly different).

The ANOVA analysis also revealed a significant interaction between question and type of fusion ($F(57, 2109) = 7.459, p < .001, \eta^2 = .168$).[2] What this means is that relative ranking varies for different questions. To better understand this interaction, we performed a series of Friedman tests for each question (the Friedman test is a standard non-parametric test for ranked data). The Friedman analyses revealed that the overall pattern (Q-based union $>$ Q-based intersection $>$ Generic Union / Intersection) was found to be significant for 13 out of the 20 questions. For four of the remaining seven questions, Q-based union ranked first as well, while for two questions Q-based intersection was ranked as the best answer. For the remaining question, there was no significant difference between the four fusion types.

4 Conclusion and discussion

In this paper we have addressed two questions. First: is Q-based fusion a better defined task than generic fusion? Here, the answer seems to be "yes": Q-based fusions are shorter, display less variety in length, result in more identically fused sentences and have higher normalized Rouge scores, where the differences are larger for intersection than for union. Inspection of the fused sentences reveals that there is simply more potential variation on the word level (do I select this word from one input sentence or from the other?) for union fusion than for intersection fusion. Second: which kind of fusion (if any) do users of a medical QA system prefer? Here a consistent preference order was found, with rank 1 = Q-based union, rank 2 = Q-based Intersection, rank 3/4 = Generic intersection / union. Thus: participants clearly prefer Q-based fusions, and prefer more complete answers over shorter ones.

In future research, we intend to collect new data with different questions per sentence pair, to find out to what extent the question and its phrasing drive the fusion process. In addition, we will also let sentences from different domains be fused, based on the hypothesis that fusion strategies may differ across domains.

[2] Naturally, there can be no main effect of question, since there is no variance; the ranks 1-4 are fixed for each question.

References

Regina Barzilay and Kathleen McKeown. 2005. Sentence Fusion for Multidocument News Summarization. *Computational Linguistics*, 31(3), 297-328.

Hal Daumé III and Daniel Marcu. 2004. Generic Sentence Fusion is an Ill-Defined Summarization Task. *Proceedings of the ACL Text Summarization Branches Out Workshop*, Barcelona, Spain.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using N-gram co-occurrence statistics. *Proceedings of NAACL '03*, Edmonton, Canada.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in Sentence Fusion. *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, UK.

Intrinsic vs. Extrinsic Evaluation Measures for Referring Expression Generation

Anja Belz
Natural Language Technology Group
University of Brighton
Brighton BN2 4GJ, UK
a.s.belz@brighton.ac.uk

Albert Gatt
Department of Computing Science
University of Aberdeen
Aberdeen AB24 3UE, UK
a.gatt@abdn.ac.uk

Abstract

In this paper we present research in which we apply (i) the kind of intrinsic evaluation metrics that are characteristic of current comparative HLT evaluation, and (ii) extrinsic, human task-performance evaluations more in keeping with NLG traditions, to 15 systems implementing a language generation task. We analyse the evaluation results and find that there are no significant correlations between intrinsic and extrinsic evaluation measures for this task.

1 Introduction

In recent years, NLG evaluation has taken on a more comparative character. NLG now has evaluation results for comparable, but independently developed systems, including results for systems that regenerate the Penn Treebank (Langkilde, 2002) and systems that generate weather forecasts (Belz and Reiter, 2006). The growing interest in comparative evaluation has also resulted in a tentative interest in shared-task evaluation events, which led to the first such event for NLG (the Attribute Selection for Generation of Referring Expressions, or ASGRE, Challenge) in 2007 (Belz and Gatt, 2007), with a second event (the Referring Expression Generation, or REG, Challenge) currently underway.

In HLT in general, comparative evaluations (and shared-task evaluation events in particular) are dominated by intrinsic evaluation methodologies, in contrast to the more extrinsic evaluation traditions of NLG. In this paper, we present research in which we applied both intrinsic and extrinsic evaluation methods to the same task, in order to shed light on how the two correlate for NLG tasks. The results show a surprising lack of correlation between the two types of measures, suggesting that intrinsic metrics and extrinsic methods can represent two very different views of how well a system performs.

2 Task, Data and Systems

Referring expression generation (REG) is concerned with the generation of expressions that describe entities in a given piece of discourse. REG research goes back at least to the 1980s (Appelt, Grosz, Joshi, McDonald and others), but the field as it is today was shaped in particular by Dale and Reiter's work (Dale, 1989; Dale and Reiter, 1995). REG tends to be divided into the stages of *attribute selection* (selecting properties of entities) and *realisation* (converting selected properties into word strings). Attribute selection in its standard formulation was the shared task in the ASGRE Challenge: given an intended referent ('target') and the other domain entities ('distractors') each with possible attributes, select a set of attributes for the target referent.

The ASGRE data (which is now publicly available) consists of all 780 singular items in the TUNA corpus (Gatt et al., 2007) in two subdomains, consisting of descriptions of furniture and people. Each data item is a paired attribute set (as derived from a human-produced RE) and domain representation (target and distractor entities represented as possible attributes and values).

ASGRE participants were asked to submit the outputs produced by their systems for an unseen test data set. The outputs from 15 of these systems, shown in the left column of Table 1, were used in

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 197–200,
Columbus, Ohio, USA, June 2008.

the experiments reported below. Systems differed in terms of whether they were trainable, performed exhaustive search and hardwired use of certain attributes types, among other algorithmic properties (see the ASGRE papers for full details). In the case of one system (IS-FBS), a buggy version was originally submitted and used in Exp 1. It was replaced in Exp 2 by a corrected version; the former is marked by a * in what follows.

3 Evaluation Methods

1. Extrinsic evaluation measures: We conducted two task-performance evaluation experiments (the first was part of the ASGRE Challenge, the second is new), in which participants identified the referent denoted by a description by clicking on a picture in a visual display of target and distractor entities. To enable subjects to read the outputs of peer systems, we converted them from the attribute-value format described above to something more readable, using a simple attribute-to-word converter.

Both experiments used a Repeated Latin Squares design, and involved 30 participants and 2,250 individual trials (see Belz & Gatt (2007) for full details).

In Exp 1, subjects were shown the domain on the same screen as the description. Two dependent measures were used: (i) combined reading and identification time (RIT), measured from the point at which the description and pictures appeared on the screen to the point at which a picture was selected by mouse-click; and (ii) error rate (ER-1).

In Exp 2, subjects first read the description and then initiated the presentation of domain entities. We computed: (i) reading time (RT), measured from the presentation of a description to the point where a subject requested the presentation of the domain; (ii) identification time (IT), measured from the presentation of the domain to the point where a subject clicked on a picture; and (iii) error rate (ER-2).

2. REG-specific intrinsic measures: *Uniqueness* is the proportion of attribute sets generated by a system which identify the referent uniquely (i.e. none of the distractors). *Minimality* is the proportion of attribute sets which are minimal as well as unique (i.e. there is no smaller unique set of attributes). These measures were included because they are commonly named as desiderata for attribute selection algorithms in the REG field (Dale, 1989). The minimality check used in this paper treats referent type as a simple attribute, as the ASGRE systems tended to do.[1]

3. Set-similarity measures: The *Dice similarity coefficient* computes the similarity between a peer attribute set A_1 and a (human-produced) reference attribute set A_2 as $\frac{2\times|A_1\cap A_2|}{|A_1|+|A_2|}$. *MASI* (Passonneau, 2006) is similar but biased in favour of similarity where one set is a subset of the other.

4. String-similarity measures: In order to apply string-similarity metrics, peer and reference outputs were converted to word-strings by the method described under 1 above. *String-edit distance* (SE) is straightforward Levenshtein distance with a substitution cost of 2 and insertion/deletion cost of 1. We also used the version of string-edit distance ('SEB') of Bangalore et al. (2000) which normalises for length. *BLEU* computes the proportion of word n-grams ($n \leq 4$ is standard) that a peer output shares with several reference outputs. The *NIST* MT evaluation metric (Doddington, 2002) is an adaptation of BLEU which gives more importance to less frequent (hence more informative) n-grams. We also used two versions of the ROUGE metric (Lin and Hovy, 2003), *ROUGE-2* and *ROUGE-SU4* (based on non-contiguous, or 'skip', n-grams), which were official scores in the DUC 2005 summarization task.

4 Results

Results for all evaluation measures and all systems are shown in Table 1. Uniqueness results are not included, as all systems scored 100%.

We ran univariate analyses of variance (ANOVAs) using SYSTEM as the independent variable (15 levels), testing its effect on the extrinsic task-performance measures. For error rate (ER), we used a Kruskal-Wallis ranks test to compare identification accuracy rates across systems[2]. The main effect of SYSTEM was significant on RIT ($F(14, 2249) = 6.401$, $p < .001$), RT ($F(14, 2249) = 2.56$, $p < .01$), and IT ($F(14, 2249) = 1.93$, $p < .01$). In neither experiment was there a significant effect on ER.

[1] As a consequence, the Minimality results we report here look different from those in the ASGRE report.

[2] A non-paramteric test was more appropriate given the large number of zero values in ER proportions, and a high dependency of variance on the mean.

	extrinsic					*REG*	*string-similarity*						*set-similarity*	
	RIT	RT	IT	ER-1	ER-2	Min	RSU4	R-2	NIST	BLEU	SE	SEB	Dice	MASI
CAM-B	2784.80	1309.07	1952.39	9.33	5.33	8.11	.673	.647	2.70	.309	4.42	.307	.620	.403
CAM-BU	2659.37	1251.32	1877.95	9.33	4	10.14	.663	.638	2.61	.317	4.23	.359	.630	.420
CAM-T	2626.02	1475.31	1978.24	10	5.33	0	.698	.723	3.50	.415	3.67	.496	.725	.560
CAM-TU	2572.82	1297.37	1809.04	8.67	4	0	.677	.691	3.28	.407	3.71	.494	.721	.557
DIT-DS	2785.40	1304.12	1859.25	10.67	2	0	.651	.679	4.23	.457	3.55	.525	.750	.595
GR-FP	2724.56	1382.04	2053.33	8.67	3.33	4.73	.65	.649	3.24	.358	3.87	.441	.689	.480
GR-SC	2811.09	1349.05	1899.59	11.33	2	4.73	.644	.644	2.42	.305	4	.431	.671	.466
IS-FBN	3570.90	1837.55	2188.92	15.33	6	1.35	.771	.772	4.75	.521	3.15	.438	.770	.601
IS-FBS	–	1461.45	2181.88	–	7.33	100	.485	.448	2.11	.166	5.53	.089	.368	.182
*IS-FBS	4008.99	–	–	10	–	39.86	–	–	–	–	–	–	.527	.281
IS-IAC	2844.17	1356.15	1973.19	8.67	6	0	.612	.623	3.77	.442	3.43	.559	.746	.597
NIL	1960.31	1482.67	1960.31	10	5.33	20.27	.525	.509	3.32	.32	4.12	.447	.625	.477
T-AS+	2652.85	1321.20	1817.30	9.33	4.67	0	.671	.684	2.62	.298	4.24	.37	.660	.452
T-AS	2864.93	1229.42	1766.35	10	4.67	0	.683	.692	2.99	.342	4.10	.393	.645	.422
T-RS+	2759.76	1278.01	1814.93	6.67	1.33	0	.677	.697	2.85	.303	4.32	.36	.669	.459
T-RS	2514.37	1255.28	1866.94	8.67	4.67	0	.694	.711	3.16	.341	4.18	.383	.655	.432

Table 1: Results for all systems and evaluation measures (ER-1 = error rate in Exp 1, ER-2 = error rate in Exp 2). (R = ROUGE; system IDs as in the ASGRE papers, except GR = GRAPH; T = TITCH).

Table 2 shows correlations between the automatic metrics and the task-performance measures from Exp 1. RIT and ER-1 are not included because of the presence of *IS-FBS in Exp 1 (but see individual results below). For reasons of space, we refer the reader to the table for individual correlation results.

We also computed correlations between the task-performance measures across the two experiments (leaving out the IS-FBS system). Correlation between RIT and RT was .827**; between RIT and IT .675**; and there was no significant correlation between the error rates. The one difference evident between RT and IT is that ER correlates only with IT (not RT) in Exp 2 (see Table 2).

5 Discussion

In Table 2, the four broad types of metrics we have investigated (task-performance, REG-specific, string similarity, set similarity) are indicated by vertical and horizontal lines. The results within each of the resulting boxes are very homogeneous. There are significant (and mostly strong) correlations not only among the string-similarity metrics and among the set-similarities, but also across the two types. There are also significant correlations between the three task-performance measures.

However, the correlation figures between the task-performance measures and all others are weak and not significant. The one exception is the correlation between NIST and RT which is actually in the wrong direction (better NIST implies *worse* reading times).

This is an unambiguous result and it shows clearly that similarity to human-produced reference texts is not necessarily indicative of quality as measured by human task performance.

The emergence of comparative evaluation in NLG raises the broader question of how systems that generate language should be compared. In MT and summarisation it is more or less taken as read that systems which generate more human-like language are better systems. However, it has not been shown that more human-like outputs result in better performance from an extrinsic perspective. Intuitively, it might be expected that higher humanlikeness entails better task-performance (here, shorter reading/identification times, lower error). The lack of significant covariation between intrinsic and extrinsic measures in our experiments suggests otherwise.

6 Conclusions

Our aim in this paper was to shed light on how the intrinsic evaluation methodologies that dominate current comparative HLT evaluations correlate with human task-performance evaluations more in keeping with NLG traditions. We used the data and systems from the recent ASGRE Challenge, and compared a total of 17 different evaluation methods for 15 different systems implementing the ASGRE task.

Our most striking result is that none of the metrics that assess humanlikeness correlate with any of the task-performance measures, while strong correlations are observed *within* the two classes of mea-

	extrinsic			*REG*	*string-similarity*						*set-similarity*	
	RT	IT	ER-2	Min	R-SU4	R-2	NIST	BLEU	SE	SEB	Dice	MASI
RT	1	.8**	.46	.18	.10	.05	.54*	.39	-.30	.02	.12	.23
IT	.8**	1	.59*	.56*	-.24	-.33	.22	.04	.09	-.31	-.28	-.17
ER-2	.46	.59*	1	.51	-.29	-.36	.03	-.08	.22	-.34	-.39	-.29
Min	.18	.56*	.51	1	-.76**	-.81**	-.46	-.66**	.79**	-.8**	-.90**	-.79**
R-SU4	.10	-.24	-.29	-.76**	1	.98**	.45	.63*	-.63*	.42	.72**	.57*
R-2	.05	-.33	-.36	-.81**	.98**	1	.51	.68**	-.69**	.53*	.78**	.65**
NIST	.54*	.22	.03	-.46	.45	.51	1	.94**	-.84**	.68**	.74**	.82**
BLEU	.39	.04	-.08	-.66**	.63*	.68**	.94**	1	-.96**	.82**	.89**	.93**
SE	-.30	.09	.22	.79**	-.63*	-.69**	-.84**	-.96**	1	-.92**	-.96**	-.97**
SEB	.02	-.31	-.34	-.8**	.42	.53*	.68**	.82**	-.92**	1	.92**	.95**
Dice	.12	-.28	-.39	-.90**	.72**	.78**	.74**	.89**	-.96**	.92**	1	.97**
MASI	.23	-.17	-.29	-.79**	.57*	.65**	.82**	.93**	-.97**	.95**	.97**	1

Table 2: Pairwise correlations between all automatic measures and the task-performance results from Exp 2. (* = significant at .05; ** at .01). R = ROUGE.

sures – intrinsic and extrinsic. Somewhat worryingly, our results show that a system's ability to produce human-like outputs may be completely unrelated to its effect on human task-performance.

Our main conclusions for REG evaluation are that we need to be cautious in relying on humanlikeness as a quality criterion, and that we leave extrinsic evaluation behind at our peril as we move towards more comparative forms of evaluation.

Given that the intrinsic metrics that dominate in competetive HLT evaluations are not assessed in terms of correlation with extrinsic notions of quality, our results sound a more general note of caution about using intrinsic measures (and humanlikeness metrics in particular) without extrinsic validation.

Acknowledgments

We gratefully acknowledge the contribution made to the evaluations by the faculty and staff at Brighton University who participated in the identification experiments. Thanks are also due to Robert Dale, Kees van Deemter, Ielka van der Sluis and the anonymous reviewers for very helpful comments. The biggest contribution was, of course, made by the participants in the ASGRE Challenge who created the systems involved in the evaluations.

References

S. Bangalore, O. Rambow, and S. Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the 1st International Conference on Natural Language Generation (INLG '00)*, pages 1–8.

A. Belz and A. Gatt. 2007. The attribute selection for GRE challenge: Overview and evaluation results. In *Proceedings of the 2nd UCNLG Workshop: Language Generation and Machine Translation (UCNLG+MT)*, pages 75–83.

A. Belz and E. Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proc. EACL'06*, pages 313–320.

R. Dale and E. Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

R. Dale. 1989. Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. ARPA Workshop on Human Language Technology*.

A. Gatt, I. van der Sluis, and K. van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the 11th European Workshop on Natural Language Generation (ENLG'07)*, pages 49–56.

I. Langkilde. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG '02)*.

C.-Y. Lin and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. HLT-NAACL 2003*, pages 71–78.

R. Passonneau. 2006. Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In *Proceedings of the 5th Language Resources and Evaluation Converence (LREC'06)*.

Correlation between ROUGE and Human Evaluation of Extractive Meeting Summaries

Feifan Liu, Yang Liu
The University of Texas at Dallas
Richardson, TX 75080, USA
ffliu,yangl@hlt.utdallas.edu

Abstract

Automatic summarization evaluation is critical to the development of summarization systems. While ROUGE has been shown to correlate well with human evaluation for content match in text summarization, there are many characteristics in multiparty meeting domain, which may pose potential problems to ROUGE. In this paper, we carefully examine how well the ROUGE scores correlate with human evaluation for extractive meeting summarization. Our experiments show that generally the correlation is rather low, but a significantly better correlation can be obtained by accounting for several unique meeting characteristics, such as disfluencies and speaker information, especially when evaluating system-generated summaries.

1 Introduction

Meeting summarization has drawn an increasing attention recently; therefore a study on the automatic evaluation metrics for this task is timely. Automatic evaluation helps to advance system development and avoids the labor-intensive and potentially inconsistent human evaluation. ROUGE (Lin, 2004) has been widely used for summarization evaluation. In the news article domain, ROUGE scores have been shown to be generally highly correlated with human evaluation in content match (Lin, 2004). However, there are many differences between written texts (e.g., news wire) and spoken documents, especially in the meeting domain, for example, the presence of disfluencies and multiple speakers, and the lack of structure in spontaneous utterances. The question of whether ROUGE is a good metric for meeting summarization is unclear. (Murray et al., 2005) have reported that ROUGE-1 (unigram match) scores have low correlation with human evaluation in meetings.

In this paper we investigate the correlation between ROUGE and human evaluation of extractive meeting summaries and focus on two issues specific to the meeting domain: disfluencies and multiple speakers. Both human and system generated summaries are used. Our analysis shows that by integrating meeting characteristics into ROUGE settings, better correlation can be achieved between the ROUGE scores and human evaluation based on Spearman's rho in the meeting domain.

2 Related work

Automatic summarization evaluation can be broadly classified into two categories (Jones and Galliers, 1996): intrinsic and extrinsic evaluation. Intrinsic evaluation, such as relative utility based metric proposed in (Radev et al., 2004), assesses a summarization system in itself (for example, informativeness, redundancy, and coherence). Extrinsic evaluation (Mani et al., 1998) tests the effectiveness of a summarization system on other tasks. In this study, we concentrate on the automatic intrinsic summarization evaluation. It has been extensively studied in text summarization. Different approaches have been proposed to measure matches using words or more meaningful semantic units, for example, ROUGE (Lin, 2004), factoid analysis (Teufel and Halteren, 2004), pyramid method (Nenkova and Passonneau, 2004), and Basic Element (BE) (Hovy et al., 2006).

With the increasing recent research of summarization moving into speech, especially meeting recordings, issues related to spoken language are yet to be explored for their impact on the evaluation metrics. Inspired by automatic speech recognition (ASR) evaluation, (Hori et al., 2003) proposed the summarization accuracy metric (SumACCY) based on a word network created by merging manual summaries. However (Zhu and Penn, 2005) found a statistically significant difference between the ASR-inspired metrics and those taken from text summarization (e.g., RU, ROUGE) on a subset of the Switchboard data. ROUGE has been used in meeting summarization evaluation (Murray et al., 2005; Galley, 2006), yet the question remained whether ROUGE is a good metric for the meeting domain. (Murray et al., 2005) showed low correlation of ROUGE and human evaluation in meeting summarization evaluation; however, they

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 201–204,
Columbus, Ohio, USA, June 2008.

simply used ROUGE as is and did not take into account the meeting characteristics during evaluation.

In this paper, we ask the question of whether ROUGE correlates with human evaluation of extractive meeting summaries and whether we can modify ROUGE to account for the meeting style for a better correlation with human evaluation.

3 Experimental Setup

3.1 Data

We used the ICSI meeting data (Janin et al., 2003) that contains naturally-occurring research meetings. All the meetings have been transcribed and annotated with dialog acts (DA) (Shriberg et al., 2004), topics, and extractive summaries (Murray et al., 2005).

For this study, we used the same 6 test meetings as in (Murray et al., 2005; Galley, 2006). Each meeting already has 3 human summaries from 3 common annotators. We recruited another 3 human subjects to generate 3 more human summaries, in order to create more data points for a reliable analysis. The Kappa statistics for those 6 different annotators varies from 0.11 to 0.35 for different meetings. The human summaries have different length, containing around 6.5% of the selected DAs and 13.5% of the words respectively. We used four different system summaries for each of the 6 meetings: one based on the MMR method in MEAD (Carbonell and Goldstein, 1998; et al., 2003), the other three are the system output from (Galley, 2006; Murray et al., 2005; Xie and Liu, 2008). All the system generated summaries contain around 5% of the DAs and 16% of the words of the entire meeting. Thus, in total we have 36 human summaries and 24 system summaries on the 6 test meetings, on which the correlation between ROUGE and human evaluation is calculated and investigated.

All the experiments in this paper are based on human transcriptions, with a central interest on whether some characteristics of the meeting recordings affect the correlation between ROUGE and human evaluations, without the effect from speech recognition or automatic sentence segmentation errors.

3.2 Automatic ROUGE Evaluation

ROUGE (Lin, 2004) measures the n-gram match between system generated summaries and human summaries. In most of this study, we used the same options in ROUGE as in the DUC summarization evaluation (NIST, 2007), and modify the input to ROUGE to account for the following two phenomena.

- Disfluencies

  Meetings contain spontaneous speech with many disfluencies, such as filled pauses (uh, um), discourse markers (e.g., I mean, you know), repetitions, corrections, and incomplete sentences. There have been efforts on the study of the impact of disfluencies on summarization techniques (Liu et al., 2007; Zhu and Penn, 2006) and human readability (Jones et al., 2003). However, it is not clear whether disfluencies impact automatic evaluation of extractive meeting summarization.

  Since we use extractive summarization, summary sentences may contain difluencies. We hand annotated the transcripts for the 6 meetings and marked the disfluencies such that we can remove them to obtain cleaned up sentences for those selected summary sentences. To study the impact of disfluencies, we run ROUGE using two different inputs: summaries based on the original transcription, and the summaries with disfluencies removed.

- Speaker information

  The existence of multiple speakers in meetings raises questions about the evaluation method. (Galley, 2006) considered some location constrains in meeting summarization evaluation, which utilizes speaker information to some extent. In this study we use the data in separate channels for each speaker and thus have the speaker information available for each sentence. We associate the speaker ID with each word, treat them together as a new 'word' in the input to ROUGE.

3.3 Human Evaluation

Five human subjects (all undergraduate students in Computer Science) participated in human evaluation. In total, there are 20 different summaries for each of the 6 test meetings: 6 human-generated, 4 system-generated, and their corresponding ones with disfluencies removed. We assigned 4 summaries with different configurations to each human subject: human vs. system generated summaries, with or without disfluencies. Each human evaluated 24 summaries in total, for the 6 test meetings.

For each summary, the human subjects were asked to rate the following statements using a scale of 1-5 according to the extent of their agreement with them.

- S1: The summary reflects the discussion flow in the meeting very well.
- S2: Almost all the important topic points of the meeting are represented.
- S3: Most of the sentences in the summary are relevant to the original meeting.
- S4: The information in the summary is not redundant.
- S5: The relationship between the importance of each topic in the meeting and the amount of summary space given to that topic seems appropriate.
- S6: The relationship between the role of each speaker and the amount of summary speech selected for that speaker seems appropriate.
- S7: Some sentences in the summary convey the same meaning.
- S8: Some sentences are not necessary (e.g., in terms of importance) to be included in the summary.
- S9: The summary is helpful to someone who wants to know what are discussed in the meeting.

These statements are an extension of those used in (Murray et al., 2005) for human evaluation of meeting summaries. The additional ones we added were designed to account for the discussion flow in the meetings. Some of the statements above are used to measure similar aspects, but from different perspectives, such as S5 and S6, S4 and S7. This may reduce some accidental noise in human evaluation. We grouped these statements into 4 categories: Informative Structure (IS): S1, S5 and S6; Informative Coverage (IC): S2 and S9; Informative Relevance (IRV): S3 and S8; and Informative Redundancy (IRD): S4 and S7.

4 Results

4.1 Correlation between Human Evaluation and Original ROUGE Score

Similar to (Murray et al., 2005), we also use Spearman's rank coefficient (rho) to investigate the correlation between ROUGE and human evaluation. We have 36 human summaries and 24 system summaries for the 6 meetings in our study. For each of the human summaries, the ROUGE scores are generated using the other 5 human summaries as references. For system generated summaries, we calculate the ROUGE score using 5 human references, and then obtain the average from 6 such setups. The correlation results are presented in Table 1. In addition to the overall average for human evaluation (H_AVG), we calculated the average score for each evaluation category (see Section 3.3). For ROUGE evaluation, we chose the F-measure for R-1 (unigram) and R-SU4 (skip-bigram with maximum gap length of 4), which is based on our observation that other scores in ROUGE are always highly correlated (rho$>$0.9) to either of them for this task. We compute the correlation separately for the human and system summaries in order to avoid the impact due to the inherent difference between the two different summaries.

Correlation on Human Summaries					
	H_AVG	H_IS	H_IC	H_IRV	H_IRD
R-1	0.09	0.22	0.21	0.03	-0.20
R-SU4	0.18	0.33	0.38	0.04	-0.30
Correlation on System Summaries					
R-1	-0.07	-0.02	-0.17	-0.27	-0.02
R-SU4	0.08	0.05	0.01	-0.15	0.14

Table 1: Spearman's rho between human evaluation (H) and ROUGE (R) with basic setting.

We can see that R-SU4 obtains a higher correlation with human evaluation than R-1 on the whole, but still very low, which is consistent with the previous conclusion from (Murray et al., 2005). Among the four categories, better correlation is achieved for information structure (IS) and information coverage (IC) compared to the other two categories. This is consistent with what ROUGE is designed for, "recall oriented understudy gisting evaluation" — we expect it to model IS and IC well by ngram and skip-bigram matching but not relevancy (IRV) and redundancy (IRD) effectively. In addition, we found low correlation on system generated summaries, suggesting it is more challenging to evaluate those summaries both by humans and the automatic metrics.

4.2 Impacts of Disfluencies on Correlation

Table 2 shows the correlation results between ROUGE (R-SU4) and human evaluation on the original and cleaned up summaries respectively. For human summaries, after removing disfluencies, the correlation between ROUGE and human evaluation improves on the whole, but degrades on information structure (IS) and information coverage (IC) categories. However, for system summaries, there is a significant gain of correlation on those two evaluation categories, even though no improvement on the overall average score. Our hypothesis for this is that removing disfluencies helps remove the noise in the system generated summaries and make them more easily to be evaluated by human and machines. In contrast, the human created summaries have better quality in terms of the information content and may not suffer as much from the disfluencies contained in the summary.

Correlation on Human Summaries					
	H_AVG	H_IS	H_IC	H_IRV	H_IRD
Original	0.18	0.33	0.38	0.04	-0.30
Disfluencies removed	0.21	0.21	0.31	0.19	-0.16
Correlation on System Summaries					
Original	0.08	0.05	0.01	-0.15	0.14)
Disfluencies removed	0.08	0.22	0.19	-0.02	-0.07

Table 2: Effect of disfluencies on the correlation between R-SU4 and human evaluation.

4.3 Incorporating Speaker Information

We further incorporated speaker information in ROUGE setting using the summaries with disfluencies removed. Table 3 presents the resulting correlation values between ROUGE SU4 score and human evaluation. For human summaries, adding speaker information slightly degraded the correlation, but it is still better compared to using the original transcripts (results in Table 1). For the system summaries, the overall correlation is significantly improved, with some significant improvement in the information redundancy (IRD) category. This suggests that by leveraging speaker information, ROUGE can assign better credits or penalties to system generated summaries (same words from different speakers will not be counted as a match), and thus yield better correlation with human evaluation; whereas for human summaries, this may not happen often. For similar sentences from different speakers, human annotators are more likely to agree with each

other in their selection compared to automatic summarization.

Correlation on Human Summaries					
Speaker Info.	H_AVG	H_IS	H_IC	H_IRV	H_IRD
NO	0.21	0.21	0.31	0.19	-0.16
YES	0.20	0.20	0.27	0.12	-0.09
Correlation on System Summaries					
NO	0.08	0.22	0.19	-0.02	-0.07
YES	0.14	0.20	0.16	0.02	0.21

Table 3: Effect of speaker information on the correlation between R-SU4 and human evaluation.

5 Conclusion and Future Work

In this paper, we have made a first attempt to systematically investigate the correlation of automatic ROUGE scores with human evaluation for meeting summarization. Adaptations on ROUGE setting based on meeting characteristics are proposed and evaluated using Spearman's rank coefficient. Our experimental results show that in general the correlation between ROUGE scores and human evaluation is low, with ROUGE SU4 score showing better correlation than ROUGE-1 score. There is significant improvement in correlation when disfluencies are removed and speaker information is leveraged, especially for evaluating system-generated summaries. In addition, we observe that the correlation is affected differently by those factors for human summaries and system-generated summaries.

In our future work we will examine the correlation between each statement and ROUGE scores to better represent human evaluation results instead of using simply the average over all the statements. Further studies are also needed using a larger data set. Finally, we plan to investigate meeting summarization evaluation using speech recognition output.

Acknowledgments

The authors thank University of Edinburgh for providing the annotated ICSI meeting corpus and Michel Galley for sharing his tool to process the annotated data. We also thank Gabriel Murray and Michel Galley for letting us use their automatic summarization system output for this study. This work is supported by NSF grant IIS-0714132. Any opinions expressed in this work are those of the authors and do not necessarily reflect the views of NSF.

References

J. Carbonell and J. Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336.

M. Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *EMNLP*, pages 364–372.

C. Hori, T. Hori, and S. Furui. 2003. Evaluation methods for automatic speech summarization. In *EUROSPEECH*, pages 2825–2828.

E. Hovy, C. Lin, L. Zhou, and J. Fukumoto. 2006. Automated summarization evaluation with basic elements. In *LREC*.

A. Janin, D. Baron, J. Edwards, D. Ellis, G. Gelbart, N. Norgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The icsi meeting corpus. In *ICASSP*.

K. S. Jones and J. Galliers. 1996. Evaluating natural language processing systems: An analysis and review. *Lecture Notes in Artificial Intelligence*.

D. Jones, F. Wlof, E. Gilbson, E. Williams, E. Fedorenko, D. Reynolds, and M. Zissman. 2003. Measuring the readability of automatic speech-to-text transcripts. In *EUROSPEECH*, pages 1585–1588.

C. Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out at ACL*, pages 74–81.

Y. Liu, F. Liu, B. Li, and S. Xie. 2007. Do disfluencies affect meeting summarization? a pilot study on the impact of disfluencies. In *MLMI Workshop, Poster Session.*

I. Mani, T. Firmin, D. House, M. Chrzanowski, G. Klein, L. Hirschman, B. Sundheim, and L. Obrst. 1998. The tipster summac text summarization evaluation: Final report. Technical report, The MITRE Corporation.

G. Murray, S. Renals, J. Carletta, and J. Moore. 2005. Evaluating automatic summaries of meeting recordings. In *ACL 2005 MTSE Workshop*, pages 33–40.

A. Nenkova and R. Passonneau. 2004. Evaluating content selection in summarization: the pyramid method. In *HLT/NAACL.*

NIST. 2007. Document understanding conference (DUC). *http://duc.nist.gov/.*

D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, E. Drabek, W. Lam, D. Liu, H. Qi, H. Saggion, S. Teufel, M. Topper, and A. Winkel. 2003. The MEAD Multidocument Summarizer. http://www.summarization.com/mead/.

D. R. Radev, H. Jing, M. Stys, and T. Daniel. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40:919–938.

E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. In *SIGDAL Workshop*, pages 97–100.

S. Teufel and H. Halteren. 2004. Evaluating information content by factoid analysis: Human annotation and stability. In *EMNLP*.

S. Xie and Y. Liu. 2008. Using corpus and knowledge-based similarity measure in maximum marginal relevance for meeting summarization. In *ICASSP*.

X. Zhu and G. Penn. 2005. Evaluation of sentence selection for speech summarization. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

X. Zhu and G. Penn. 2006. Comparing the roles of textual, acoustic and spoken-language features on spontaneous-conversation summarization. In *HLT/NAACL.*

FastSum:
Fast and accurate query-based multi-document summarization

Frank Schilder and **Ravikumar Kondadadi**
Research & Development
Thomson Corp.
610 Opperman Drive, Eagan, MN 55123, USA
FirstName.LastName@Thomson.com

Abstract

We present a fast query-based multi-document summarizer called FastSum based solely on word-frequency features of clusters, documents and topics. Summary sentences are ranked by a regression SVM. The summarizer does not use any expensive NLP techniques such as parsing, tagging of names or even part of speech information. Still, the achieved accuracy is comparable to the best systems presented in recent academic competitions (i.e., Document Understanding Conference (DUC)). Because of a detailed feature analysis using Least Angle Regression (LARS), FastSum can rely on a minimal set of features leading to fast processing times: *1250* news documents in *60* seconds.

1 Introduction

In this paper, we propose a simple method for effectively generating query-based multi-document summaries without any complex processing steps. It only involves sentence splitting, filtering candidate sentences and computing the word frequencies in the documents of a cluster, topic description and the topic title. We use a machine learning technique called regression SVM, as proposed by (Li et al., 2007). For the feature selection we use a new model selection technique called Least Angle Regression (LARS) (Efron et al., 2004).

Even though machine learning approaches dominated the field of summarization systems in recent DUC competitions, not much effort has been spent in finding simple but effective features. Exceptions are the SumBasic system that achieves reasonable results with only one feature (i.e., word frequency in document clusters) (Nenkova and Vanderwende, 2005). Our approach goes beyond SumBasic by proposing an even more powerful feature that proves to be the best predictor in all three recent DUC corpora. In order to prove that our feature is more predictive than other features we provide a rigorous feature analysis by employing LARS.

Scalability is normally not considered when different summarization systems are compared. Processing time of more than several seconds per summary should be considered unacceptable, in particular, if you bear in mind that using such a system should help a user to process lots of data faster. Our focus is on selecting the minimal set of features that are computationally less expensive than other features (i.e., full parse). Since FastSum can rely on a minimal set of features determined by LARS, it can process *1250* news documents in *60* seconds.[1] A comparison test with the MEAD system[2] showed that FastSum is more than 4 times faster.

2 System description

We use a machine learning approach to rank all sentences in the topic cluster for summarizability. We use some features from Microsoft's PYTHY system (Toutonova et al., 2007), but added two new features, which turned out to be better predictors.

First, the pre-processing module carries out tokenization and sentence splitting. We also created a sentence simplification component which is based

[1] 4-way/2.0GHz PIII Xeon 4096Mb Memory

[2] http://www.summarization.com/mead/

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 205–208,
Columbus, Ohio, USA, June 2008.

on a few regular expressions to remove unimportant components of a sentence (e.g., *As a matter of fact,*). This processing step does not involve any syntactic parsing though. For further processing, we ignore all sentences that do not have at least two exact word matches or at least three fuzzy matches with the topic description.[3]

Features are mainly based on word frequencies of words in the clusters, documents and topics. A cluster contains 25 documents and is associated with a topic. The topic contains a topic title and the topic descriptions. The topic title is list of key words or phrases describing the topic. The topic description contains the actual query or queries (e.g., *Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999.*).

The features we used can be divided into two sets; word-based and sentence-based. Word-based features are computed based on the probability of words for the different containers (i.e., cluster, document, topic title and description). At runtime, the different probabilities of all words in a candidate sentence are added up and normalized by length. Sentence-based features include the length and position of the sentence in the document. The starred features **1** and **4** are introduced by us, whereas the others can be found in earlier literature.[4]

- ***1** *Topic title frequency* (1): ratio of number of words t_i in the sentence s that also appear in the topic title $\mathcal{T}$ to the total number of words $t_{1..|s|}$ in the sentence s: $\frac{\sum_{i=1}^{|s|} f_{\mathcal{T}}(t_i)}{|s|}$, where $f_{\mathcal{T}} = \begin{cases} 1 & : \ t_i \in \mathcal{T} \\ 0 & : \ otherwise \end{cases}$

- **2** *Topic description frequency* (2): ratio of number of words t_i in the sentence s that also appear in the topic description $\mathcal{D}$ to the total number of words $t_{1..|s|}$ in the sentence s: $\frac{\sum_{i=1}^{|s|} f_{\mathcal{D}}(t_i)}{|s|}$, where $f_{\mathcal{D}} = \begin{cases} 1 & : \ t_i \in \mathcal{D} \\ 0 & : \ otherwise \end{cases}$

- **3** *Content word frequency*(3): the average content word probability $p_c(t_i)$ of all content words $t_{1..|s|}$ in a sentence s. The content word probability is defined as $p_c(t_i) = \frac{n}{N}$, where n is the number of times the word occurred in the cluster and N is the total number of words in the cluster: $\frac{\sum_{i=1}^{|s|} p_c(t_i)}{|s|}$

- ***4** *Document frequency* (4): the average document probability $p_d(t_i)$ of all content words $t_{1..|s|}$ in a sentence s. The document probability is defined as $p_d(t_i) = \frac{d}{D}$, where d is the number of *documents* the word t_i occurred in for a given cluster and D is the total number of documents in the cluster: $\frac{\sum_{i=1}^{|s|} p_d(t_i)}{|s|}$

The remaining features are *Headline frequency* (**5**), *Sentence length* (**6**), *Sentence position (binary)* (**7**), and *Sentence position (real)* (**8**)

Eventually, each sentence is associated with a score which is a linear combination of the above mentioned feature values. We ignore all sentences that do not have at least two exact word matches.[5] In order to learn the feature weights, we trained a SVM on the previous year's data using the same feature set. We used a regression SVM. In regression, the task is to estimate the functional dependence of a dependent variable on a set of independent variables. In our case, the goal is to estimate the score of a sentence based on the given feature set. In order to get training data, we computed the word overlap between the sentences from the document clusters and the sentences in DUC model summaries. We associated the word overlap score to the corresponding sentence to generate the regression data. As a last step, we use the pivoted QR decomposition to handle redundancy. The basic idea is to avoid redundancy by changing the relative importance of the rest of the sentences based on the currently selected sentence. The final summary is created from the ranked sentence list after the redundancy removal step.

3 Results

We compared our system with the top performing systems in the last two DUC competitions. With our best performing features, we get ROUGE-2 (Lin, 2004) scores of 0.11 and 0.0925 on 2007 and 2006

[3]Fuzzy matches are defined by the OVERLAP similarity (Bollegala et al., 2007) of at least 0.1.

[4]The numbers are used in the feature analysis, as in figure 2.

[5]This threshold was derived experimentally with previous data.

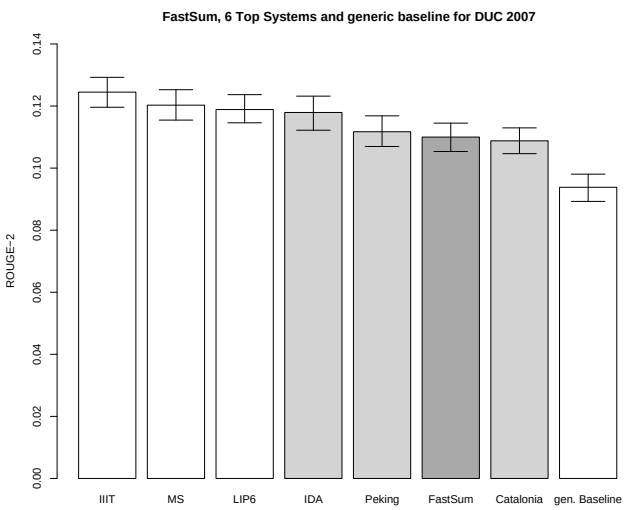


Figure 1: ROUGE-2 results including 95%-confidence intervals for the top 6 systems, FastSum and the generic baseline for DUC 2007

DUC data, respectively. These scores correspond to rank 6th for DUC 2007 and the 2nd rank for DUC 2006. Figure 1 shows a graphical comparison of our system with the top 6 systems in DUC 2007. According to an ANOVA test carried out by the DUC organizers, these 6 systems are significant better than the remaining 26 participating systems.

Note that our system is better than the PYTHY system for 2006, if no sentence simplification was carried out (DUC 2006: 0.089 (without simplification); 0.096 (with simplification)). Sentence simplification is a computationally expensive process, because it requires a syntactic parse.

We evaluated the performance of the FastSum algorithm using each of the features separately. Table 1 shows the ROUGE score (recall) of the summaries generated when we used each of the features by themselves on 2006 and 2007 DUC data, trained on the data from the respective previous year. Using only the Document frequency feature by itself leads to the second best system for DUC 2006 and to the tenth best system for DUC 2007.

This first simple analysis of features indicates that a more rigorous feature analysis would have benefits for building simpler models. In addition, feature selection could be guided by the complexity of the features preferring those features that are computationally inexpensive.

Feature name	2007	2006
Title word frequency	0.096	0.0771
Topic word frequency	0.0996	0.0883
Content word frequency	0.1046	0.0839
Document frequency	**0.1061**	**0.0903**
Headline frequency	0.0938	0.0737
Sentence length	0.054	0.0438
Sentence position(binary)	0.0522	0.0484
Sentence position (real-valued)	0.0544	0.0458

Table 1: ROUGE-2 scores of individual features

We chose a so-called model selection algorithm to find a minimal set of features. This problem can be formulated as a shrinkage and selection method for linear regression. The Least Angle Regression (LARS) (Efron et al., 2004) algorithm can be used for computing the least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996).At each stage in LARS, the feature that is most correlated with the response is added to the model. The coefficient of the feature is set in the direction of the sign of the feature's correlation with the response.

We computed LARS on the DUC data sets from the last three years. The graphical results for 2007 are shown in figure 2. In a LARS graph, features are plotted on the x-axis and the corresponding coefficients are shown on y-axis. The value on the x-axis is the ratio of norm of the coefficent vector to the maximal norm with no constraint. The earlier a feature appears on the x-axis, the better it is. Table 2 summarizes the best four features we determined with LARS for the three available DUC data sets.

Year	Top Features
2005	4 2 5 1
2006	4 3 2 1
2007	4 3 5 2

Table 2: The 4 top features for the DUC 2005, 2006 and 2007 data

Table 2 shows that feature 4, document frequency, is consistently the most important feature for all three data sets. Content word frequency (3), on the other hand, comes in as second best feature for 2006 and 2007, but not for 2005. For the 2005 data, the Topic description frequency is the second best feature. This observation is reflected by our single fea-

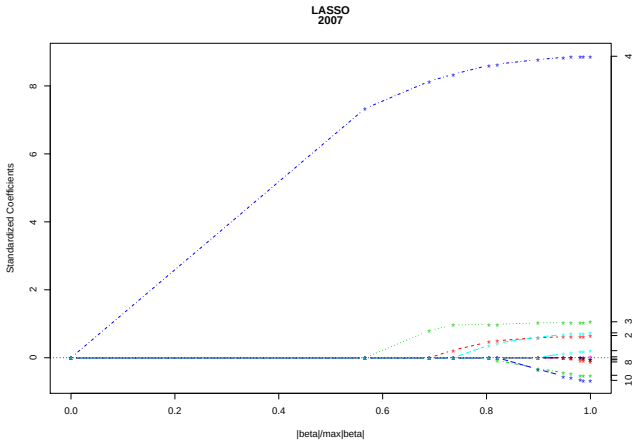


Figure 2: Graphical output of LARS analysis:
Top features for 2007: 4 Document frequency, 3 Content word frequency, 5 Headline frequency, 2 Topic description frequency

ture analysis for DUC 2006, as shown in table 1. Similarly, Vanderwende et al. (2006) report that they gave the Topic description frequency a much higher weight than the Content word frequency.

Consequently, we have shown that our new feature Document frequency is consistently the best feature for all three past DUC corpora.

4 Conclusions

We proposed a fast query-based multi-document summarizer called FastSum that produces state-of-the-art summaries using a small set of predictors, two of those are proposed by us: document frequency and topic title frequency. A feature analysis using least angle regression (LARS) indicated that the document frequency feature is the most useful feature consistently for the last three DUC data sets. Using document frequency alone can produce competitive results for DUC 2006 and DUC 2007. The two most useful feature that takes the topic description (i.e., the queries) into account is based on the number of words in the topic description and the topic title. Using a limited feature set of the 5 best features generates summaries that are comparable to the top systems of the DUC 2006 and 2007 main task and can be generated in real-time, since no computationally expensive features (e.g., parsing) are used.

From these findings, we draw the following conclusions. Since a feature set mainly based on word frequencies can produce state-of-the-art summaries, we need to analyze further the current set-up for the query-based multi-document summarization task. In particular, we need to ask the question whether the selection of relevant documents for the DUC topics is in any way biased. For DUC, the document clusters for a topic containing relevant documents were always pre-selected by the assessors in preparation for DUC. Our analysis suggests that simple word frequency computations of these clusters and the documents alone can produce reasonable summaries. However, the human selecting the relevant documents may have already influenced the way summaries can automatically be generated. Our system and systems such as SumBasic or SumFocus may just exploit the fact that relevant articles prescreened by humans contain a high density of good content words for summarization.[6]

References

D. Bollegala, Y. Matsuo, and M. Ishizuka. 2007. Measuring Semantic Similarity between Words Using Web Search Engines. In *Proc. of 16th International World Wide Web Conference (WWW 2007)*, pages 757–766, Banff, Canada.

B. Efron, T. Hastie, I.M. Johnstone, and R. Tibshirani. 2004. Least angle regression. *Annals of Statistics*, 32(2):407–499.

S. Gupta, A. Nenkova, and D. Jurafsky. 2007. Measuring Importance and Query Relevance in Topic-focused Multi-document Summarization. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 193–196, Prague, Czech Republic.

S. Li, Y. Ouyang, W. Wang, and B. Sun. 2007. Multi-document summarization using support vector regression. In *Proceedings of DUC 2007, Rochester, USA*.

C. Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*.

A. Nenkova and L. Vanderwende. 2005. The impact of frequency on summarization. In *MSR-TR-2005-101*.

R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288.

K. Toutonova, C. Brockett, J. Jagarlamudi, H. Suzuko, and L. Vanderwende. 2007. The PYTHY Summarization System: Microsoft Research at DUC2007. In *Proc. of DUC 2007*, Rochester, USA.

L. Vanderwende, H. Suzuki, and C. Brockett. 2006. Microsoft Research at DUC 2006: Task-focused summarization with sentence simplification and lexical expansion. In *Proc. of DUC 2006, New York, USA*.

[6]Cf. Gupta et al. (2007) who come to a similar conclusion by comparing between word frequency and log-likelihood ratio.

Construct State Modification in the Arabic Treebank

Ryan Gabbard
Department of Computer and Information Science
University of Pennsylvania
gabbard@seas.upenn.edu

Seth Kulick
Linguistic Data Consortium
Institute for Research in Cognitive Science
University of Pennsylvania
skulick@seas.upenn.edu

Abstract

Earlier work in parsing Arabic has speculated that attachment to construct state constructions decreases parsing performance. We make this speculation precise and define the problem of attachment to construct state constructions in the Arabic Treebank. We present the first statistics that quantify the problem. We provide a baseline and the results from a first attempt at a discriminative learning procedure for this task, achieving 80% accuracy.

1 Introduction

Earlier work on parsing the Arabic Treebank (Kulick et al., 2006) noted that prepositional phrase attachment was significantly worse on the Arabic Treebank (ATB) than the English Penn Treebank (PTB) and speculated that this was due to the ubiquitous presence of construct state NPs in the ATB. Construct state NPs, also known as iDAfa[1] (إِضَافَة) constructions, are those in which (roughly) two or more words, usually nouns, are grouped tightly together, often corresponding to what in English would be expressed with a noun-noun compound or a possessive construction (Ryding, 2005)[pp.205–227]. In the ATB these constructions are annotated as a NP headed by a NOUN with an NP complement. (Kulick et al., 2006) noted that this created very different contexts for PP attachment to "base NPs", likely leading to the lower results for PP attachment.

In this paper we make their speculation precise and define the problem of attachment to construct state constructions in the ATB by extracting out such iDAfa constructions[2] and their modifiers. We provide the first statistics we are aware of that quantify the number and complexity of iDAfas in the ATB and the variety of modifier attachments within them. Additionally, we provide the first baseline for this problem as well as preliminary results from a discriminative learning procedure for the task.

2 The Problem in More Detail

As mentioned above, iDAfa constructions in the ATB are annotated as a NOUN with an NP complement (ATB, 2008). This can also be recursive, in that the NP complement can itself be an iDAfa construction. For example, Figure 1 shows such a complex iDAfa. We refer to an iDAfa of the form `(NP NOUN (NP NOUN))` as a two–level iDAfa, one of the form `(NP NOUN (NP NOUN (NP NOUN)))` as a three–level iDAfa (as in Figure 1), and so on. Modification can take place at any of the NPs in these iDAfas, using the usual adjunction structure, as in Figure 2 (in which the modifier itself contains an iDAfa as the object of the PREP `fiy`).[3]

This annotation of the iDAfa construction has a crucial impact upon the usual problem of PP attachment. Consider first the PP attachment problem for the PTB. The PTB annotation style (Bies et

[1] Throughout this paper we use the Buckwalter Arabic transliteration scheme (Buckwalter, 2004).

[2] Throughout the rest of this paper, we will refer for convenience to iDAfa constructions instead of "construct state NPs".

[3] In all these tree examples we leave out the Part of Speech tags to lessen clutter, and likewise for the nonterminal function tags.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 209–212,
Columbus, Ohio, USA, June 2008.

```
(NP $awAriE
    [streets]
    (NP madiyn+ap
       [city]
        (NP luwnog  byt$)))
            [Long] [Beach]
```
شَوَارِع مَدِينَة لونغ بيتش

Figure 1: A three level idafa, meaning *the streets of the city of Long Beach*

```
(NP $awAriE
    [streets]
    (NP (NP madiyn+ap
            [city]
            (NP luwnog  byt$))
                [Long] [Beach]
        (PP fiy
            [in]
            (NP wilAy+ap
                [state]
                (NP kAliyfuwrniyA)))))
                    [California]
```
شَوَارِع مَدِينَة لونغ بيتش فِي وِلَايَة كَالِفورنِيَا

Figure 2: Three level iDAfa with modification, meaning *the streets of the city of Long Beach in the state of California*

al., 1995) forces multiple PP modifiers of the same NP to be at the same level, disallowing the structure (B) in favor of structure (A) in Figure 3, and parsers can take advantage of this restriction. For example, (Collins, 1999)[pp. 211-12] uses the notion of a "base NP" (roughly, a non–recursive NP, that is, one without an internal NP) to control PP attachment, so that the parser will not mistakenly generate the (B) structure, since it learns that PPs attach to non–recursive, but not recursive, NPs.

Now consider again the PP attachment problem in the ATB. The ATB guidelines also enforce the restriction in Figure 3, so that multiple modifiers of an NP within an iDAfa will be at the same level (e.g., another PP modifier of "the city of Long Beach" in Figure 2 would be at the same level as "in the state..."). However, the iDAfa annotation, independently of this annotation constraint, results in the PP modification of many NPs that are not base NPs, as with the PP modifier "in the state..." in Figure 2. One way to view what is happening here is that Arabic uses the iDAfa construction to express what is often

(A) multi-level PP attachment at same level — allowed

```
(NP (NP ...)
    (PP ....)
    (PP ....))
```

(B) multi-level PP attachment at different levels — not allowed

```
(NP (NP (NP ...)
        (PP ....)
    (PP ....))
```

Figure 3: Multiple PP attachment in the PTB

```
(NP (NP streets)
    (PP of
        (NP (NP the city)
            (PP of (NP Long Beach))
            (PP in (NP the state)
                   (PP of
                     (NP California))))))
```

Figure 4: The English analog of Figure 2

a PP in English. The PTB analog of the troublesome iDAfa with PP attachment in Figure 2 would be the simpler structure in Figure 4, with two PP attachments to the base NP "the city." The PP modifier "of Long Beach" in English becomes part of iDAfa construction in Arabic.

In addition, PPs can modify any level in an iDAfa construction, so there can be modification within an iDAfa of either a recursive or base NP. There can also be modifiers of multiple terms in an iDAfa.[4]

The upshot is that the PP modification is more free in the ATB than in the PTB, and base NPs are no longer adequate to control PP attachment. (Kulick et al., 2006) present data showing that PP attachment to a non–recursive NP is virtually non–existent in the PTB, while it is the 16th most frequent dependency in the ATB, and that the performance of the parser they worked with (the Bikel implementation (Bikel, 2004) of the Collins parser) was significantly lower on PP attachment for the ATB than for PTB.

The data we used was the recently completed revision of 100K words from the ATB3 ANNAHAR corpus (Maamouri et al., 2007). We extracted all oc-

[4] An iDAfa cannot be interrupted by modifiers for non-final terms, meaning that multiple modifiers will be grouped together following the iDAfa. Also, a single adjective can modify a noun within the lowest NP, i.e., inside the base NP.

Number of Modifiers	Percent of iDAfas
1	72.4
2	20.6
3	5.2
4	1.0
5	0.2
8	0.6

Table 1: Number of modifiers per iDAfa

Depth	Percent of Idafas
2	75.5
3	19.9
4	3.8
5	0.8
6	0.1

Table 2: Distribution of depths of iDAfas

currences of NP constituents with a NOUN or NOUN–like head (NOUN_PROP, NUM, NOUN_QUANT) and a NP complement.

This extraction results in 9472 iDAfas of which 3877 of which have modifiers. The average number of idafas per sentence is 3.06.

3 Some Results

In the usual manner, we divided the data into training, development test, and test sections according to an 80/10/10 division. As the work in this paper is preliminary, the test section is not used and all results are from the dev–test section.

By extracting counts from the training section, we obtained some information about the behavior of iDAfas. In Table 1 we see that of iDAfas which have at least one modifier, most (72%) have only one modifier, and a sizable number (21%) have two, while a handful have as many as eight. Almost all iDAfas are of depth three or less (Table 2), with the deepest depth in our training set being six.

Finally, we observe that the distributions of attachment depths of modifiers differs significantly for different depths of iDAfas (table 3). All depths have somewhat of a preference for attachment at the bottom (43% for depth two and 36% for depths three and four), but the top is a much more popular attachment site for depth two idafas (39%) than it is for deeper ones. Level one attachments are very common for depth three iDAfas for reasons which are unclear.

	Depth 2	Depth 3	Depth 4
Level 0	39.0	19.3	16.1
Level 1	17.9	34.8	14.1
Level 2	43.0	9.9	23.6
Level 3		36.0	10.1
Level 4			36.2

Table 3: For each total iDAfa depth, the percentage of attachments at each level. iDAfa depths of five or above are omitted due to the small number of such cases.

Based on these observations, we would expect a simple baseline which attaches at the most common depth to do quite poorly. We confirm this by building a statistical model for iDAfa attachment, which we then use for exploring some features which might be useful for the task, either as a separate post–processing step or within a parser.

To simplify the learning task, we make the independence assumption that all modifier attachments to an iDAfa are independent of one another subject to the constraint that later attachments may not attach deeper than earlier ones. We then model the probabilities of each of these attachments with a maximum entropy model and use a straightforward dynamic programming search to find the most probable assignment to all the attachments together. Formally, we assign each attachment a numerical depth (0 for top, 1 for the position below the top, and so on) and then we find

$$\underset{a_1,\ldots,a_n}{\operatorname{argmax}} \prod_{1}^{n} P(a_1)\ldots P(a_n) \text{ s.t. } \forall x : a_x <= a_{x-1}$$

Our baseline system uses only the depth of the attachment as a feature. We built further systems which used the following bundles of features:

AttSym Adds the part–of–speech tag or non–terminal symbol of the modifier.

Lex Pairs the headword of the modifier with the noun it is modifying.

TotDepth Conjunction of the attachment location, the AttSym feature, and the total depth of the iDAfa.

Features	Accuracy
Base	39.7
Base+AttSym	76.1
Base+Lex	58.4
Base+Lex+AttSym	**79.9**
Base+Lex+AttSym+TotDepth	78.7
Base+Lex+AttSym+GenAgr	79.3

Table 4: Attachment accuracy on development test data for our model trained on various feature bundles.

GenAgr A "full" gender feature consisting of the AttSym feature conjoined with the the pair of the gender and number suffixes of the head of the modifier and the word being modified and a "simple" gender feature which is the same except it omits number.

Results are in table 4. Our most useful feature is clearly AttSym, with Lex also providing significant information. Combining them allows us to achieve 80% accuracy. However, attempts to improve on this by using gender agreement or taking advantage of the differing attachment distributions for different iDAfa depths (3) were ineffective. In the case of gender agreement, it may be ineffective because non–human plurals have feminine singular gender agreement, but there is no annotation for humanness in the ATB.

4 Conclusion

We have presented an initial exploration of the iDAfa attachment problem in Arabic and have presented the first data on iDAfa attachment distributions. We have also demonstrated that a combination of lexical information and the top symbols of modifiers can achieve 80% accuracy on the task.

There is much room for further work here. It is possible a more sophisticated statistical model which eliminates the assumption that modifier attachments are independent of each other and which does global rather than local normalization would be more effective. We also plan to look into adding more features or enhancing existing features (e.g. try to get more effective gender agreement by approximating annotation for humanness). Some constructions, such as the false iDAfa, require more investigation, and we can also expand the range of investigation to include coordination within an iDAfa.

The more general plan is to incorporate this work within a larger Arabic NLP system. This could perhaps be as a phase following a base phrase chunker (Diab, 2007), or after a parser, either correcting or completing the parser output.

Acknowledgments

We thank Mitch Marcus, Ann Bies, Mohamed Maamouri, and the members of the Arabic Treebank project for helpful discussions. This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract Nos. HR0011-06-C-0022 and HR0011-06-1-0003. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

ATB. 2008. Arabic Treebank Morphological and Syntactic guidelines. http://projects.ldc.upenn.edu/ArabicTreebank.

Ann Bies, Mark Ferguson, Karen Karz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II-style Penn Treebank project. Technical report, University of Pennsylvania.

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4).

Tim Buckwalter. 2004. Arabic morphological analyzer version 2.0. LDC2004L02. Linguistic Data Consortium.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Department of Computer and Information Sciences, University of Pennsylvania.

Mona Diab. 2007. Improved Arabic base phrase chunking with a new enriched pos tag set. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages*.

Seth Kulick, Ryan Gabbard, and Mitchell Marcus. 2006. Parsing the Arabic Treebank: Analysis and improvements. In *Proceedings of TLT 2006*. Treebanks and Linguistic Theories.

Mohamed Maamouri, Ann Bies, Seth Kulick, Fatma Gadeche, and Wigdan Mekki. 2007. Arabic treebank 3(a) - v2.6. LDC2007E65. Linguistic Data Consortium.

Karin C. Ryding. 2005. *A Reference Grammar of Modern Standard Arabic*. Cambridge University Press.

Unlexicalised Hidden Variable Models of Split Dependency Grammars*

Gabriele Antonio Musillo
Department of Computer Science
and Department of Linguistics
University of Geneva
1211 Geneva 4, Switzerland
musillo4@etu.unige.ch

Paola Merlo
Department of Linguistics
University of Geneva
1211 Geneva 4, Switzerland
merlo@lettres.unige.ch

Abstract

This paper investigates transforms of split dependency grammars into unlexicalised context-free grammars annotated with hidden symbols. Our best unlexicalised grammar achieves an accuracy of 88% on the Penn Treebank data set, that represents a 50% reduction in error over previously published results on unlexicalised dependency parsing.

1 Introduction

Recent research in natural language parsing has extensively investigated probabilistic models of phrase-structure parse trees. As well as being the most commonly used probabilistic models of parse trees, probabilistic context-free grammars (PCFGs) are the best understood. As shown in (Klein and Manning, 2003), the ability of PCFG models to disambiguate phrases crucially depends on the expressiveness of the symbolic backbone they use.

Treebank-specific heuristics have commonly been used both to alleviate inadequate independence assumptions stipulated by naive PCFGs (Collins, 1999; Charniak, 2000). Such methods stand in sharp contrast to partially supervised techniques that have recently been proposed to induce hidden grammatical representations that are finer-grained than those that can be read off the parsed sentences in treebanks (Henderson, 2003; Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006).

*Part of this work was done when Gabriele Musillo was visiting the MIT Computer Science and Artificial Intelligence Laboratory, funded by a grant from the Swiss NSF (PBGE2-117146). Many thanks to Michael Collins and Xavier Carreras for their insightful comments on the work presented here.

This paper presents extensions of such grammar induction techniques to dependency grammars. Our extensions rely on transformations of dependency grammars into efficiently parsable context-free grammars (CFG) annotated with hidden symbols. Because dependency grammars are reduced to CFGs, any learning algorithm developed for PCFGs can be applied to them. Specifically, we use the Inside-Outside algorithm defined in (Pereira and Schabes, 1992) to learn transformed dependency grammars annotated with hidden symbols. What distinguishes our work from most previous work on dependency parsing is that our models are not lexicalised. Our models are instead decorated with hidden symbols that are designed to capture both lexical and structural information relevant to accurate dependency parsing without having to rely on any explicit supervision.

2 Transforms of Dependency Grammars

Contrary to phrase-structure grammars that stipulate the existence of phrasal nodes, dependency grammars assume that syntactic structures are connected acyclic graphs consisting of vertices representing terminal tokens related by directed edges representing dependency relations. Such terminal symbols are most commonly assumed to be words. In our unlexicalised models reported below, they are instead assumed to be part-of-speech (PoS) tags. A typical dependency graph is illustrated in Figure 1 below.

Various projective dependency grammars exemplify the concept of split bilexical dependency grammar (SBG) defined in (Eisner, 2000). [1] SBGs are

[1] An SBG is a tuple $\langle V, W, L, R \rangle$ such that:

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 213–216,
Columbus, Ohio, USA, June 2008.

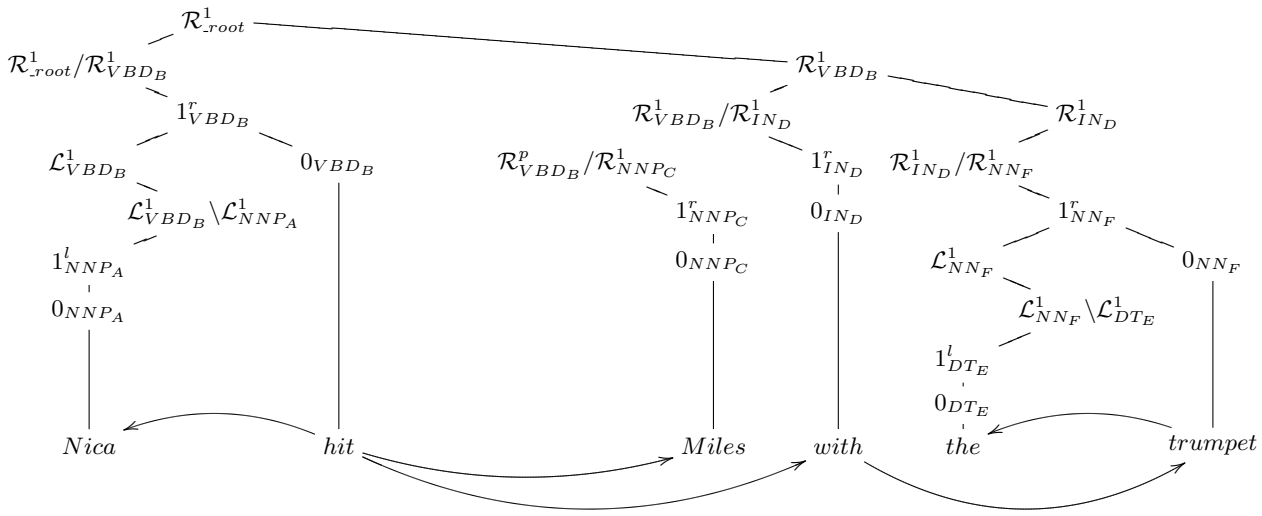


Figure 1: A projective dependency graph for the sentence *Nica hit Miles with the trumpet* paired with its second-order unlexicalised derivation tree annotated with hidden variables.

closely related to CFGs as they both define structures that are rooted ordered projective trees. Such a close relationship is clarified in this section.

It follows from the equivalence of finite automata and regular grammars that any SBG can be transformed into an equivalent CFG. Let $D = \langle V, W, L, R\rangle$ be a SBG and $G = \langle N, W, P, S\rangle$ a CFG. To transform D into G we to define the set P of productions, the set N of non-terminals, and the start symbol S as follows:

- For each v in W, transform the automaton L_v into a right-linear grammar G_{L_v} whose start symbol is $\mathcal{L}^1_v$; by construction, G_{L_v} consists of rules such as $\mathcal{L}^p_v \rightarrow u\ \mathcal{L}^q_v$ or $\mathcal{L}^p_v \rightarrow \epsilon$, where terminal symbols such as u belong to W and non-terminals such as $\mathcal{L}^p_v$ correspond to the states of the L_v automaton; include all ϵ-productions in P, and, if a rule such as $\mathcal{L}^p_v \rightarrow u\ \mathcal{L}^q_v$ is in G_{L_v}, include the rule $\mathcal{L}^p_v \rightarrow 2^l_u\ \mathcal{L}^q_v$ in P.
- For each v in V, transform the automaton R_v into a left-linear grammar G_{R_v} whose start symbol is $\mathcal{R}^1_v$; by construction, G_{R_v} consists of rules such as $\mathcal{R}^p_v \rightarrow \mathcal{R}^q_v\ u$ or $\mathcal{R}^p_v \rightarrow \epsilon$, where terminal symbols such as u belongs to W and non-terminals such as $\mathcal{R}^p_v$ correspond to the states of the R_v automaton; include all ϵ-productions in P, and, if a rule such as $\mathcal{R}^p_v \rightarrow \mathcal{R}^q_v\ u$ is in G_{R_v}, include the rule $\mathcal{R}^p_v \rightarrow \mathcal{R}^q_v\ 2^r_u$ in P.
- For each symbol 2^l_u occurring in P, include the productions $2^l_u \rightarrow\ \mathcal{L}^1_u\ 1^l_u$, $1^l_u \rightarrow\ 0_u\ \mathcal{R}^1_u$, and $0_u \rightarrow\ u$ in P; for each symbol 2^r_u in P, include the productions $2^r_u \rightarrow\ 1^r_u\ \mathcal{R}^1_u$, $1^r_u \rightarrow\ \mathcal{L}^1_u\ 0_u$, and $0_u \rightarrow\ u$ in P.
- Set the start symbol S to $\mathcal{R}^1_{\_root}$. [2]

Parsing CFGs resulting from such transforms runs in $O(n^4)$. The head index v decorating non-terminals such as 1^l_v, 1^r_v, 0_v, $\mathcal{L}^p_v$ and $\mathcal{R}^q_v$ can be computed in $O(1)$ given the left and right indices of the sub-string $w_{i,j}$ they cover. [3] Observe, however, that if 2^l_v or 2^r_v derives $w_{i,j}$, then v does not functionally depend on either i or j. Because it is possible for the head index v of 2^l_v or 2^r_v to vary from i to j, v has to be tracked by the parser, resulting in an overall $O(n^4)$ time complexity.

In the following, we show how to transform our $O(n^4)$ CFGs into $O(n^3)$ grammars by ap-

- V is a set of terminal symbols which include a distinguished element $\_root$;
- L is a function that, for any $v \in W(= V - \{\_root\})$, returns a finite automaton that recognises the well-formed sequences in W^* of left dependents of v;
- R is a function that, for each $v \in V$, returns a finite automaton that recognises the well-formed sequences of right dependents in W^* for v.

[2] CFGs resulting from such transformations can further be normalised by removing the ϵ-productions from P.

[3] Indeed, if 1^l_v or 0_v derives $w_{i,j}$, then $v = i$; if 1^r_v derives $w_{i,j}$, then $v = j$; if $w_{i,j}$ is derived from $\mathcal{L}^p_v$, then $v = j + 1$; and if $w_{i,j}$ is derived from $\mathcal{R}^q_v$, then $v = i - 1$.

plying transformations, closely related to those in (McAllester, 1999) and (Johnson, 2007), that eliminate the 2_v^l and 2_v^r symbols.

We only detail the elimination of the symbols 2_v^r. The elimination of the 2_v^l symbols can be derived symmetrically. By construction, a 2_v^r symbol is the right successor of a non-terminal $\mathcal{R}_u^p$. Consequently, 2_v^r can only occur in a derivation such as

$$\alpha\ \mathcal{R}_u^p\ \beta\ \vdash\ \alpha\ \mathcal{R}_u^q\ 2_v^r\ \beta\ \vdash\ \alpha\ \mathcal{R}_u^q\ 1_v^r\ \mathcal{R}_v^1\ \beta.$$

To substitute for the problematic 2_v^r non-terminal in the above derivation, we derive the form $\mathcal{R}_u^q\ 1_v^r\ \mathcal{R}_v^1$ from $\mathcal{R}_u^p/\mathcal{R}_v^1\ \ \mathcal{R}_v^1$ where $\mathcal{R}_u^p/\mathcal{R}_v^1$ is a new non-terminal whose right-hand side is $\mathcal{R}_u^q\ 1_v^r$. We thus transform the above derivation into the derivation $\alpha\ \mathcal{R}_u^p\ \beta\ \vdash\ \alpha\ \mathcal{R}_u^p/\mathcal{R}_v^1\ \mathcal{R}_v^1\beta\ \vdash\ \alpha\ \mathcal{R}_u^q\ 1_v^r\ \mathcal{R}_v^1\ \beta$. [4]

Because $u = i - 1$ and $v = j$ if $\mathcal{R}_u^p/\mathcal{R}_v^1$ derives $w_{i,j}$, and $u = j + 1$ and $v = i$ if $\mathcal{L}_u^p\backslash\mathcal{L}_v^1$ derives $w_{i,j}$, the parsing algorithm does not have to track any head indices and can consequently parse strings in $O(n^3)$ time.

The grammars described above can be further transformed to capture linear second-order dependencies involving three distinct head indices. A second-order dependency structure is illustrated in Figure 1 that involves two adjacent dependents, *Miles* and *with*, of a single head, *hit*.

To see how linear second-order dependencies can be captured, consider the following derivation of a sequence of right dependents of a head u:

$$\alpha\ \mathcal{R}_u^p/\mathcal{R}_v^1\ \beta \vdash\ \alpha\ \mathcal{R}_u^q\ 1_v^r\ \beta\ \vdash\ \alpha\ \mathcal{R}_u^q/\mathcal{R}_w^1\ \mathcal{R}_w^1\ \ 1_v^r\ \beta.$$

The form $\mathcal{R}_u^q/\mathcal{R}_w^1\ \mathcal{R}_w^1\ \ 1_v$ mentions three heads: u is the the head that governs both v and w, and w precedes v. To encode the linear relationship between w and v, we redefine the right-hand side of $\mathcal{R}_u^p/\mathcal{R}_v^1$ as $\mathcal{R}_u^q/\mathcal{R}_w^1\ \langle\mathcal{R}_w^1,\ 1_v^r\rangle$ and include the production $\langle\mathcal{R}_w^1,\ 1_v^r\rangle\ \rightarrow\ \ \mathcal{R}_w^1\ 1_v^r$ in the productions. The relationship between the dependents w and v of the head u is captured, because $\mathcal{R}_u^p/\mathcal{R}_v^1$ jointly generates $\mathcal{R}_w^1$ and 1_v^r. [5]

Any second-order grammar resulting from transforming the derivations of right and left dependents in the way described above can be parsed in $O(n^3)$, because the head indices decorating its symbols can be computed in $O(1)$.

In the following section, we show how to enrich both our first-order and second-order grammars with hidden variables.

3 Hidden Variable Models

Because they do not stipulate the existence of phrasal nodes, commonly used unlabelled dependency models are not sufficiently expressive to discriminate between distinct projections of a given head. Both our first-order and second-order grammars conflate distributionally distinct projections if they are projected from the same head. [6]

To capture various distinct projections of a head, we annotate each of the symbols that refers to it with a unique hidden variable. We thus constrain the distribution of the possible values of the hidden variables in a linguistically meaningful way. Figure 1 illustrates such constraints: the same hidden variable $_B$ decorates each occurrence of the PoS tag `VBD` of the head *hit*.

Enforcing such agreement constraints between hidden variables provides a principled way to capture not only phrasal information but also lexical information. Lexical pieces of information conveyed by a minimal projection such as 0_{VBD_B} in Figure 1 will consistently be propagated through the derivation tree and will condition the generation of the right and left dependents of *hit*.

In addition, states such as p and q that decorate non-terminal symbols such as $\mathcal{R}_u^p$ or $\mathcal{L}_u^q$ can also capture structural information, because they can encode the most recent steps in the derivation history. In the models reported in the next section, these states are assumed to be hidden and a distribution over their possible values is automatically induced.

4 Empirical Work and Discussion

The models reported below were trained, validated, and tested on the commonly used sections from the Penn Treebank. Projective dependency trees, ob-

[4] Symmetrically, the derivation $\alpha\ \mathcal{L}_u^p\ \beta\ \vdash\ \alpha\ \ 2_v^l\ \mathcal{L}_u^q\ \beta\ \vdash\ \alpha\ \mathcal{L}_v^1\ 1_v^l\ \mathcal{L}_u^q\ \beta$ involving the 2_v^l symbol is transformed into $\alpha\ \mathcal{L}_u^p\ \beta\ \vdash\ \alpha\ \mathcal{L}_v^1\ \ \mathcal{L}_u^p\backslash\mathcal{L}_v^1\ \beta\ \vdash\ \alpha\ \mathcal{L}_v^1\ 1_v^l\ \mathcal{L}_u^q\ \beta$.

[5] Symmetrically, to transform the derivation of a sequence of left dependents of u, we redefine the right-hand side of $\mathcal{L}_u^p\backslash\mathcal{L}_v^1$ as $\langle 1_v^l, \mathcal{L}_w^1\rangle\ \mathcal{L}_u^q\backslash\mathcal{L}_w^1$ and include the production $\langle 1_v^l, \mathcal{L}_w^1\rangle\ \rightarrow\ 1_v^l\ \mathcal{L}_w^1$ in the set of rules.

[6] As observed in (Collins, 1999), an unambiguous verbal head such as *prove* bearing the `VB` tag may project a clause with an overt subject as well as a clause without an overt subject, but only the latter is a possible dependent of subject control verbs such as *try*.

Development Data – section 24	*per word*	*per sentence*
FOM: $q = 1, h = 1$	75.7	9.9
SOM: $q = 1, h = 1$	80.5	16.2
FOM: $q = 2, h = 2$	81.9	17.4
FOM: $q = 2, h = 4$	84.7	22.0
SOM: $q = 2, h = 2$	84.3	21.5
SOM: $q = 1, h = 4$	87.0	25.8
Test Data – section 23	*per word*	*per sentence*
(Eisner and Smith, 2005)	75.6	NA
SOM: $q = 1, h = 4$	88.0	30.6
(McDonald, 2006)	91.5	36.7

Table 1: Accuracy results on the development and test data set, where q denotes the number of hidden states and h the number of hidden values annotating a PoS tag involved in our first-order (FOM) and second-order (SOM) models.

tained using the rules stated in (Yamada and Matsumoto, 2003), were transformed into first-order and second-order structures. CFGs extracted from such structures were then annotated with hidden variables encoding the constraints described in the previous section and trained until convergence by means of the Inside-Outside algorithm defined in (Pereira and Schabes, 1992) and applied in (Matsuzaki et al., 2005). To efficiently decode our hidden variable models, we pruned the search space as in (Petrov et al., 2006). To evaluate the performance of our models, we report two of the standard measures: the *per word* and *per sentence* accuracy (McDonald, 2006).

Figures reported in the upper section of Table 1 measure the effect on accuracy of the transforms we designed. Our baseline first-order model ($q = 1, h = 1$) reaches a poor per word accuracy that suggests that information conveyed by bare PoS tags is not fine-grained enough to accurately predict dependencies. Results reported in the second line shows that modelling adjacency relations between dependents as second-order models do is relevant to accuracy. The third line indicates that annotating both the states and the PoS tags of a first-order model with two hidden values is sufficient to reach a performance comparable to the one achieved by a naive second-order model. However, comparing the results obtained by our best first-order models to the accuracy achieved by our best second-order model conclusively shows that first-order models exploit such dependencies to a much lesser extent. Overall, such results provide a first solution to the problem left open in (Johnson, 2007) as to whether second-order transforms are relevant to parsing accuracy or not.

The lower section of Table 1 reports the results achieved by our best model on the test data set and compare them both to those obtained by the only unlexicalised dependency model we know of (Eisner and Smith, 2005) and to those achieved by the state-of-the-art dependency parser in (McDonald, 2006). While clearly not state-of-the-art, the performance achieved by our best model suggests that massive lexicalisation of dependency models might not be necessary to achieve competitive performance. Future work will lie in investigating the issue of lexicalisation in the context of dependency parsing by weakly lexicalising our hidden variable models.

References

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL'00*.

Michael John Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT'05*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H.Bunt and A. Nijholt, eds., *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Jamie Henderson. 2003. Inducing history representations for broad-coverage statistical parsing. In *NAACL-HLT'03*.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. In *ACL'06*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL'03*.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL'05*.

David McAllester. 1999. A reformulation of eisner and satta's cubit time parser for split head automata grammars. http://ttic.uchicago.edu/dmcallester.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation form partially bracketed corpora. In *ACL'92*.

Slav Petrov, Leon Barrett Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL'06*.

Detlef Prescher. 2005. Head-driven PCFGs with latent-head statistics. In *IWPT'05*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vectore machines. In *IWPT'03*.

Computing Confidence Scores for All Sub Parse Trees

Feng Lin

Department of Computer Science and Engineering
Fudan University
Shanghai 200433, P.R. China
fenglin@fudan.edu.cn

Fuliang Weng

Research and Technology Center
Robert Bosch LLC
Palo Alto, CA, 94303, USA
fuliang.weng@us.bosch.com

Abstract

Computing confidence scores for applications, such as dialogue system, information retrieving and extraction, is an active research area. However, its focus has been primarily on computing word-, concept-, or utterance-level confidences. Motivated by the need from sophisticated dialogue systems for more effective dialogs, we generalize the confidence annotation to all the subtrees, the first effort in this line of research. The other contribution of this work is that we incorporated novel long distance features to address challenges in computing multi-level confidence scores. Using Conditional Maximum Entropy (CME) classifier with all the selected features, we reached an annotation error rate of 26.0% in the SWBD corpus, compared with a subtree error rate of 41.91%, a closely related benchmark with the Charniak parser from (Kahn et al., 2005).

1 Introduction

There has been a good amount of interest in obtaining confidence scores for improving word or utterance accuracy, dialogue systems, information retrieving & extraction, and machine translation (Zhang and Rudnicky, 2001; Guillevic et al., 2002; Gabsdil et al., 2003; Ueffing et al., 2007).

However, these confidence scores are limited to relatively simple systems, such as command-n-control dialogue systems. For more sophisticated dialogue systems (e.g., Weng et al., 2007), identification of reliable phrases must be performed at different granularity to ensure effective and friendly dialogues. For example, in a request of MP3 music domain "Play a rock song by Cher", if we want to communicate to the user that the system is not confident of the phrase "a rock song," the confidence scores for each word, the artist name "Cher," and the whole sentence would not be enough. For tasks of information extraction, when extracted content has internal structures, confidence scores for such phrases are very useful for reliable returns.

As a first attempt in this research, we generalize confidence annotation algorithms to all sub parse trees and tested on a human-human conversational corpus, the SWBD. Technically, we also introduce a set of long distance features to address the challenges in computing multi-level confidence scores.

This paper is organized as follows: Section 2 introduces the tasks and the representation for parse trees; Section 3 presents the features used in the algorithm; Section 4 describes the experiments in the SWBD corpus; Section 5 concludes the paper.

2 Computing Confidence Scores for Parse Trees

The confidence of a sub-tree is defined as the posterior probability of its correctness, given all the available information. It is $P(sp\,is\,correct|\,x)$ – the posterior probability that the parse sub-tree *sp* is correct, given related information *x*. In real applications, typically a threshold or cutoff *t* is needed:

$$sp\ is \begin{cases} correct,\ if\ P(sp\,is\,correct|\,x) \geq t \\ incorrec,\ if\ \ P(sp\,is\,correct|\,x) < t \end{cases} \quad (1)$$

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 217–220,
Columbus, Ohio, USA, June 2008.

In this work, the probability $P(sp\,is\,correct|\,x)$ is calculated using CME modeling framework:

$$P(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_j \lambda_j f_j(x, y) \right) \quad (2)$$

where $y \in \{sp\ is\ correct,\ sp\ is\ incorrect\}$, x is the syntactic context of the parse sub-tree sp, f_j are the features, λ_j are the corresponding weights, and $Z(x)$ is the normalization factor.

The parse trees used in our system are lexicalized binary trees. However, the confidence computation is independent of any parsing method used in generating the parse tree as long as it generates the binary dependency relations. An example of the lexicalized binary trees is given in Figure 1, where three important components are illustrated: the left sub-tree, the right sub-trees, and the marked head and dependency relation.

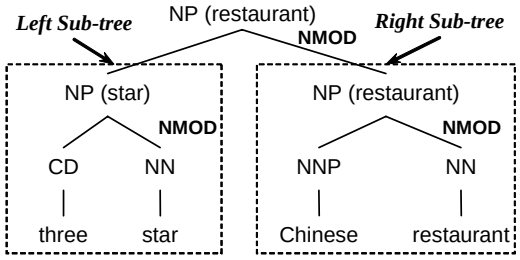


Figure 1. Example of parse sub-tree's structure for phrase "three star Chinese restaurant"

Because the parse tree is already given, a bottom-up left-right algorithm is used to traverse through the parse tree: for each subtree, compute its confidence, and annotate it as correct or wrong.

3 Features

Four major categories of features are used, including, words, POS tags, scores and syntactic information. Due to the space limitation, we only give a detailed description of the most important one[1], lexical-syntactic features.

The lexical-syntactic features include lexical, POS tag, and syntactic features. Word and POS tag features include the head and modifier words of the parse sub-tree and the two children of the root, as well as their combinations. The POS tags and hierarchical POS tags of the corresponding words are also considered to avoid data sparseness. The adopted hierarchical tags are: Verb-related (V), Noun-related (N), Adjectives (ADJ), and Adverbs (ADV), similar to (Zhang et al, 2006).

Long distance structural features in statistical parsing lead to significant improvements (Collins et al., 2000; Charniak et al., 2005). We incorporate some of the reported features in the feature space to be explored, and they are enriched with different POS categories and grammatical types. Two examples are given below.

One example is the Single-Level Joint Head and Dependency Relation (SL-JHD). This feature is pairing the head word of a given sub-tree with its last dependency relation. To address the data sparseness problem, two additional SL-JHD features are considered: a pair of the POS tag of the head of a given sub-tree and its dependency relation, a pair of the hierarchical POS tag of the head of a given sub-tree and its dependency relation. For example, for the top node in Figure 2, (restaurant NCOMP), (NN, NCOMP), and (N, NCOMP) are the examples for the three SL-JHD features. To compute the confidence score of the sub-tree, we include the three JHD features for the top node, and the JHD features for its two children. Thus, for the sub-tree in Figure 2, the following nine JHD features are included in the feature space, i.e., (restaurant NCOMP), (NN, NCOMP), (N, NCOMP), (restaurant NMOD), (NN NMOD), (N NMOD), (with POBJ), (IN POBJ), and (ADV POBJ).

The other example feature is Multi-Level Joint Head and Dependency Relation (ML-JHD), which takes into consideration the dependency relations at multiple levels. This feature is an extension of SL-JHD. Instead of including only single level head and dependency relations, the ML-JHD feature includes the hierarchical POS tag of the head and dependency relations for all the levels of a given sub-tree. For example, given the sub-tree in Figure 3, (NCOMP, N, NMOD, N, NMOD, N, POBJ, ADV, NMOD, N) is the ML-JHD feature for the top node (marked by the dashed circle).

In addition, three types of features are included: dependency relations, neighbors of the head of the current subtree, and the sizes of the sub-tree and its left and right children. The dependency relations include the top one in the subtree. The neighbors are typically within a preset distance from the head word. The sizes refer to the numbers of words or non-terminals in the subtree and its children.

[1] The other important one is the dependency score, which is the conditional probability of the last dependency relation in the subtree, given its left and right child trees

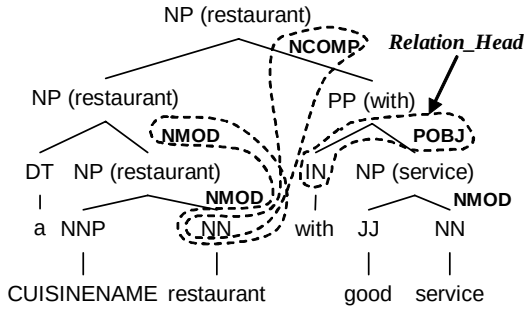


Figure 2. SL-JHD Features

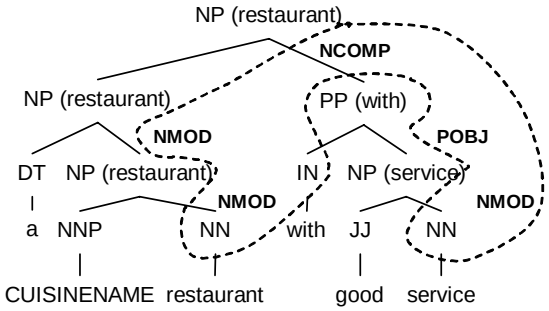


Figure 3. ML-JHD Features

4 Experiments

Experiments were conducted to see the performance of our algorithm in human to human dialogs – the ultimate goal of a dialogue system. In our work, we use a version of the Charniak's parser from (Aug. 16, 2005) to parse the re-segmented SWBD corpus (Kahn et al., 2005), and extract the parse sub-trees from the parse trees as experimental data.

The parser's training procedure is the same as (Kahn et al., 2005). The only difference is that they use golden edits in the parsing experiments while we delete all the edits in the UW Switchboard corpus. The F-score of the parsing result of the Charniak parser without edits is 88.24%.

The Charniak parser without edits is used to parse the training data, testing data and tuning data. We remove the sentences with only one word and delete the interjections in the hypothesis parse trees. Finally, we extract parse sub-trees from these hypothesis parse trees. Based on the gold parse trees, a parse sub-tree is labeled with 1 (correct), if it has all the words, their POS tags and syntactic structures correct. Otherwise, it is 0 (incorrect). Among the 424,614 parse sub-trees from the training data, 316,182 sub-trees are labeled with 1; among the 38,774 parse sub-trees from testing data, 22,521 ones are labeled with 1; and among the 67,464 parse sub-trees from the tuning data, 38,619 ones are labeled with 1. In the testing data, there are 5,590 sentences, and the percentage of complete bracket match[2] is 57.11%, and the percentage of parse sub-trees with correct labels at the sentence level is 48.57%. The percentage of correct parse sub-trees is lower than that of the complete bracket match due to its stricter requirements.

Table 1 shows our analysis of the testing data. There, the first column indicates the phrase length categories from the parse sub-trees. Among all the parse trees in the test data, 82.84% (first two rows) have a length equal to or shorter than 10 words. We converted the original parse sub-trees from the Charniak parser into binary trees.

Length	Sub-tree Types	Number	Ratio
<=10	Correct	21,593	55.70%
	Incorrect	10,525	27.14%
>10	Correct	928	2.39%
	Incorrect	5,728	14.77%

Table 1. The analysis of testing data.

We apply the model (2) from section 2 on the above data for all the following experiments. The performance is measured based on the confidence annotation error rate (Zhang and Rudnicky, 2001).

$$Annot.Error = \frac{Number Of\ Subtrees Annotated As Incorrect}{Total\ Number\ Of\ Subtrees}$$

Two sets of experiments are designed to demonstrate the improvements of our confidence computing algorithm, as well as the newly introduced features (see Table 2 and Table 3).

Experiments were conducted to evaluate the effectiveness of each feature category for the sub-tree level confidence annotation on SWBD corpus (Table 2). The baseline system uses the conventional features: words and POS tags. Additional feature categories are included separately. The syntactic feature category shows the biggest improvement among all the categories.

To see the additive effect of the feature spaces for the multi-level confidence annotation, another set of experiments were performed (Table 3). Three feature spaces are included incrementally: dependency score, hierarchical tags and syntactic features. Each category provides sizable reduction in error rate. Totally, it reduces the error rate by

[2] Complete bracket match is the percentage of sentences where bracketing recall and precision are both 100%.

	Feature Space Description	Annot. Error	Relative Error Decrease
Baseline	Base features: Words, POS tag	36.2%	\
Set 1	Base features + Dependency score	32.8%	9.4%
Set 2	Base features + Hierarchical tags	35.3%	2.5%
Set 3	Base features + Syntactic features	29.3%	19.1%

Table 2. Comparison of different feature space (on SWBD corpus).

	Feature Space Description	Annot. Error	Relative Error Decrease
Baseline	Base features: Words, POS tag	36.2%	\
Set 4	+ Dependency score	32.8%	9.4%
Set 5	+ Dependency score + hierarchical tags	32.7%	9.7%
Set 6	+ Dependency score + hierarchical tags + syntactic features	26.0%	28.2%

Table 3. Summary of experiment results with different feature space (on SWBD corpus).

10.2%, corresponding to 28.2% of a relative error reduction over the baseline. The best result of annotation error rate is 26% for Switchboard data, which is significantly lower than the 41.91% sub-tree parsing error rate (see Table 1: 41.91% = 27.14%+14.77%). So, our algorithm would also help the best parsing algorithms during rescoring (Charniak et al., 2005; McClosky et al., 2006).

We list the performance of the parse sub-trees with different lengths for Set 6 in Table 4, using the F-score as the evaluation measure.

Length	Sub-tree Category	F-score
<=10	Correct	82.3%
	Incorrect	45.9%
>10	Correct	33.1%
	Incorrect	86.1%

Table 4. F-scores for various lengths in Set 15.

The F-score difference between the ones with correct labels and the ones with incorrect labels are significant. We suspect that it is caused by the different amount of training data. Therefore, we simply duplicated the training data for the sub-trees with incorrect labels. For the sub-trees of length equal to or less than 10 words, this training method leads to a 79.8% F-score for correct labels, and a 61.4% F-score for incorrect labels, which is much more balanced than those in the first set of results.

5 Conclusion

In this paper, we generalized confidence annotation algorithms to multiple-level parse trees and demonstrated the significant benefits of using long distance features in SWBD corpora. It is foreseeable that multi-level confidence annotation can be used for many other language applications such as parsing, or information retrieval.

References

Eugene Charniak and Mark Johnson. 2005. *Coarse-to-fine n-best parsing and MaxEnt discriminative reranking*. Proc. ACL, pages 173–180.

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

Malte Gabsdil and Johan Bos. 2003. *Combining Acoustic Confidence Scores with Deep Semantic Analysis for Clarification Dialogues*. Proc. IWCS, pages 137-150.

Didier Guillevic, et al. 2002. *Robust semantic confidence scoring*. Proc. ICSLP, pages 853-856.

Jeremy G. Kahn, et al. 2005. *Effective Use of Prosody in Parsing Conversational Speech.* Proc. EMNLP, pages 233-240.

David McClosky, Eugene Charniak and Mark Johnson. 2006. *Reranking and Self-Training for Parser Adaptation.* Proc. COLING-ACL, pages 337-344.

Nicola Ueffing and Hermann Ney. 2007. *Word-Level Confidence Estimation for Machine Translation.* Computational Linguistics, 33(1):9-40.

Fuliang Weng, et al., 2007. *CHAT to Your Destination.* Proc. of the 8th SIGDial workshop on Discourse and Dialogue, pages 79-86.

Qi Zhang, Fuliang Weng and Zhe Feng. 2006. *A Pro-gressive Feature Selection Algorithm for Ultra Large Feature Spaces.* Proc. COLING-ACL, pages 561-568.

Rong Zhang and Alexander I. Rudnicky. 2001. *Word level confidence annotation using combinations of features.* Proc. Eurospeech, pages 2105-2108.

Adapting a WSJ-Trained Parser to Grammatically Noisy Text

Jennifer Foster, Joachim Wagner and Josef van Genabith
National Centre for Language Technology
Dublin City University
Ireland
jfoster, jwagner, josef@computing.dcu.ie

Abstract

We present a robust parser which is trained on a treebank of ungrammatical sentences. The treebank is created automatically by modifying Penn treebank sentences so that they contain one or more syntactic errors. We evaluate an existing Penn-treebank-trained parser on the ungrammatical treebank to see how it reacts to noise in the form of grammatical errors. We re-train this parser on the training section of the ungrammatical treebank, leading to an significantly improved performance on the ungrammatical test sets. We show how a classifier can be used to prevent performance degradation on the original grammatical data.

1 Introduction

The focus in English parsing research in recent years has moved from Wall Street Journal parsing to improving performance on other domains. Our research aim is to improve parsing performance on text which is mildly ungrammatical, i.e. text which is well-formed enough to be understood by people yet which contains the kind of grammatical errors that are routinely produced by both native and non-native speakers of a language. The intention is not to detect and correct the error, but rather to *ignore* it. Our approach is to introduce grammatical noise into WSJ sentences while retaining as much of the structure of the original trees as possible. These sentences and their associated trees are then used as training material for a statistical parser. It is important that parsing on grammatical sentences is not harmed and we introduce a parse-probability-based classifier which allows both grammatical and ungrammatical sentences to be accurately parsed.

2 Background

Various strategies exist to build robustness into the parsing process: grammar constraints can be relaxed (Fouvry, 2003), partial parses can be concatenated to form a full parse (Penstein Rosé and Lavie, 1997), the input sentence can itself be transformed until a parse can be found (Lee et al., 1995), and mal-rules describing particular error patterns can be included in the grammar (Schneider and McCoy, 1998). For a parser which tends to fail when faced with ungrammatical input, such techniques are needed. The overgeneration associated with a statistical data-driven parser means that it does not typically fail on ungrammatical sentences. However, it is not enough to return some analysis for an ungrammatical sentence. If the syntactic analysis is to guide semantic analysis, it must reflect as closely as possible what the person who produced the sentence was trying to express. Thus, while statistical, data-driven parsing has solved the robustness problem, it is not clear that it is has solved the *accurate* robustness problem.

The problem of adapting parsers to accurately handle ungrammatical text is an instance of the domain adaptation problem where the target domain is grammatically noisy data. A parser can be adapted to a target domain by training it on data from the new domain – the problem is to quickly produce high-quality training material. Our solution is to simply modify the existing training material so that it resembles material from the noisy target domain.

In order to tune a parser to syntactically ill-formed text, a treebank is automatically transformed into an ungrammatical treebank. This transformation process has two parts: 1. the yield of each tree is transformed into an ungrammatical sentence by introducing a syntax error; 2. each tree is minimally transformed, but left intact as much as possible to reflect the syntactic structure of the original "intended" sen-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 221–224,
Columbus, Ohio, USA, June 2008.

tence prior to error insertion. Artificial ungrammaticalities have been used in various NLP tasks (Smith and Eisner, 2005; Okanohara and Tsujii, 2007)

The idea of an automatically generated ungrammatical treebank was proposed by Foster (2007). Foster generates an ungrammatical version of the WSJ treebank and uses this to train two statistical parsers. The performance of both parsers significantly improves on the artificially created ungrammatical test data, but significantly degrades on the original grammatical test data. We show that it is possible to obtain significantly improved performance on ungrammatical data without a concomitant performance decline on grammatical data.

3 Generating Noisy Treebanks

Generating Noisy Sentences We apply the error introduction procedure described in detail in Foster (2007). Errors are introduced into sentences by applying the operations of word substitution, deletion and insertion. These operations can be iteratively applied to generate increasingly noisy sentences. We restrict our attention to ungrammatical sentences with a edit-distance of one or two words from the original sentence, because it is reasonable to expect a parser's performance to degrade as the input becomes more ill-formed. The operations of substitution, deletion and insertion are not carried out entirely at random, but are subject to some constraints derived from an empirical study of ill-formed English sentences (Foster, 2005). Three types of word substitution errors are produced: agreement errors, real word spelling errors and verb form errors. Any word that is not an adjective or adverb can be deleted from any position within the input sentence, but some part-of-speech tags are favoured over others, e.g. it is more likely that a determiner will be deleted than a noun. The error creation procedure can insert an arbitrary word at any position within a sentence but it has a bias towards inserting a word directly after the same word or directly after a word with the same part of speech. The empirical study also influences the frequency at which particular errors are introduced, with missing word errors being the most frequent, followed by extra word errors, real word spelling errors, agreement errors, and finally, verb form errors. Table 1 shows examples of the kind of ill-formed sentences that are produced when we apply the procedure to Wall Street Journal sentences.

Generating Trees for Noisy Sentences The tree structures associated with the modified sentences are also modified, but crucially, this modification is minimal, since a truly robust parser should return an analysis for a mildly ungrammatical sentence that remains as similar as possible to the analysis it returns for the original grammatical sentence.

Assume that (1) is an original treebank tree for the sentence *A storm is brewing*. Example (2) is then the tree for the ungrammatical sentence containing an *is/it* confusion. No part of the original tree structure is changed apart from the yield.

(1) *(S (NP A storm) (VP (VBZ is) (VP (VBG brewing))))*
(2) *(S (NP A storm) (VP (VBZ **it**) (VP (VBG brewing))))*

An example of a missing word error is shown in (3) and (4). A pre-terminal dominating an empty node is introduced into the tree at the point where the word has been omitted.

(3) *(S (NP Annotators) (VP (VBP parse) (NP the sentences)))*
(4) *(S (NP Annotators) (VP **(-NONE- 0)** (NP the sentences)))*

An example of an extra word error is shown in (5), (6) and (7). For this example, two ungrammatical trees, (6) and (7), are generated because there are two possible positions in the original tree where the extra word can be inserted which will result in a tree with the yield *He likes of the cake* and which will not result in the creation of any additional structure.

(5) *(S (NP He) (VP (VBZ likes) (NP (DT the) (NN cake))))*
(6) *(S (NP He) (VP (VBZ likes) **(IN of)** (NP (DT the) (NN cake))))*
(7) *(S (NP He) (VP (VBZ likes) (NP **(IN of)** (DT the) (NN cake))))*

4 Parser Adaptation Experiments

In order to obtain training data for our parsing experiments, we introduce syntactic noise into the usual WSJ training material, Sections 2-21, using the procedures outlined in Section 3, i.e. for every sentence-tree pair in *WSJ2-21*, we introduce an error into the sentence and then transform the tree so that it covers the newly created ungrammatical sentence. For 4 of the 20 sections in *WSJ2-21*, we apply the noise introduction procedure to its own output to

Error Type	*WSJ00*
Missing Word	*likely to* ***bring*** *new attention to the problem* → *likely to new attention to the problem*
Extra Word	*the $ 5.9 million it posted* → *the $ 5.9 million* ***I*** *it posted*
Real Word Spell	*Mr Vinken* ***is*** *chairman of Elsevier* → *Mr. Vinken* ***if*** *chairman of Elsevier*
Agreement	***this*** *event took place 35 years ago* → ***these*** *event took place 35 years ago*
Verb Form	*But the Soviets might still* ***face*** *legal obstacles* → *But the Soviets might still* ***faces*** *legal obstacles*

Table 1: Automatically Generated Ungrammatical Sentences

create even noisier data. Our first development set is a noisy version of *WSJ00*, *Noisy00*, produced by applying the noise introduction procedure to the 1,921 sentences in *WSJ00*. Our second development set is an even noisier version of *WSJ00*, *Noisiest00*, which is created by applying our noise introduction procedure to the output of *Noisy00*. We apply the same process to *WSJ23* to obtain our two test sets.

For all our parsing experiments, we use the June 2006 version of the two-stage parser reported in Charniak and Johnson (2005). Evaluation is carried out using Parseval labelled precision/recall. For extra word errors, there may be more than one gold standard tree (see (6) and (7)). When this happens the parser output tree is evaluated against all gold standard trees and the maximum f-score is chosen.

We carry out five experiments. In the first experiment, *E0*, we apply the parser, trained on well-formed data, to noisy input. The purpose of *E0* is to ascertain how well a parser trained on grammatical sentences, can ignore grammatical noise. *E0* provides a baseline against which the subsequent experimental results can be judged. In the *E1* experiments, the parser is retrained using the ungrammatical version of *WSJ2-21*. In experiment *E1error*, the parser is trained on ungrammatical material only, i.e. the noisy version of *WSJ2-21*. In experiment *E1mixed*, the parser is trained on grammatical and ungrammatical material, i.e. the original *WSJ2-21* is merged with the noisy *WSJ2-21*. In the *E2* experiments, a classifier is applied to the input sentence. If the sentence is classified as ungrammatical, a version of the parser that has been trained on ungrammatical data is employed. In the *E2ngram* experiment, we train a J48 decision tree classifier. Following Wagner et al. (2007), the decision tree features are part-of-speech n-gram frequency counts, with n ranging from 2 to 7 and with a subset of the BNC as the frequency reference corpus. The decision tree is trained on the original *WSJ2-21* and the ungrammatical *WSJ2-21*. In the *E2prob* experiment, the input sentence is parsed with two parsers, the original parser (the *E0* parser) and the parser trained on ungrammatical material (either the *E1error* or the *E1mixed* parser). A very simple classifier is used to decide which parser output to choose: if the *E1* parser returns a higher parse probability for the most likely tree than the *E0* parser, the *E1* parser output is returned. Otherwise the *E0* parser output is returned.

The baseline $E0$ results are in the first column of Table 2. As expected, the performance of a parser trained on well-formed input degrades when faced with ungrammatical input. It is also not surprising that its performance is worse on *Noisiest00* (-8.8% f-score) than it is on *Noisy00* (-4.3%) since the *Noisiest00* sentences contain two errors rather than one.

The *E1* results occupy the second and third columns of Table 2. An up arrow indicates a statistically significant improvement over the baseline results, a down arrow a statistically significant decline and a dash a change which is not statistically significant ($p < 0.01$). Training the parser on ungrammatical data has a positive effect on its performance on *Noisy00* and *Noisiest00* but has a negative effect on its performance on *WSJ00*. Training on a combination of grammatical and ungrammatical material gives the best results for all three development sets. Therefore, for the *E2* experiments we use the *E1mixed* parser rather than the *E1error* parser.

The *E2* results are shown in the last two columns of Table 2 and the accuracy of the two classifiers in Table 3. Over the three test sets, the *E2prob* classifier outperforms the *E2ngram* classifier. Both classifiers misclassify approximately 45% of the *Noisy00* sentences. However, the sentences misclassified by the *E2prob* classifier are those that are handled well by the *E0* parser, and this is reflected in the parsing results for *Noisy00*. An important feature of the

Dev Set	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
	E0			*E1-error*			*E1-mixed*			*E2prob*			*E2ngram*		
WSJ00	91.5	90.3	90.9	91.0−	89.4 ↓	90.2	91.3−	89.8 ↓	90.5	91.5−	90.2−	90.9	91.3−	89.9↓	90.6
Noisy00	87.5	85.6	86.6	89.4 ↑	86.6 ↑	88.0	89.4 ↑	86.8 ↑	88.1	89.1 ↑	86.8 ↑	87.9	88.7↑	86.2↑	87.5
Noisiest00	83.5	80.8	82.1	87.6 ↑	83.6 ↑	85.6	87.6 ↑	83.8 ↑	85.7	87.2 ↑	83.7 ↑	85.4	86.6↑	83.0↑	84.8

Table 2: Results of Parsing Experiments

Development Set	E2prob	E2ngram
WSJ00	76.7%	63.3%
Noisy00	55.1%	55.6%
Noisiest00	70.2%	66.0%

Table 3: $E2$ Classifier Accuracy

Test Set	P	R	F	P	R	F
	E0			*E2prob*		
WSJ23	91.7	90.8	91.3	91.7−	90.7−	91.2
Noisy23	87.4	85.6	86.5	89.2 ↑	87.0 ↑	88.1
Noisiest23	83.2	80.8	82.0	87.4 ↑	84.1 ↑	85.7

Table 4: Final Results for Section 23 Test Sets

E2prob classifier is that its use results in a constant performance on the grammatical data - with no significant degradation from the baseline.

Taking the *E2prob* results as our optimum, we carry out the same experiment again on our *WSJ23* test sets. The results are shown in Table 4. The same effect can be seen for the test sets as for the development sets - a significantly improved performance on the ungrammatical data *without* an accompanying performance decrease for the grammatical data. The *Noisy23* breakdown by error type is shown in Table 5. The error type which the original parser is most able to ignore is an agreement error. For this error type alone, the ungrammatical training material seems to hinder the parser. The biggest improvement occurs for real word spelling errors.

5 Conclusion

We have shown that it is possible to tune a WSJ-trained statistical parser to ungrammatical text *without affecting its performance on grammatical text*. This has been achieved using an automatically generated ungrammatical version of the WSJ treebank and a simple binary classifier which compares parse probabilities. The next step in this research is to see how the method copes on 'real' errors - this will require manual parsing of a suitably large test set.

Error Type	P	R	F	P	R	F
	E0			*E2-prob*		
Missing Word	88.5	83.7	86.0	88.9	84.3	86.5
Extra Word	87.2	89.4	88.3	89.2	89.7	89.4
Real Word Spell	84.3	83.0	83.7	89.5	88.2	88.9
Agreement	90.4	88.8	89.6	90.3	88.6	89.4
Verb Form	88.6	87.0	87.8	89.1	87.9	88.5

Table 5: *Noisy23*: Breakdown by Error Type

Acknowledgments We thank the IRCSET Embark Initiative (postdoctoral fellowship P/04/232) for supporting this research.

References

Eugene Charniak and Mark Johnson. 2005. Course-to-fine n-best-parsing and maxent discriminative reranking. In *Proceedings of ACL-2005*.

Jennifer Foster. 2005. *Good Reasons for Noting Bad Grammar: Empirical Investigations into the Parsing of Ungrammatical Written English*. Ph.D. thesis, University of Dublin, Trinity College.

Jennifer Foster. 2007. Treebanks gone bad: Parser evaluation and retraining using a treebank of ungrammatical sentences. *IJDAR*, 10(3-4), December.

Frederik Fouvry. 2003. *Robust Processing for Constraint-based Grammar Formalisms*. Ph.D. thesis, University of Essex.

Kong Joo Lee, Cheol Jung Kweon, Jungyun Seo, and Gil Chang Kim. 1995. A robust parser based on syntactic information. In *Proceedings of EACL-1995*.

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative examples. In *Proceedings of ACL-2007*.

Carolyn Penstein Rosé and Alon Lavie. 1997. An efficient distribution of labor in a two stage robust interpretation process. In *Proceedings of EMNLP-1997*.

David Schneider and Kathleen McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proceedings of ACL/COLING-1998*.

Noah A. Smith and Jason Eisner. 2005. Contrastive Estimation: Training Log-Linear Models on Unlabeled Data. In *Proceedings of ACL-2005*.

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2007. A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In *Proceedings of EMNLP-CoNLL-2007*.

Enriching spoken language translation with dialog acts

Vivek Kumar Rangarajan Sridhar
Shrikanth Narayanan
Speech Analysis and Interpretation Laboratory
University of Southern California
vrangara@usc.edu,shri@sipi.usc.edu

Srinivas Bangalore
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932, U.S.A.
srini@research.att.com

Abstract

Current statistical speech translation approaches predominantly rely on just text transcripts and do not adequately utilize the rich contextual information such as conveyed through prosody and discourse function. In this paper, we explore the role of context characterized through *dialog acts* (DAs) in statistical translation. We demonstrate the integration of the dialog acts in a phrase-based statistical translation framework, employing 3 limited domain parallel corpora (Farsi-English, Japanese-English and Chinese-English). For all three language pairs, in addition to producing interpretable DA enriched target language translations, we also obtain improvements in terms of objective evaluation metrics such as lexical selection accuracy and BLEU score.

1 Introduction

Recent approaches to statistical speech translation have relied on improving translation quality with the use of phrase translation (Och and Ney, 2003; Koehn, 2004). The quality of phrase translation is typically measured using n-gram precision based metrics such as BLEU (Papineni et al., 2002) and NIST scores. However, in many dialog based speech translation scenarios, vital information beyond what is robustly captured by words and phrases is carried by the communicative act (e.g., *question*, *acknowledgement*, etc.) representing the function of the utterance. Our approach for incorporating dialog act tags in speech translation is motivated by the fact that it is important to capture and convey not only *what* is being communicated (the words) but *how* something is being communicated (the context). Augmenting current statistical translation frameworks with *dialog acts* can potentially improve translation quality and facilitate successful cross-lingual interactions in terms of improved information transfer.

Dialog act tags have been previously used in the VERBMOBIL statistical speech-to-speech translation system (Reithinger et al., 1996). In that work, the predicted DA tags were mainly used to improve speech recognition, semantic evaluation, and information extraction modules. Discourse information in the form of speech acts has also been used in interlingua translation systems (Mayfield et al., 1995) to map input text to semantic concepts, which are then translated to target text.

In contrast with previous work, in this paper we demonstrate how dialog act tags can be directly exploited in phrase based statistical speech translation systems (Koehn, 2004). The framework presented in this paper is particularly suited for human-human and human-computer interactions in a dialog setting, where information loss due to erroneous content may be compensated to some extent through the correct transfer of the appropriate dialog act. The dialog acts can also be potentially used for imparting correct utterance level intonation during speech synthesis in the target language. Figure 1 shows an example where the detection and transfer of dialog act information is beneficial in resolving ambiguous intention associated with the translation output.

Source: آیا این مسکنه
Ref: is this a painkiller
Hyp: this is a painkiller
Enriched Hyp: this is a painkiller (Yes-No-Question)

Figure 1: Example of speech translation output enriched with dialog act

The remainder of this paper is organized as follows: Section 2 describes the dialog act tagger used in this work, Section 3 formulates the problem, Section 4 describes the parallel corpora used in our experiments, Section 5 summarizes our experimental results and Section 6 concludes the paper with a brief discussion and outline for future work.

2 Dialog act tagger

In this work, we use a dialog act tagger trained on the Switchboard DAMSL corpus (Jurafsky et al.,

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 225–228,
Columbus, Ohio, USA, June 2008.

1998) using a maximum entropy (maxent) model. The Switchboard-DAMSL (SWBD-DAMSL) corpus consists of 1155 dialogs and 218,898 utterances from the Switchboard corpus of telephone conversations, tagged with discourse labels from a shallow discourse tagset. The original tagset of 375 unique tags was clustered to obtain 42 dialog tags as in (Jurafsky et al., 1998). In addition, we also grouped the 42 tags into 7 disjoint classes, based on the frequency of the classes and grouped the remaining classes into an "Other" category constituting less than 3% of the entire data. The simplified tagset consisted of the following classes: *statement, acknowledgment, abandoned, agreement, question, appreciation, other*.

We use a maximum entropy sequence tagging model for the automatic DA tagging. Given a sequence of utterances $U = u_1, u_2, \cdots, u_n$ and a dialog act vocabulary ($d_i \in \mathcal{D}, |\mathcal{D}| = K$), we need to assign the best dialog act sequence $D^* = d_1, d_2, \cdots, d_n$. The classifier is used to assign to each utterance a dialog act label conditioned on a vector of local contextual feature vectors comprising the lexical, syntactic and acoustic information. We used the machine learning toolkit LLAMA (Haffner, 2006) to estimate the conditional distribution using maxent. The performance of the maxent dialog act tagger on a test set comprising 29K utterances of SWBD-DAMSL is shown in Table 1.

Cues used (current utterance)	Accuracy (%)	
	42 tags	**7 tags**
Lexical	69.7	81.9
Lexical+Syntactic	70.0	82.4
Lexical+Syntactic+Prosodic	70.4	82.9

Table 1: Dialog act tagging accuracies for various cues on the SWBD-DAMSL corpus.

3 Enriched translation using DAs

If S_s, T_s and S_t, T_t are the speech signals and equivalent textual transcription in the source and target language, and L_s the enriched representation for the source speech, we formalize our proposed enriched S2S translation in the following manner:

$$S_t^* = \arg\max_{S_t} P(S_t|S_s) \quad (1)$$

$$P(S_t|S_s) = \sum_{T_t,T_s,L_s} P(S_t, T_t, T_s, L_s|S_s) \quad (2)$$

$$\approx \sum_{T_t,T_s,L_s} P(S_t|T_t, L_s).P(T_t, T_s, L_s|S_s) \quad (3)$$

where Eq.(3) is obtained through conditional independence assumptions. Even though the recognition and translation can be performed jointly (Matusov et al., 2005), typical S2S translation frameworks compartmentalize the ASR, MT and TTS, with each component maximized for performance individually.

$$\begin{aligned}\max_{S_t} P(S_t|S_s) \approx & \max_{S_t} P(S_t|T_t^*, L_s^*) \\ & \times \max_{T_t} P(T_t|T_s^*, L_s^*) \\ & \times \max_{L_s} P(L_s|T_s^*, S_s) \times \max_{T_s} P(T_s|S_s)\end{aligned} \quad (4)$$

where T_s^*, T_t^* and S_t^* are the arguments maximizing each of the individual components in the translation engine. L_s^* is the rich annotation detected from the source speech signal and text, S_s and T_s^* respectively. In this work, we do not address the speech synthesis part and assume that we have access to the reference transcripts or 1-best recognition hypothesis of the source utterances. The rich annotations (L_s) can be syntactic or semantic concepts (Gu et al., 2006), prosody (Agüero et al., 2006), or, as in this work, dialog act tags.

3.1 Phrase-based translation with dialog acts

One of the currently popular and predominant schemes for statistical translation is the phrase-based approach (Koehn, 2004). Typical phrase-based SMT approaches obtain word-level alignments from a bilingual corpus using tools such as GIZA++ (Och and Ney, 2003) and extract phrase translation pairs from the bilingual word alignment using heuristics. Suppose, the SMT had access to source language dialog acts (L_s), the translation problem may be reformulated as,

$$\begin{aligned}T_t^* &= \arg\max_{T_t} P(T_t|T_s, L_s) \\ &= \arg\max_{T_t} P(T_s|T_t, L_s).P(T_t|L_s)\end{aligned} \quad (5)$$

The first term in Eq.(5) corresponds to a dialog act specific MT model and the second term to a dialog act specific language model. Given sufficient amount of training data such a system can possibly generate hypotheses that are more accurate than the scheme without the use of dialog acts. However, for small scale and limited domain applications, Eq.(5) leads to an implicit partitioning of the data corpus

	Training						Test					
	Farsi	Eng	Jap	Eng	Chinese	Eng	Farsi	Eng	Jap	Eng	Chinese	Eng
Sentences	8066		12239		46311		925		604		506	
Running words	76321	86756	64096	77959	351060	376615	5442	6073	4619	6028	3826	3897
Vocabulary	6140	3908	4271	2079	11178	11232	1487	1103	926	567	931	898
Singletons	2819	1508	2749	1156	4348	4866	903	573	638	316	600	931

Table 2: Statistics of the training and test data used in the experiments.

and might generate inferioir translations in terms of lexical selection accuracy or BLEU score.

A natural step to overcome the sparsity issue is to employ an appropriate back-off mechanism that would exploit the phrase translation pairs derived from the complete data. A typical phrase translation table consists of 5 phrase translation scores for each pair of phrases, source-to-target phrase translation probability (λ_1), target-to-source phrase translation probability (λ_2), source-to-target lexical weight (λ_3), target-to-word lexical weight (λ_4) and phrase penalty (λ_5= 2.718). The lexical weights are the product of word translation probabilities obtained from the word alignments. To each phrase translation table belonging to a particular DA-specific translation model, we append those entries from the baseline model that are not present in phrase table of the DA-specific translation model. The appended entries are weighted by a factor α.

$$(T_s \rightarrow T_t)_{L_s^*} = (T_s \rightarrow T_t)_{L_s} \cup \{\alpha.(T_s \rightarrow T_t) \; s.t. \; (T_s \rightarrow T_t) \notin (T_s \rightarrow T_t)_{L_s}\} \quad (6)$$

where $(T_s \rightarrow T_t)$ is a short-hand[1] notation for a phrase translation table. $(T_s \rightarrow T_t)_{L_s}$ is the DA-specific phrase translation table, $(T_s \rightarrow T_t)$ is the phrase translation table constructed from entire data and $(T_s \rightarrow T_t)_{L_s^*}$ is the newly interpolated phrase translation table. The interpolation factor α is used to weight each of the four translation scores (phrase translation and lexical probabilities for the bilanguage) with the phrase penalty remaining a constant. Such a scheme ensures that phrase translation pairs belonging to a specific DA model are weighted higher and also ensures better coverage than a partitioned data set.

4 Data

We report experiments on three different parallel corpora: Farsi-English, Japanese-English and Chinese-English. The Farsi-English data used in this paper was collected for human-mediated doctor-patient mediated interactions in which an English speaking doctor interacts with a Persian speaking patient (Narayanan et al., 2006). We used a subset of this corpus consisting of 9315 parallel sentences.

The Japanese-English parallel corpus is a part of the "How May I Help You" (HMIHY) (Gorin et al., 1997) corpus of operator-customer conversations related to telephone services. The corpus consists of 12239 parallel sentences. The conversations are spontaneous even though the domain is limited. The Chinese-English corpus corresponds to the IWSLT06 training and 2005 development set comprising 46K and 506 sentences respectively (Paul, 2006).

5 Experiments and Results

In all our experiments we assume that the same dialog act is shared by a parallel sentence pair. Thus, even though the dialog act prediction is performed for English, we use the predicted dialog act as the dialog act for the source language sentence. We used the Moses[2] toolkit for statistical phrase-based translation. The language models were trigram models created only from the training portion of each corpus. Due to the relatively small size of the corpora used in the experiments, we could not devote a separate development set for tuning the parameters of the phrase-based translation scheme. Hence, the experiments are strictly performed on the training and test sets reported in Table 2[3].

The lexical selection accuracy and BLEU scores for the three parallel corpora is presented in Table 3. Lexical selection accuracy is measured in terms of the F-measure derived from recall ($\frac{|Res \cap Ref|}{|Ref|} * 100$) and precision ($\frac{|Res \cap Ref|}{|Res|} * 100$), where Ref is the set of words in the reference translation and Res is

[1] $(T_s \rightarrow T_t)$ represents the mapping between source alphabet sequences to target alphabet sequences, where every pair $(t_1^s, \cdots, t_n^s, t_1^t, \cdots, t_m^t)$ has a weight sequence $\lambda_1, \cdots, \lambda_5$ (five weights).

[2] http://www.statmt.org/moses

[3] A very small subset of the data was reserved for optimizing the interpolation factor (α) described in Section 3.1

Language pair	F-score (%)			BLEU (%)		
	w/o DA tags	w/ DA tags		w/o DA tags	w/ DA tags	
		7tags	42tags		7tags	42tags
Farsi-English	56.46	57.32	57.74	22.90	23.50	23.75
Japanese-English	79.05	79.40	79.51	54.15	54.21	54.32
Chinese-English	65.85	67.24	67.49	48.59	52.12	53.04

Table 3: F-measure and BLEU scores with and without use of dialog act tags.

the set of words in the translation output. Adding dialog act tags (either 7 or 42 tag vocabulary) consistently improves both the lexical selection accuracy and BLEU score for all the language pairs. The improvements for Farsi-English and Chinese-English corpora are more pronounced than the improvements in Japanese-English corpus. This is due to the skewed distribution of dialog acts in the Japanese-English corpus; 80% of the test data are *statements* while *other* and *questions* category make up 16% and 3.5% of the data respectively. The important observation here is that, appending DA tags in the form described in this work, can improve translation performance even in terms of conventional objective evaluation metrics. However, the performance gain measured in terms of objective metrics that are designed to reflect only the orthographic accuracy during translation is not a complete evaluation of the translation quality of the proposed framework. We are currently planning of adding human evaluation to bring to fore the usefulness of such rich annotations in interpreting and supplementing typically noisy translations.

6 Discussion and Future Work

It is important to note that the dialog act tags used in our translation system are predictions from the maxent based DA tagger described in Section 2. We do not have access to the reference tags; thus, some amount of error is to be expected in the DA tagging. Despite the lack of reference DA tags, we are still able to achieve modest improvements in the translation quality. Improving the current DA tagger and developing suitable adaptation techniques are part of future work.

While we have demonstrated here that using dialog act tags can improve translation quality in terms of word based automatic evaluation metrics, the real benefits of such a scheme would be attested through further human evaluations. We are currently working on conducting subjective evaluations.

References

P. D. Agüero, J. Adell, and A. Bonafonte. 2006. Prosody generation for speech-to-speech translation. In *Proc. of ICASSP*, Toulouse, France, May.

A. Gorin, G. Riccardi, and J. Wright. 1997. How May I Help You? *Speech Communication*, 23:113–127.

L. Gu, Y. Gao, F. H. Liu, and M. Picheny. 2006. Concept-based speech-to-speech translation using maximum entropy models for statistical natural concept generation. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):377–392, March.

P. Haffner. 2006. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 48(iv):239–261.

D. Jurafsky, R. Bates, N. Coccaro, R. Martin, M. Meteer, K. Ries, E. Shriberg, S. Stolcke, P. Taylor, and C. Van Ess-Dykema. 1998. Switchboard discourse language modeling project report. Technical report research note 30, Johns Hopkins University, Baltimore, MD.

P. Koehn. 2004. Pharaoh: A beam search decoder for phrasebased statistical machine translation models. In *Proc. of AMTA-04*, pages 115–124.

E. Matusov, S. Kanthak, and H. Ney. 2005. On the integration of speech recognition and statistical machine translation. In *Proc. of Eurospeech*.

L. Mayfield, M. Gavalda, W. Ward, and A. Waibel. 1995. Concept-based speech translation. In *Proc. of ICASSP*, volume 1, pages 97–100, May.

S. Narayanan et al. 2006. Speech recognition engineering issues in speech to speech translation system design for low resource languages and domains. In *Proc. of ICASSP*, Toulose, France, May.

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

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. Technical report, IBM T.J. Watson Research Center.

M. Paul. 2006. Overview of the IWSLT 2006 Evaluation Campaign. In *Proc. of the IWSLT*, pages 1–15, Kyoto, Japan.

N. Reithinger, R. Engel, M. Kipp, and M. Klesen. 1996. Predicting dialogue acts for a speech-to-speech translation system. In *Proc. of ICSLP*, volume 2, pages 654–657, Oct.

Speakers' Intention Prediction Using Statistics of Multi-level Features in a Schedule Management Domain

Donghyun Kim
Diquest Research Center
Diquest Inc.
Seoul, Korea
kdh2007@sogang.ac.kr

Hyunjung Lee
Computer Science & Engineering
Sogang University
Seoul, Korea
juvenile@sogang.ac.kr

Choong-Nyoung Seon
Computer Science & Engineering
Sogang University
Seoul, Korea
wilowisp@gmail.com

Harksoo Kim
Computer & Communications Engineering
Kangwon National University
Chuncheon, Korea
nlpdrkim@kangwon.ac.kr

Jungyun Seo
Computer Science & Engineering
Sogang University
Seoul, Korea
seojy@sogang.ac.kr

Abstract

Speaker's intention prediction modules can be widely used as a pre-processor for reducing the search space of an automatic speech recognizer. They also can be used as a pre-processor for generating a proper sentence in a dialogue system. We propose a statistical model to predict speakers' intentions by using multi-level features. Using the multi-level features (morpheme-level features, discourse-level features, and domain knowledge-level features), the proposed model predicts speakers' intentions that may be implicated in next utterances. In the experiments, the proposed model showed better performances (about 29% higher accuracies) than the previous model. Based on the experiments, we found that the proposed multi-level features are very effective in speaker's intention prediction.

1 Introduction

A dialogue system is a program in which a user and system communicate in natural language. To understand user's utterance, the dialogue system should identify his/her intention. To respond his/her question, the dialogue system should generate the counterpart of his/her intention by referring to dialogue history and domain knowledge. Most previous researches on speakers' intentions have been focused on intention identification techniques. On the contrary, intention prediction techniques have been not studied enough although there are many practical needs, as shown in Figure 1.

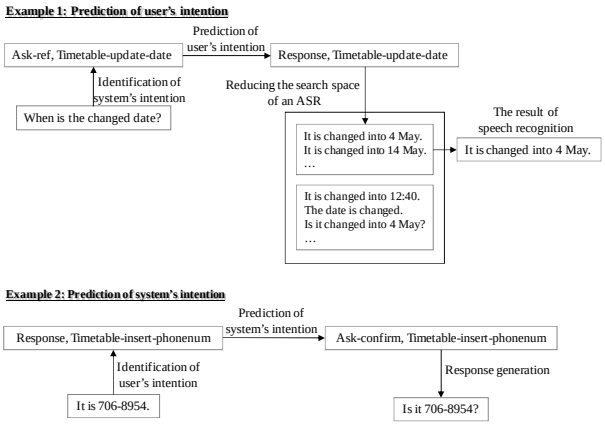


Figure 1. Motivational example

In Figure 1, the first example shows that an intention prediction module can be used as a pre-processor for reducing the search space of an ASR (automatic speech recognizer). The second example shows that an intention prediction module can be used as a pre-processor for generating a proper sentence based on dialogue history.

There are some researches on user's intention prediction (Ronnie, 1995; Reithinger, 1995). Reithinger's model used *n*-grams of speech acts as input features. Reithinger showed that his model can reduce the searching complexity of an ASR to 19~60%. However, his model did not achieve good performances because the input features were not rich enough to predict next speech acts. The researches on system's intention prediction have been treated as a part of researches on dialogue models such as a finite-state model, a frame-based

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 229–232,
Columbus, Ohio, USA, June 2008.

model (Goddeau, 1996), and a plan-based model (Litman, 1987). However, a finite-state model has a weak point that dialogue flows should be predefined. Although a plan-based model can manage complex dialogue phenomena using plan inference, a plan-based model is not easy to be applied to the real world applications because it is difficult to maintain plan recipes. In this paper, we propose a statistical model to reliably predict both user's intention and system's intention in a schedule management domain. The proposed model determines speakers' intentions by using various levels of linguistic features such as clue words, previous intentions, and a current state of a domain frame.

2 Statistical prediction of speakers' intentions

2.1 Generalization of speakers' intentions

In a goal-oriented dialogue, speaker's intention can be represented by a semantic form that consists of a speech act and a concept sequence (Levin, 2003). In the semantic form, the speech act represents the general intention expressed in an utterance, and the concept sequence captures the semantic focus of the utterance.

Table 1. Speech acts and their meanings

Speech act	Description
Greeting	The opening greeting of a dialogue
Expressive	The closing greeting of a dialogue
Opening	Sentences for opening a goal-oriented dialogue
Ask-ref	WH-questions
Ask-if	YN-questions
Response	Responses of questions or requesting actions
Request	Declarative sentences for requesting actions
Ask-confirm	Questions for confirming the previous actions
Confirm	Reponses of ask-confirm
Inform	Declarative sentences for giving some information
Accept	Agreement

Table 2. Basic concepts in a schedule management domain.

Table name	Operation name	Field name
Timetable	Insert, Delete, Select, Update	Agent, Date, Day-of-week, Time, Person, Place
Alarm	Insert, Delete, Select, Update	Date, Time

Based on these assumptions, we define 11 domain-independent speech acts, as shown in Table 1, and 53 domain-dependent concept sequences according to a three-layer annotation scheme (*i.e.* Fully connecting basic concepts with bar symbols) (Kim, 2007) based on Table 2. Then, we generalize speaker's intention into a pair of a speech act and a concept sequence. In the remains of this paper, we call a pair of a speech act and a concept sequence) an intention.

2.2 Intention prediction model

Given n utterances $U_{1,n}$ in a dialogue, let SI_{n+1} denote speaker's intention of the $n+1$th utterance. Then, the intention prediction model can be formally defined as the following equation:

$$P(SI_{n+1} \mid U_{1,n}) \approx \arg\max_{SA_{n+1}, CS_{n+1}} P(SA_{n+1}, CS_{n+1} \mid U_{1,n}) \quad (1)$$

In Equation (1), SA_{n+1} and CS_{n+1} are the speech act and the concept sequence of the $n+1$th utterance, respectively. Based on the assumption that the concept sequences are independent of the speech acts, we can rewrite Equation (1) as Equation (2).

$$P(SI_{n+1} \mid U_{1,n}) \approx \arg\max_{SA_{n+1}, CS_{n+1}} P(SA_{n+1} \mid U_{1,n}) P(CS_{n+1} \mid U_{1,n}) \quad (2)$$

In Equation (2), it is impossible to directly compute $P(SA_{n+1} \mid U_{1,n})$ and $P(CS_{n+1} \mid U_{1,n})$ because a speaker expresses identical contents with various surface forms of n sentences according to a personal linguistic sense in a real dialogue. To overcome this problem, we assume that n utterances in a dialogue can be generalized by a set of linguistic features containing various observations from the first utterance to the nth utterance. Therefore, we simplify Equation (2) by using a linguistic feature set FS_{n+1} (a set of features that are accumulated from the first utterance to nth utterance) for predicting the $n+1$th intention, as shown in Equation (3).

$$P(SI_{n+1} \mid U_{1,n}) \approx \arg\max_{SA_{n+1}, CS_{n+1}} P(SA_{n+1} \mid FS_{n+1}) P(CS_{n+1} \mid FS_{n+1}) \quad (3)$$

All terms of the right hand side in Equation (3) are represented by conditional probabilities given a various feature values. These conditional probabilities can be effectively evaluated by CRFs (conditional random fields) (Lafferty, 2001) that globally consider transition probabilities from the first ut-

terance to the *n+1*th utterance, as shown in Equation (4).

$$P_{CRF}(SA_{1,n+1} \mid FS_{1,n+1}) = \frac{1}{Z(FS_{1,n+1})} \exp\left(\sum_{i=1}^{n+1} \sum_{j} \lambda_j F_j(SA_i, FS_i)\right) \quad (4)$$

$$P_{CRF}(CS_{1,n+1} \mid FS_{1,n+1}) = \frac{1}{Z(FS_{1,n+1})} \exp\left(\sum_{i=1}^{n+1} \sum_{j} \lambda_j F_j(CS_i, FS_i)\right)$$

In Equation (4), $F_j(SA_i, FS_i)$ and $F_j(CS_i, FS_i)$ are feature functions for predicting the speech act and the concept sequence of the *i*th utterance, respectively. $Z(FS)$ is a normalization factor. The feature functions receive binary values (*i.e.* zero or one) according to absence or existence of each feature.

2.3 Multi-level features

The proposed model uses multi-level features as input values of the feature functions in Equation (4). The followings give the details of the proposed multi-level features.

- Morpheme-level feature: Sometimes a few words in a current utterance give important clues to predict an intention of a next utterance. We propose two types of morpheme-level features that are extracted from a current utterance: One is lexical features (content words annotated with parts-of-speech) and the other is POS features (part-of-speech bi-grams of all words in an utterance). To obtain the morpheme-level features, we use a conventional morphological analyzer. Then, we remove non-informative feature values by using a well-known χ^2 statistic because the previous works in document classification have shown that effective feature selection can increase precisions (Yang, 1997).
- Discourse-level feature: An intention of a current utterance affects that dialogue participants determine intentions of next utterances because a dialogue consists of utterances that are sequentially associated with each other. We propose discourse-level features (bigrams of speakers' intentions; a pair of a current intention and a next intention) that are extracted from a sequence of utterances in a current dialogue.
- Domain knowledge-level feature: In a goal-oriented dialogue, dialogue participants accomplish a given task by using shared domain knowledge. Since a frame-based model is more flexible than a finite-state model and is more easy-implementable than a plan-based model, we adopt the frame-based model in order to describe domain knowledge. We propose two types of domain knowledge-level features; slot-modification features and slot-retrieval features. The slot-modification features represent which slots are filled with suitable items, and the slot-retrieval features represent which slots are looked up. The slot-modification features and the slot-retrieval features are represented by binary notation. In the slot-modification features, '1' means that the slot is filled with a proper item, and '0' means that the slot is empty. In the slot-retrieval features, '1' means that the slot is looked up one or more times. To obtain domain knowledge-level features, we predefined speakers' intentions associated with slot modification (*e.g.* 'response & timetable-update-date') and slot retrieval (*e.g.* 'request & timetable-select-date'), respectively. Then, we automatically generated domain knowledge-level features by looking up the predefined intentions at each dialogue step.

3 Evaluation

3.1 Data sets and experimental settings

We collected a Korean dialogue corpus simulated in a schedule management domain such as appointment scheduling and alarm setting. The dialogue corpus consists of 956 dialogues, 21,336 utterances (22.3 utterances per dialogue). Each utterance in dialogues was manually annotated with speech acts and concept sequences. The manual tagging of speech acts and concept sequences was done by five graduate students with the knowledge of a dialogue analysis and post-processed by a student in a doctoral course for consistency. To experiment the proposed model, we divided the annotated messages into the training corpus and the testing corpus by a ratio of four (764 dialogues) to one (192 dialogues). Then, we performed 5-fold cross validation. We used training factors of CRFs as L-BGFS and Gaussian Prior.

3.2 Experimental results

Table 3 and Table 4 show the accuracies of the proposed model in speech act prediction and concept sequence prediction, respectively.

Table 3. The accuracies of speech act prediction

Features	Accuracy-S (%)	Accuracy-U (%)
Morpheme-level features	76.51	72.01
Discourse-level features	87.31	72.80
Domain knowledge-level feature	63.44	49.03
All features	88.11	76.25

Table 4. The accuracies of concept sequence prediction

Features	Accuracy-S (%)	Accuracy-U (%)
Morpheme-level features	66.35	59.40
Discourse-level features	86.56	62.62
Domain knowledge-level feature	37.68	49.03
All features	87.19	64.21

In Table 3 and Table 4, *Accuracy-S* means the accuracy of system's intention prediction, and *Accuracy-U* means the accuracy of user's intention prediction. Based on these experimental results, we found that multi-level features include different types of information and cooperation of the multi-level features brings synergy effect. We also found the degree of feature importance in intention prediction (*i.e.* discourse level features > morpheme-level features > domain knowledge-level features).

To evaluate the proposed model, we compare the accuracies of the proposed model with those of Reithinger's model (Reithinger, 1995) by using the same training and test corpus, as shown in Table 5.

Table 5. The comparison of accuracies

Speaker	Type	Reithinger's model	The proposed model
System	Speech act	43.37	88.11
	Concept sequence	68.06	87.19
User	Speech act	37.59	76.25
	Concept sequence	49.48	64.21

As shown in Table 5, the proposed model outperformed Reithinger's model in all kinds of predictions. We think that the differences between accuracies were mainly caused by input features: The proposed model showed similar accuracies to Reithinger's model when it used only domain knowledge-level features.

4 Conclusion

We proposed a statistical prediction model of speakers' intentions using multi-level features. The model uses three levels (a morpheme level, a discourse level, and a domain knowledge level) of features as input features of the statistical model based on CRFs. In the experiments, the proposed model showed better performances than the previous model. Based on the experiments, we found that the proposed multi-level features are very effective in speaker's intention prediction.

Acknowledgments

This research (paper) was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

References

D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. 1996. "A Form-Based Dialogue Manager for Spoken Language Applications", *Proceedings of International Conference on Spoken Language Processing*, 701-704.

D. Litman and J. Allen. 1987. *A Plan Recognition Model for Subdialogues in Conversations*, Cognitive Science, 11:163-200.

H. Kim. 2007. *A Dialogue-based NLIDB System in a Schedule Management Domain: About the method to Find User's Intentions*, Lecture Notes in Computer Science, 4362:869-877.

J. Lafferty, A. McCallum, and F. Pereira. 2001. "Conditional Random Fields: Probabilistic Models for Segmenting And Labeling Sequence Data", *Proceedings of ICML*, 282-289.

L. Levin, C. Langley, A. Lavie, D. Gates, D. Wallace, and K. Peterson. 2003. "Domain Specific Speech Acts for Spoken Language Translation", *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*.

N. Reithinger and E. Maier. 1995. "Utilizing Statistical Dialog Act Processing in VerbMobil", *Proceedings of ACL*, 116-121.

R. W. Smith and D. R. Hipp, 1995. *Spoken Natural Language Dialogue Systems: A Practical Approach*, Oxford University Press.

Y. Yang and J. Pedersen. 1997. "A Comparative Study on Feature Selection in Text Categorization", *Proceedings of the 14th International Conference on Machine Learning*.

Active Learning with Confidence

Mark Dredze and **Koby Crammer**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{mdredze,crammer}@cis.upenn.edu

Abstract

Active learning is a machine learning approach to achieving high-accuracy with a small amount of labels by letting the learning algorithm choose instances to be labeled. Most of previous approaches based on discriminative learning use the margin for choosing instances. We present a method for incorporating confidence into the margin by using a newly introduced online learning algorithm and show empirically that confidence improves active learning.

1 Introduction

Successful applications of supervised machine learning to natural language rely on quality labeled training data, but annotation can be costly, slow and difficult. One popular solution is Active Learning, which maximizes learning accuracy while minimizing labeling efforts. In active learning, the learning algorithm itself selects unlabeled examples for annotation. A variety of techniques have been proposed for selecting examples that maximize system performance as compared to selecting instances randomly.

Two learning methodologies dominate NLP applications: probabilistic methods — naive Bayes, logistic regression — and margin methods — support vector machines and passive-aggressive. Active learning for probabilistic methods often uses uncertainty sampling: label the example with the lowest probability prediction (the most "uncertain") (Lewis and Gale, 1994). The equivalent technique for margin learning associates the margin with prediction certainty: label the example with the lowest margin (Tong and Koller, 2001). Common intuition equates large margins with high prediction confidence.

However, confidence and margin are two distinct properties. For example, an instance may receive a large margin based on a single feature which has been updated only a small number of times. Another example may receive a small margin, but its features have been learned from a large number of examples. While the first example has a larger margin it has low confidence compared to the second. Both the margin value and confidence should be considered in choosing which example to label.

We present active learning with confidence using a recently introduced online learning algorithm called Confidence-Weighted linear classification. The classifier assigns labels according to a Gaussian distribution over margin values instead of a single value, which arises from parameter confidence (variance). The variance of this distribution represents the confidence in the mean (margin). We then employ this distribution for a new active learning criteria, which in turn could improve other margin based active learning techniques. Additionally, we favor the use of an online method since online methods have achieved good NLP performance and are fast to train — an important property for interactive learning. Experimental validation on a number of datasets shows that active learning with confidence can improve standard methods.

2 Confidence-Weighted Linear Classifiers

Common online learning algorithms, popular in many NLP tasks, are not designed to deal with the particularities of natural language data. Fea-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 233–236,
Columbus, Ohio, USA, June 2008.

ture representations have very high dimension and most features are observed on a small fraction of instances. Confidence-weighted (CW) linear classification (Dredze et al., 2008), a new online algorithm, maintains a probabilistic measure of parameter confidence leading to a measure of prediction confidence, potentially useful for active learning. We summarize CW learning to familiarize the reader.

Parameter confidence is formalized with a distribution over weight vectors, specifically a Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and diagonal covariance $\Sigma \in \mathbb{R}^{N \times N}$. The values μ_j and $\Sigma_{j,j}$ represent knowledge of and confidence in the parameter for feature j. The smaller $\Sigma_{j,j}$, the more confidence we have in the mean parameter value μ_j.

A model predicts the label with the highest probability, $\max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)} [y(\boldsymbol{w} \cdot \boldsymbol{x}) \geq 0]$. The Gaussian distribution over parameter vectors $\boldsymbol{w}$ induces a univariate Gaussian distribution over the unsigned-margin $\mathrm{M} = \boldsymbol{w} \cdot \boldsymbol{x}$ parameterized by $\boldsymbol{\mu}$, Σ and the instance $\boldsymbol{x}$: $\mathrm{M} \sim \mathcal{N}(M, V)$, where the mean is $M = \boldsymbol{\mu} \cdot \boldsymbol{x}$ and the variance $V = \boldsymbol{x}^\top \Sigma \boldsymbol{x}$.

CW is an online algorithm inspired by the Passive Aggressive (PA) update (Crammer et al., 2006) — which ensures a positive margin while minimizing parameter change. CW replaces the Euclidean distance used in the PA update with the KL divergence over Gaussian distributions. It also replaces the minimal margin constraint with a minimal probability constraint: with some given probability $\eta \in (0.5, 1]$ a drawn classifier will assign the correct label. This strategy yields the following objective solved on each round of learning:

$$(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = \min \mathrm{D}_{\mathrm{KL}} \left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \parallel \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i) \right) \\ \text{s.t. } \Pr \left[y_i \left(\boldsymbol{\mu} \cdot \boldsymbol{x}_i \right) \geq 0 \right] \geq \eta \,,$$

where $(\boldsymbol{\mu}_i, \Sigma_i)$ are the parameters on round i and $\left(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}\right)$ are the new parameters after update. The constraint ensures that the resulting parameters will correctly classify $\boldsymbol{x}_i$ with probability at least η. For convenience we write $\phi = \Phi^{-1}(\eta)$, where Φ is the cumulative function of the normal distribution. The optimization problem above is not convex, but a closed form approximation of its solution has the following additive form: $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i y_i \Sigma_i \boldsymbol{x}_i$ and $\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha_i \phi \boldsymbol{x}_i \boldsymbol{x}_i^\top$ for,

$$\alpha_i = \frac{-(1+2\phi M_i) + \sqrt{(1+2\phi M_i)^2 - 8\phi\,(M_i - \phi V_i)}}{4\phi V_i} \,.$$

Each update changes the feature weights $\boldsymbol{\mu}$, and increases confidence (variance Σ always decreases).

3 Active Learning with Confidence

We consider pool based active learning. An active learning algorithm is given a pool of unlabeled instances $\mathcal{U} = \{\boldsymbol{x}_i\}_{i=1}^n$, a learning algorithm $\mathcal{A}$ and a set of labeled examples initially set to be $\mathcal{L} = \emptyset$. On each round the active learner uses its selection criteria to return a single instance $\boldsymbol{x}_i$ to be labeled by an annotator with $y_i \in \{-1, +1\}$ (for binary classification). The instance and label are added to the labeled set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\boldsymbol{x}_i, y_i)\}$ and passed to the learning algorithm $\mathcal{A}$, which in turn generates a new model. At the end of labeling the algorithm returns a classifier trained on the final labeled set. Effective active learning minimizes prediction error and the number of labeled examples.

Most active learners for margin based algorithms rely on the magnitude of the margin. Tong and Koller (2001) motivate this approach by considering the half-space representation of the hypothesis space for learning. They suggest three margin based active learning methods: Simple margin, MaxMin margin, and Ratio margin. In Simple margin, the algorithm predicts an unsigned margin M for each instance in $\mathcal{U}$ and returns for labeling the instance with the smallest margin. The intuition is that instances for which the classifier is uncertain (small margin) provide the most information for learning. Active learning based on PA algorithms runs in a similar fashion but full SVM retraining on every round is replaced with a single PA update using the new labeled example, greatly increasing learning speed.

Maintaining a distribution over prediction functions makes the CW algorithm attractive for active learning. Instead of using a geometrical quantity ("margin"), it use a probabilistic quantity and picks the example whose label is predicted with the lowest probability. Formally, the margin criteria, $\boldsymbol{x} = \arg\min_{\boldsymbol{z} \in \mathcal{U}} (\boldsymbol{w} \cdot \boldsymbol{z})$, is replaced with a probabilistic criteria $\boldsymbol{x} = \arg\min_{\boldsymbol{z} \in \mathcal{U}} \left| \left(\Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)} [\mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{z}) = 1] \right) - \frac{1}{2} \right|$.

The selection criteria naturally captures the notion that we should label the example with the highest uncertainty. Interestingly, we can show (omitted due to lack of space) that the probabilistic criteria can be translated into a *corrected* geometrical criteria. In practice, we can compute this normalized margin as $\bar{\mathrm{M}} = M/\sqrt{V}$. We call this selection criteria Active Confident Learning (ACL).

4 Evaluation

To evaluate our active learning methods we used a similar experimental setup to Tong and Koller (2001). Each active learning algorithm was given two labeled examples, one from each class, for initial training of a classifier, and remaining data as unlabeled examples. On each round the algorithm selected a single instance for which it was then given the correct label. The algorithm updated the online classifier and evaluated it on held out test data to measure learning progress.

We selected four binary NLP datasets for evaluation: 20 Newsgroups[1] and Reuters (Lewis et al., 2004) (used by Tong and Koller) and sentiment classification (Blitzer et al., 2007) and spam (Bickel, 2006). For each dataset we extracted binary unigram features and sentiment was prepared according to Blitzer et al. (2007). From 20 Newsgroups we created 3 binary decision tasks to differentiate between two similar labels from computers, science and talk. We created 3 similar problems from Reuters from insurance, business services and retail distribution. Sentiment used 4 Amazon domains (book, dvd, electronics, kitchen). Spam used the three users from task A data. Each problem had 2000 instances except for 20 Newsgroups, which used between 1850 and 1971 instances. This created 13 classification problems across four tasks.

Each active learning algorithm was evaluated using a PA (with slack variable $c = 1$) or CW classifier ($\phi = 1$) using 10-fold cross validation. We evaluated several methods in the Simple margin framework: PA Margin and CW Margin, which select examples with the smallest margin, and ACL. As a baseline we included selecting a random instance. We also evaluated CW and a PA classifier trained on all training instances. Each method was evaluated by labeling up to 500 labels, about 25% of the training data. The 10 runs on each dataset for each problem appear in the left and middle panel of Fig. 1, which show the test accuracy after each round of active learning. Horizontal lines indicate CW (solid) and PA (dashed) training on all instances. Legend numbers are accuracy after 500 labels. The left panel averages results over 20 Newsgroups, and the middle panel averages results over *all* 13 datasets.

To achieve 80% of the accuracy of training on all data, a realistic goal for less than 100 labels, PA Margin required 93% the number of labels of PA Random, while CW Margin needed only 73% of the labels of CW Random. By using fewer labels compared to random selection baselines, CW Margin learns faster in the active learning setting as compared with PA. Furthermore, adding confidence reduced labeling cost compared to margin alone. ACL improved over CW Margin on every task and after almost every round; it required 63% of the labels of CW Random to reach the 80% mark.

We computed the fraction of labels CW Margin and ACL required (compared to CW Random) to achieve the 80% accuracy mark of training with all data. The results are summarized in the right panel of Fig. 1, where we plot one point per dataset. Points above the diagonal-line demonstrate the superiority of ACL over CW Margin. ACL required fewer labels than CW margin twice as often as the opposite occurred (8 vs 4). Note that CW Margin used *more* labels than CW Random in three cases, while ACL only once, and this one time only about a dozen labels were needed. To conclude, not only does CW Margin outperforms PA Margin for active-learning, CW maintains additional valuable information (confidence), which further improves performance.

5 Related Work

Active learning has been widely used for NLP tasks such as part of speech tagging (Ringger et al., 2007), parsing (Tang et al., 2002) and word sense disambiguation (Chan and Ng, 2007). Many methods rely on entropy-based scores such as uncertainty sampling (Lewis and Gale, 1994). Others use margin based methods, such as Kim et al. (2006), who combined margin scores with corpus diversity, and Sassano (2002), who considered SVM active learning

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

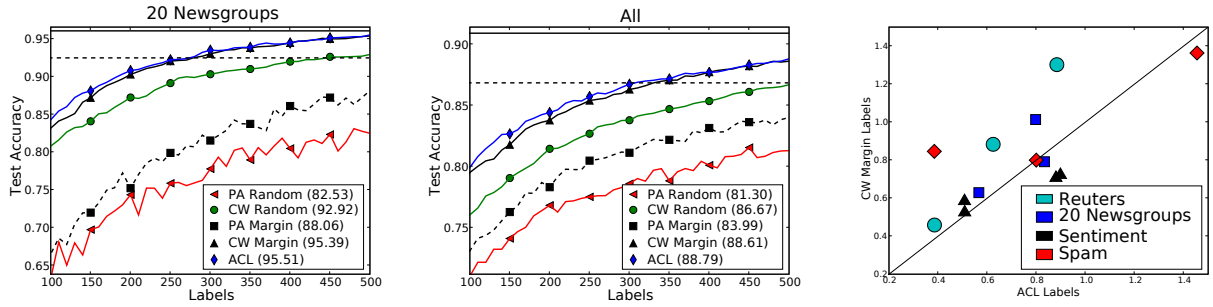


Figure 1: Results averaged over 20 Newsgroups (left) and all datasets (center) showing test accuracy over active learning rounds. The right panel shows the amount of labels needed by CW Margin and ACL to achieve 80% of the accuracy of training on all data - each points refers to a different dataset.

for Japanese word segmentation. Our confidence based approach can be used to improve these tasks. Furthermore, margin methods can outperform probabilistic methods; CW beats maximum entropy on many NLP tasks (Dredze et al., 2008).

A theoretical analysis of margin based methods selected labels that maximize the reduction of the version space, the hypothesis set consistent with the training data (Tong and Koller, 2001). Another approach selects instances that minimize the future error in probabilistic algorithms (Roy and McCallum, 2001). Since we consider an online learning algorithm our techniques can be easily extended to online active learning (Cesa-Bianchi et al., 2005; Dasgupta et al., 2005; Sculley, 2007).

6 Conclusion

We have presented techniques for incorporating confidence into the margin for active learning and have shown that CW selects better examples than PA, a popular online algorithm. This approach creates opportunities for new active learning frameworks that depend on margin confidence.

References

S. Bickel. 2006. Ecml-pkdd discovery challenge overview. In *The Discovery Challenge Workshop*.

J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Nicolò Cesa-Bianchi, Gábor Lugosi, and Gilles Stolt. 2005. Minimizing regret with label efficient prediction. *IEEE Tran. on Inf. Theory*, 51(6), June.

Y. S. Chan and H. T. Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Association for Computational Linguistics (ACL)*.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

S. Dasgupta, A.T. Kalai, and C. Monteleoni. 2005. Analysis of perceptron-based active learning. In *COLT*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *ICML*.

S. Kim, Yu S., K. Kim, J-W Cha, and G.G. Lee. 2006. Mmr-based active machine learning for bio named entity recognition. In *NAACL/HLT*.

D. D. Lewis and W. A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*.

D. D. Lewis, Y. Yand, T. Rose, and F. Li. 2004. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397.

E. Ringger, P. McClanahan, R. Haertel, G. Busby, M. Carmen, J. Carroll, K. Seppi, and D. Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *ACL Linguistic Annotation Workshop*.

N. Roy and A. McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *ICML*.

Manabu Sassano. 2002. An empirical study of active learning with support vector machines for japanese word segmentation. In *ACL*.

D. Sculley. 2007. Online active learning methods for fast label-efficient spam filtering. In *CEAS*.

M. Tang, X. Luo, and S. Roukos. 2002. Active learning for statistical natural language parsing. In *ACL*.

S. Tong and D. Koller. 2001. Supprt vector machine active learning with applications to text classification. *JMLR*.

splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications

Yoav Goldberg and **Michael Elhadad**
Ben Gurion University of the Negev
Department of Computer Science
POB 653 Be'er Sheva, 84105, Israel
{yoavg,elhadad}@cs.bgu.ac.il

Abstract

We present a fast, space efficient and non-heuristic method for calculating the decision function of polynomial kernel classifiers for NLP applications. We apply the method to the MaltParser system, resulting in a Java parser that parses over 50 sentences per second on modest hardware without loss of accuracy (a 30 time speedup over existing methods). The method implementation is available as the open-source splitSVM Java library.

1 Introduction

Over the last decade, many natural language processing tasks are being cast as classification problems. These are then solved by of-the-shelf machine-learning algorithms, resulting in state-of-the-art results. Support Vector Machines (SVMs) have gained popularity as they constantly outperform other learning algorithms for many NLP tasks.

Unfortunately, once a model is trained, the decision function for kernel-based classifiers such as SVM is expensive to compute, and can grow linearly with the size of the training data. In contrast, the computational complexity for the decisions functions of most non-kernel based classifiers does not depend on the size of the training data, making them orders of magnitude faster to compute. For this reason, research effort was directed at speeding up the classification process of polynomial-kernel SVMs (Isozaki and Kazawa, 2002; Kudo and Matsumoto, 2003; Wu et al., 2007). Existing accelerated SVM solutions, however, either require large amounts of memory, or resort to heuristics – computing only an approximation to the real decision function.

This work aims at speeding up the decision function computation for low-degree polynomial kernel classifiers while using only a modest amount of memory and still computing the exact function. This is achieved by taking into account the Zipfian nature of natural language data, and structuring the computation accordingly. On a sample application (replacing the libsvm classifier used by MaltParser (Nivre et al., 2006) with our own), we observe a speedup factor of 30 in parsing time.

2 Background and Previous Work

In classification based NLP algorithms, a word and its context is considered a learning sample, and encoded as *Feature Vectors*. Usually, context data includes the word being classified (w_0), its part-of-speech (PoS) tag (p_0), word forms and PoS tags of neighbouring words ($w_{-2}, \ldots, w_{+2}$, $p_{-2}, \ldots, p_{+2}$, etc.). Computed features such as the length of a word or its suffix may also be added. A feature vector (F) is encoded as an indexed list of all the features present in the training corpus. A feature f_i of the form $w_{+1} = dog$ means that the word following the one being classified is 'dog'. Every learning sample is represented by an $n = |F|$ dimensional binary vector x. $x_i = 1$ iff the feature f_i is active in the given sample, 0 otherwise. n is the number of different features being considered. This encoding leads to vectors with extremely high dimensions, mainly because of lexical features w_i.

SVM is a supervised binary classifier. The result of the learning process is the set SV of Sup-

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 237–240,
Columbus, Ohio, USA, June 2008.

port Vectors, associated weights α_i, and a constant b. The Support Vectors are a subset of the training feature vectors, and together with the weights and b they define a hyperplane that optimally separates the training samples. The basic SVM formulation is of a linear classifier, but by introducing a kernel function K that non-linearly transforms the data from R^n into a space of higher dimension, SVM can be used to perform non-linear classification. SVM's decision function is:

$$y(x) = sgn\left(\sum_{j \in SV} y_j \alpha_j K(x_j, x) + b\right)$$

where x is an n dimensional feature vector to be classified. The kernel function we consider in this paper is a polynomial kernel of degree d: $K(x_i, x_j) = (\gamma x_i \cdot x_j + c)^d$. When using binary valued features (with $\gamma = 1$ and $c = 1$), this kernel function essentially implies that the classifier considers not only the explicitly specified features, but also all available sets of size d of features. For $d = 2$, this means considering all feature pairs, while for $d = 3$ all feature triplets. In practice, a polynomial kernel with $d = 2$ usually yields the best results in NLP tasks, while higher degree kernels tend to overfit the data.

2.1 Decision Function Computation

Note that the decision function involves a summation over all support vectors x_j in SV. In natural language applications, the size $|SV|$ tends to be very large (Isozaki and Kazawa, 2002), often above 10,000. In particular, the size of the support vectors set can grow linearly with the number of training examples, of which there are usually at least tens of thousands. As a consequence, the computation of the decision function is computationally expensive. Several approaches have been designed to speed up the decision function computation.

Classifier Splitting is a common, application specific heuristic, which is used to speed up the training as well as the testing stages (Nivre et al., 2006). The training data is split into several datasets according to an application specific heuristic. A separate classifier is then trained for each dataset. For example, it might be known in advance that `nouns` usually behave differently than `verbs`. In such a case, one can train one classifier on `noun` instances, and a different classifier on `verb` instances. When testing, only one of the classifiers will be applied, depending on the PoS of the word. This technique reduces the number of support vectors in each classifier (because each classifier was trained on only a portion of the data). However, it relies on human intuition on the way the data should be split, and usually results in a degradation in performance relative to a single classifier trained on all the data points.

PKI – Inverted Indexing (Kudo and Matsumoto, 2003), stores for each feature the support vectors in which it appears. When classifying a new sample, only the set of vectors relevant to features actually appearing in the sample are considered. This approach is non-heuristic and intuitively appealing, but in practice brings only modest improvements.

Kernel Expansion (Isozaki and Kazawa, 2002) is used to transform the d-degree polynomial kernel based classifier into a linear one, with a modified decision function $y(x) = sgn(w \cdot x^d + b)$. w is a very high dimensional weight vector, which is calculated beforehand from the set of support vectors and their corresponding α_i values. (the calculation details appear in (Isozaki and Kazawa, 2002; Kudo and Matsumoto, 2003)). This speeds up the decision computation time considerably, as only $|x|^d$ weights need to be considered, $|x|$ being the number of active features in the sample to be classified, which is usually a very small number. However, even the sparse-representation version of w tends to be very large: (Isozaki and Kazawa, 2002) report that some of their second degree expanded NER models were more than 80 times slower to load than the original models (and 224 times faster to classify).[1] This approach obviously does not scale well, both to tasks with more features and to larger degree kernels.

PKE – Heuristic Kernel Expansion, was introduced by (Kudo and Matsumoto, 2003). This heuristic method addresses the deficiency of the Kernel Expansion method by using a basket-mining algorithm in order to greatly reduce the number of non-zero elements in the calculated w. A parameter is used to control the number of non-zero elements in w. The smaller the number, the smaller the memory requirement, but setting this number too low hurts classification performance, as only an approxima-

[1] Using a combination of 33 classifiers, the overall loading time is about 31 times slower, and classification time is about 21 times faster, than the non-expanded classifiers.

tion of the real decision function is calculated.

"Semi Polynomial Kernel" was introduced by (Wu et al., 2007). The intuition behind this optimization is to "extend the linear kernel SVM toward polynomial". It does not train a polynomial kernel classifier, but a regular linear SVM. A basket-mining based feature selection algorithm is used to select "useful" pairs and triplets of features prior to the training stage, and a linear classifier is then trained using these features. Training (and testing) are faster then in the polynomial kernel case, but the result suffer quite a big loss in accuracy as well.[2].

3 Fast, Non-Heuristic Computation

We now turn to present our fast, space efficient and non-heuristic approach for computing the Polynomial Kernel decision function.[3] Our approach is a combination of the PKI and the Kernel Expansion methods. While previous works considered kernels of the form $K(x,y) = (x \cdot y + 1)^d$, we consider the more general form of the polynomial kernel: $K(x,y) = (\gamma x \cdot y + c)^d$.

Our key observation is that in NLP classification tasks, few of the features (*e.g.,* `PoS is X`, or `prev word is the`) are very frequent, while most others are extremely rare (*e.g.,* `next word is polynomial`). The common features are active in many of the support-vectors, while the rare features are active only in few support vectors. This is true for most language related tasks: the Zipfian nature of language phenomena is reflected in the distribution of features in the support vectors.

It is because of common features that the PKI reverse indexing method does not yield great improvements: if at least one of the features of the current instance is active in a support vector, this vector is taken into account in the sum calculation, and the common features are active in many support vectors.

On the other hand, the long tail of rare features is the reason the Kernel Expansion methods requires so much space: every rare feature adds many possible feature pairs.

We propose a combined method. We first split common from rare features. We then use Kernel Expansion on the few common features, and PKI for the remaining rare features. This ensures small memory footprint for the expanded kernel vector, while at the same time keeping a low number of vectors from the reverse index.

3.1 Formal Details

The polynomial kernel of degree 2 is: $K(x,y) = (\gamma x \cdot y + c)^2$, where x and y are binary feature vectors. $x \cdot y$ is the dot product between the vectors, and in the case of binary feature vectors it corresponds to the count of shared features among the vectors. F is the set of all possible features.

We define F_R and F_C to be the sets of rare and common features. $F_R \cap F_C = \emptyset$, $F_R \cup F_C = F$. The mapping function $\phi_R(x)$ zeros out all the elements of x not belonging to F_R, while $\phi_C(x)$ zeroes out all the elements of x not in F_C. Thus, for every x: $\phi_R(x) + \phi_C(x) = x$, $\phi_R(x) \cdot \phi_C(x) = 0$. For brevity, denote $\phi_C(x) = x_C$, $\phi_R(x) = x_R$.

We now rewrite the kernel function:

$$\begin{aligned} K(x,y) &= K(x_R + x_C, y_R + y_C) = \\ &= (\gamma(x_R + x_C) \cdot (y_R + y_C) + c)^2 \\ &= (\gamma x_R \cdot y_R + \gamma x_C \cdot y_C + c)^2 \\ &= (\gamma x_R \cdot y_R)^2 \\ &\quad + 2\gamma^2 (x_R \cdot y_R)(x_C \cdot y_C) \\ &\quad + 2c\gamma (x_R \cdot y_R) \\ &\quad + (\gamma(x_C \cdot y_C) + c)^2 \end{aligned}$$

The first 3 terms are non-zero only when at least one rare feature exists. We denote their sum $K_R(x,y)$. The last term involves only common features. We denote it $K_C(x,y)$. Note that $K_C(x,y)$ is the polynomial kernel of degree 2 over feature vectors of only common features.

We can now write the SVM decision function as:

$$\sum_{j \in SV} y_j \alpha_j K_R(x_j, x_R) + \sum_{j \in SV} y_j \alpha_j K_C(x_j, x_C) + b$$

We calculate the first sum via PKI, taking into account only support-vectors which share at least one feature with x_R. The second sum is calculated via kernel expansion while taking into account only the

[2] This loss of accuracy in comparison to the PKE approach is to be expected, as (Goldberg and Elhadad, 2007) showed that the effect of removing features prior to the learning stage is much more severe than removing them after the learning stage.

[3] Our presentation is for the case where $d = 2$, as this is by far the most useful kernel. However, the method can be easily adapted to higher degree kernels as well. For completeness, our toolkit provides code for $d = 3$ as well as 2.

common features. Thus, only pairs of common features appear in the resulting weight vector using the same expansion as in (Kudo and Matsumoto, 2003; Isozaki and Kazawa, 2002). In our case, however, the expansion is memory efficient, because we consider only features in F_C, which is small.

Our approach is similar to the PKE approach (Kudo and Matsumoto, 2003), which used a basket mining approach to prune many features from the expansion. In contrast, we use a simpler approach to choose which features to include in the expansion, and we also compensate for the feature we did not include by the PKI method. Thus, our method generates smaller expansions while computing the exact decision function and not an approximation of it.

We take every feature occurring in less than s support vectors to be rare, and the other features to be common. By changing s we get a trade-of between space and time complexity: smaller s indicate more common features (bigger memory requirement) but also less rare features (less support vectors to include in the summation), and vice-versa. In contrast to other methods, changing s is guaranteed not to change the classification accuracy, as it does not change the computed decision function.

4 Toolkit and Evaluation

Using this method, one can accelerate SVM-based NLP application by just changing the classification function, keeping the rest of the logic intact. We implemented an open-source software toolkit, freely available at http://www.cs.bgu.ac.il/∼nlpproj/. Our toolkit reads models created by popular SVM packages (libsvm, SVMLight, TinySVM and Yamcha) and transforms them into our format. The transformed models can then be used by our efficient Java implementation of the method described in this paper. We supply wrappers for the interfaces of libsvm and the Java bindings of SVMLight. Changing existing Java code to accommodate our fast SVM classifier is done by loading a different model, and changing a single function call.

4.1 Evaluation: Speeding up MaltParser

We evaluate our method by using it as the classification engine for the Java version of MaltParser, an SVM-based state of the art dependency parser (Nivre et al., 2006). MaltParser uses the libsvm classification engine. We used the pre-trained English models (based on sections 0-22 of the Penn WSJ) supplied with MaltParser. MaltParser already uses an effective *Classifiers Splitting* heuristic when training these models, setting a high baseline for our method. The pre-trained parser consists of hundreds of different classifiers, some very small. We report here on actual memory requirement and parsing time for sections 23-24, considering the classifier combination. We took rare features to be those appearing in less than 0.5% of the support vectors, which leaves us with less than 300 common features in each of the "big" classifiers. The results are summarized in Table 1. As can be seen, our method parses

Method	Mem.	Parsing Time	Sents/Sec
Libsvm	240MB	2166 (sec)	1.73
ThisPaper	750MB	70 (sec)	53

Table 1: Parsing Time for WSJ Sections 23-24 (3762 sentences), on Pentium M, 1.73GHz

about 30 times faster, while using only 3 times as much memory. MaltParser coupled with our fast classifier parses above 3200 sentences per minute.

5 Conclusions

We presented a method for fast, accurate and memory efficient calculation for polynomial kernels decisions functions in NLP application. While the method is applied to SVMs, it generalizes to other polynomial kernel based classifiers. We demonstrated the method on the MaltParser dependency parser with a 30-time speedup factor on overall parsing time, with low memory overhead.

References

Y. Goldberg and M. Elhadad. 2007. SVM model tampering and anchored learning: A case study in hebrew. np chunking. In *Proc. of ACL2007*.

H. Isozaki and H. Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proc. of COLING2002*.

T. Kudo and Y. Matsumoto. 2003. Fast methods for kernel-based text analysis. In *ACL-2003*.

J. Nivre, J. Hall, and J. Nillson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC2006*.

Y. Wu, J. Yang, and Y. Lee. 2007. An approximate approach for training polynomial kernel svms in linear time. In *Proc. of ACL2007 (short-paper)*.

Extracting a Representation from Text for Semantic Analysis

Rodney D. Nielsen[1,2], Wayne Ward[1,2], James H. Martin[1], and Martha Palmer[1]
[1] Center for Computational Language and Education Research, University of Colorado, Boulder
[2] Boulder Language Technologies, 2960 Center Green Ct., Boulder, CO 80301
Rodney.Nielsen, Wayne.Ward, James.Martin, Martha.Palmer@Colorado.edu

Abstract

We present a novel fine-grained semantic representation of text and an approach to constructing it. This representation is largely extractable by today's technologies and facilitates more detailed semantic analysis. We discuss the requirements driving the representation, suggest how it might be of value in the automated tutoring domain, and provide evidence of its validity.

1 Introduction

This paper presents a new semantic representation intended to allow more detailed assessment of student responses to questions from an intelligent tutoring system (ITS). Assessment within current ITSs generally provides little more than an indication that the student's response expressed the target knowledge or it did not. Furthermore, virtually all ITSs are developed in a very domain-specific way, with each new question requiring the handcrafting of new semantic extraction frames, parsers, logic representations, or knowledge-based ontologies (c.f., Jordan et al., 2004). This is also true of research in the area of scoring constructed response questions (e.g., Leacock, 2004).

The goal of the representation described here is to facilitate domain-independent assessment of student responses to questions in the context of a known reference answer and to perform this assessment at a level of detail that will enable more effective ITS dialog. We have two key criteria for this representation: 1) it must be at a level that facilitates detailed assessment of the learner's understanding, indicating exactly *where* and *in what manner* the answer did not meet expectations and 2) the representation and assessment should be *learnable* by an automated system – they should not require the handcrafting of domain-specific representations of any kind.

Rather than have a single expressed versus unexpressed assessment of the reference answer as a whole, we instead break the reference answer down into what we consider to be approximately its lowest level compositional facets. This roughly translates to the set of triples composed of labeled (typed) dependencies in a dependency parse of the reference answer. Breaking the reference answer down into fine-grained facets permits a more focused assessment of the student's response, but a simple yes or no entailment at the facet level still lacks semantic expressiveness with regard to the relation between the student's answer and the facet in question, (e.g., did the student contradict the facet or completely fail to address it?) Therefore, it is also necessary to break the annotation labels into finer levels in order to specify more clearly the relationship between the student's answer and the reference answer facet. The emphasis of this paper is on this fine-grained facet-based representation – considerations in defining it, the process of extracting it, and the benefit of using it.

2 Representing the Target Knowledge

We acquired grade 3-6 responses to 287 questions from the Assessing Science Knowledge (ASK) project (Lawrence Hall of Science, 2006). The responses, which range in length from moderately short verb phrases to several sentences, cover all 16 diverse Full Option Science System teaching and learning modules spanning life science, physical science, earth and space science, scientific reasoning, and technology. We generated a corpus by transcribing a random sample (approx. 15400) of the students' handwritten responses.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 241–244,
Columbus, Ohio, USA, June 2008.

2.1 Knowledge Representation

The ASK assessments included a reference answer for each constructed response question. These reference answers were manually decomposed into fine-grained facets, roughly extracted from the relations in a syntactic dependency parse and a shallow semantic parse. The decomposition is based closely on these well-established frameworks, since the representations have been shown to be learnable by automatic systems (c.f., Gildea and Jurafsky, 2002; Nivre et al., 2006).

Figure 1 illustrates the process of deriving the constituent facets that comprise the representation of the final reference answer. We begin by determining the dependency parse following the style of MaltParser (Nivre et al., 2006). This dependency parse was then modified in several ways. The rationale for the modifications, which we elaborate below, is to increase the semantic content of facets. These more expressive facets are used later to generate features for the assessment classification task. These types of modifications to the parser output address known limitations of current statistical parser outputs, and are reminiscent of the modifications advocated by Briscoe and Carroll for more effective parser evaluation, (Briscoe, et. al, 2002). Example 1 illustrates the reference answer facets derived from the final dependencies in Figure 1, along with their glosses.

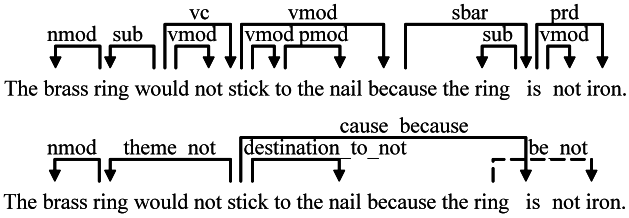


Figure 1. Reference answer representation revisions

(1) The brass ring would not stick to the nail because the ring is not iron.
(1a) NMod(ring, brass)
(1a') The ring is brass.
(1b) Theme_not(stick, ring)
(1b') The ring does not stick.
(1c) Destination_to_not(stick, nail)
(1c') Something does not stick to the nail.
(1d) Be_not(ring, iron)
(1d') The ring is not iron.
(1e) Cause_because(1b-c, 1d)
(1e') 1b and 1c are caused by 1d.

Various linguistic theories take a different stance on what term should be the governor in a number of phrase types, particularly noun phrases. In this regard, the manual parses here varied from the style of MaltParser by raising lexical items to governor status when they contextually carried more significant semantics. In our example, the verb *stick* is made the governor of *would*, whose modifiers are reattached to *stick*. Similarly, the noun phrases *the pattern of pigments* and *the bunch of leaves* typically result in identical dependency parses. However, the word *pattern* is considered the governor of *pigments*; whereas, conversely the word *leaves* is treated as the governor of *bunch* because it carries more semantics. Then, terms that were not crucial to the student answer, frequently auxiliary verbs, were removed (e.g., the modal *would* and determiners in our example).

Next, we incorporate prepositions into the dependency type labels following (Lin and Pantel, 2001). This results in the two dependencies vmod(*stick, to*) and pmod(*to, nail*), each of which carries little semantic value over its key lexical item, *stick* and *nail*, being combined into the single, more expressive dependency vmod_to(*stick, nail*), ultimately vmod is replaced with destination, as described below. Likewise, the dependencies connected by *because* are consolidated and *because* is integrated into the new dependency type.

Next, copulas and a few similar verbs are also incorporated into the dependency types. The verb's predicate is reattached to its subject, which becomes the governor, and the dependency is labeled with the verb's root. In our example, the two semantically impoverished dependencies sub(*is, ring*) and prd(*is, iron*) are combined to form the more meaningful dependency be(*ring, iron*). Then terms of negation are similarly incorporated into the dependency types.

Finally, wherever a shallow semantic parse would identify a predicate argument structure, we used the thematic role labels in VerbNet (Kipper et al., 2000) between the predicate and the argument's headword, rather than the MaltParser dependency tags. This also involved adding new structural dependencies that a typical dependency parser would not generate. For example, in the sentence *As it freezes the water will expand and crack the glass*, typically the dependency between *crack* and its subject *water* is not generated since it would lead to a non-projective tree, but it does play the role of Agent in a semantic parse. In a small number of instances, these labels were also at-

tached to noun modifiers, most notably the Location label. For example, given the reference answer fragment *The water on the floor had a much larger surface area*, one of the facets extracted was Location_on(*water, floor*).

We refer to facets that express relations between higher-level propositions as inter-propositional facets. An example of such a facet is (1e) above, connecting the proposition *the brass ring did not stick to the nail* to the proposition *the ring is not iron*. In addition to specifying the headwords of inter-propositional facets (*stick* and *is*, in 1e), we also note up to two key facets from each of the propositions that the relation is connecting (b, c, and d in example 1). Reference answer facets that are assumed to be understood by the learner a priori, (e.g., because they are part of the question), are also annotated to indicate this.

There were a total of 2878 reference answer facets, resulting in a mean of 10 facets per answer (median 8). Facets that were assumed to be understood a priori by students accounted for 33% of all facets and inter-propositional facets accounted for 11%. The results of automated annotation of student answers (section 3) focus on the facets that are not assumed to be understood a priori (67% of all facets); of these, 12% are inter-propositional.

A total of 36 different facet relation types were utilized. The majority, 21, are VerbNet thematic roles. Direction, Manner, and Purpose are PropBank adjunctive argument labels (Palmer et al., 2005). Quantifier, Means, Cause-to-Know and copulas were added to the preceding roles. Finally, anything that did not fit into the above categories retained its dependency parse type: VMod (Verb Modifier), NMod (Noun Modifier), AMod (Adjective or Adverb Modifier), and Root (Root was used when a single word in the answer, typically yes, no, agree, disagree, A-D, etc., stood alone without a significant relation to the remainder of the reference answer; this occurred only 21 times, accounting for fewer than 1% of the reference answer facets). The seven highest frequency relations are NMod, Theme, Cause, Be, Patient, AMod, and Location, which together account for 70% of the reference answer facet relations

2.2 Student Answer Annotation

For each student answer, we annotated each reference answer facet to indicate whether and how the student addressed that facet. We settled on the five annotation categories in Table 1. These labels and the annotation process are detailed in (Nielsen et al., 2008b).

Understood: Reference answer facets directly expressed or whose understanding is inferred
Contradiction: Reference answer facets contradicted by negation, antonymous expressions, pragmatics, etc.
Self-Contra: Reference answer facets that are both contradicted and implied (self contradictions)
Diff-Arg: Reference answer facets whose core relation is expressed, but it has a different modifier or argument
Unaddressed: Reference answer facets that are not addressed at all by the student's answer

Table 1. Facet Annotation Labels

3 Automated Classification

As partial validation of this knowledge representation, we present results of an automatic assessment of our student answers. We start with the hand generated reference answer facets. We generate automatic parses for the reference answers and the student answers and automatically modify these parses to match our desired representation. Then for each reference answer facet, we extract features indicative of the student's understanding of that facet. Finally, we train a machine learning classifier on training data and use it to classify unseen test examples, assigning a Table 1 label for each reference answer facet.

We used a variety of linguistic features that assess the facets' similarity via lexical entailment probabilities following (Glickman et al., 2005), part of speech tags and lexical stem matches. They include information extracted from modified dependency parses such as relevant relation types and path edit distances. Revised dependency parses are used to align the terms and facet-level information for feature extraction. Remaining details can be found in (Nielsen et al., 2008a) and are not central to the semantic representation focus of this paper. Current classification accuracy, assigning a Table 1 label to each reference answer facet to indicate the student's expressed understanding, is 79% within domain (assessing unseen answers to questions associated with the training data) and 69% out of domain (assessing answers to questions regarding entirely different science subjects). These results are 26% and 15% over the majority class baselines, respectively, and 21% and 6% over lexi-

cal entailment baselines based on Glickman et al. (2005).

4 Discussion and Future Work

Analyzing the results of reference facet extraction, there are many interesting open linguistic issues in this area. This includes the need for a more sophisticated treatment of adjectives, conjunctions, plurals and quantifiers, all of which are known to be beyond the abilities of state of the art parsers.

Analyzing the dependency parses of 51 of the student answers, about 24% had errors that could easily lead to problems in assessment. Over half of these errors resulted from inopportune sentence segmentation due to run-on student sentences conjoined by *and* (e.g., the parse of *a shorter string makes a higher pitch and a longer string makes a lower pitch*, errantly conjoined *a higher pitch* and *a longer string* as the subject of *makes a lower pitch*, leaving *a shorter string makes* without an object). We are working on approaches to mitigate this problem.

In the long term, when the ITS generates its own questions and reference answers, the system will have to construct its own reference answer facets. The automatic construction of reference answer facets must deal with all of the issues described in this paper and is a significant area of future research. Other key areas of future research involve integrating the representation described here into an ITS and evaluating its impact.

5 Conclusion

We presented a novel fine-grained semantic representation and evaluated it in the context of automated tutoring. A significant contribution of this representation is that it will facilitate more precise tutor feedback, targeted to the specific facet of the reference answer and pertaining to the specific level of understanding expressed by the student. This representation could also be useful in areas such as question answering or document summarization, where a series of entailed facets could be composed to form a full answer or summary.

The representation's validity is partially demonstrated in the ability of annotators to reliably annotate inferences at this facet level, achieving substantial agreement (86%, Kappa=0.72) and by promising results in automatic assessment of student answers at this facet level (up to 26% over baseline), particularly given that, in addition to the manual reference answer facet representation, an automatically extracted approximation of the representation was a key factor in the features utilized by the classifier.

The domain independent approach described here enables systems that can easily scale up to new content and learning environments, avoiding the need for lesson planners or technologists to create extensive new rules or classifiers for each new question the system must handle. This is an obligatory first step to the long-term goal of creating ITSs that can truly engage children in natural unrestricted dialog, such as is required to perform high quality student directed Socratic tutoring.

Acknowledgments

This work was partially funded by Award Number 0551723 from the National Science Foundation.

References

Briscoe, E., Carroll, J., Graham, J., and Copestake, A. 2002. Relational evaluation schemes. In *Proc. of the Beyond PARSEVAL Workshop at LREC*.

Gildea, D. and Jurafsky, D. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Glickman, O, Dagan, I, and Koppel, M. 2005. Web Based Probabilistic Textual Entailment. In *Proc RTE*.

Jordan, P, Makatchev, M, VanLehn, K. 2004. Combining competing language understanding approaches in an intelligent tutoring system. In *Proc ITS*.

Kipper, K, Dang, H, and Palmer, M. 2000. Class-Based Construction of a Verb Lexicon. In *Proc. AAAI*.

Lawrence Hall of Science 2006. Assessing Science Knowledge (ASK), UC Berkeley, NSF-0242510

Leacock, C. 2004. Scoring free-response automatically: A case study of a large-scale Assessment. *Examens*.

Lin, D & Pantel, P. 2001. Discovery of inference rules for Question Answering. In *Natl. Lang. Engineering*.

Nielsen, R, Ward, W, and Martin, JH. 2008a. Learning to Assess Low-level Conceptual Understanding. In *Proc. FLAIRS*.

Nielsen, R, Ward, W, Martin, JH and Palmer, P. 2008b. Annotating Students' Understanding of Science Concepts. In *Proc. LREC*.

Nivre, J, Hall, J, Nilsson, J, Eryigit, G and Marinov, S. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proc. CoNLL*.

Palmer, M, Gildea, D, & Kingsbury, P. 2005. The proposition bank: An annotated corpus of semantic roles. In *Computational Linguistics*.

Efficient Processing of Underspecified Discourse Representations

Michaela Regneri[†] [*]
regneri@coli.uni-sb.de
[*] Saarland University

Markus Egg[†]
egg@let.rug.nl
[†] University of Groningen

Alexander Koller[§]
a.koller@ed.ac.uk
[§] University of Edinburgh

Abstract

Underspecification-based algorithms for processing partially disambiguated discourse structure must cope with extremely high numbers of readings. Based on previous work on dominance graphs and weighted tree grammars, we provide the first possibility for computing an underspecified discourse description and a best discourse representation efficiently enough to process even the longest discourses in the RST Discourse Treebank.

1 Introduction

Discourse processing has emerged as a highly relevant source of information for applications such as information extraction and automatic summarisation (Taboada and Mann (2006) outline this and further applications). But discourse structures cannot always be described completely, either due to genuine ambiguity (Stede, 2004) or to the limitations of a discourse parser. In either case, only partial information on discourse structure is available. To handle such information, *underspecification* formalisms can be used. Underspecification was originally introduced in computational semantics to model structural ambiguity without disjunctively enumerating the readings, and later applied to discourse parsing (Gardent and Webber, 1998; Schilder, 2002).

However, while the existing algorithms for underspecification processing work well for semantic structures, they were not designed for discourse structures, which can be much larger. Indeed, it has never been shown that underspecified discourse reprentations (UDRs) can be processed efficiently, since the general-purpose implementations are too slow for that task.

In this paper, we present a new way to implement and process discourse underspecification in terms of *regular tree grammars* (RTGs). RTGs are used as an underspecification formalism in semantics (Koller et al., 2008). We show how to compute RTGs for discourse from dominance-based underspecified representations more efficiently (by a typical factor of 100) than before. Furthermore, we show how weighted RTGs can be used to represent constraints and preferences on the discourse structure. Taking all these results together, we show for the first time how the globally optimal discourse representation based on some preference model can be computed efficiently from an UDR.

2 Underspecified Discourse Representation

Following annotation schemes like the one of Stede (2004), we model discourse structures by binary trees. Fig. (1b-f) represent the potential structures of (1). We write each elementary discourse unit (EDU) in square brackets.

(1) [C_1 I try to read a novel] [C_2 if I feel bored] [C_3 because the TV programs disappoint me] [C_4 but I can't concentrate on anything.]

Underspecification formalisms such as *dominance graphs* (Althaus et al., 2003) can model partial information about such trees; see Fig. (1a) for the underspecified discourse representation (UDR) of (1). These graphs consist of labelled roots and unlabelled holes; the solid edges indicate that a node must be the parent of another, and the dashed edges indicate (transitive) dominance requirements. A *configuration* of a dominance graph is an arrangement of the (labelled) graph nodes into a tree that satisfies all (immediate and transitive) dominance requirements. Subgraphs that are connected by solid edges are called *fragments* and must be tree-shaped.

Using UDRs, discourse parsing can be modularised into three separate steps. First, a discourse parser segments the text and generates an UDR from it. The node labels in the UDR aren't necessarily fully specified (Egg and Redeker, 2007; Schilder,

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 245–248,
Columbus, Ohio, USA, June 2008.

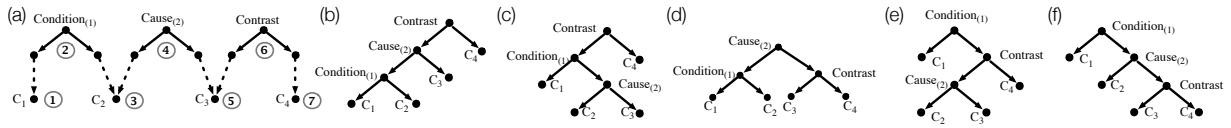


Figure 1: An underspecified discourse structure and its five configurations

2002); here we pretend that they are to simplify the presentation, as nothing in this paper hinges on it. Then weights are added to the UDR that incorporate preferences for discourse structures based on linguistic cues. Finally, the weighted UDR can either be processed directly by other applications, or, if a tree structure is required, we can compute the best configuration. In this paper, we show how an UDR dominance graph can be converted into a regular tree grammar efficiently. This simplifies the specification of weights in Step 2; we also show how to efficiently compute a best tree from a weighted RTG (Step 3). We do not discuss Step 1 in this paper.

The dominance graphs used in discourse underspecification are *constrained chains*. A constrained chain of length n consists of n *upper* fragments with two holes each and $n+1$ *lower* fragments with no holes. There must also be a numbering $1,\ldots,2n+1$ of the fragments such that for every $1 \leq i \leq n$, fragment $2i$ is an upper fragment, fragments $2i-1$ and $2i+1$ are lower fragments, and there are dominance edges from the left hole of $2i$ to the root of $2i-1$ and from the right hole of $2i$ to the root of $2i+1$ (and possibly further dominance edges). These numbers are shown in circles in Fig. (1a). In discourse dominance graphs, upper fragments correspond to discourse relations, and lower fragments correspond to EDUs; the EDUs are ordered according to their appearance in the text, and the upper fragments connect the two text spans to which they are adjacent.

3 Underspecified Processing for Discourses

Recently, Koller et al. (2008) showed how to process dominance graphs with *regular tree grammars* (Comon et al., 2007, RTGs). RTGs are a grammar formalism that describes sets of trees using production rules which rewrite non-terminal symbols (NTs) into terms consisting of tree constructors and possibly further NTs. A tree (without NTs) is accepted by the grammar if it can be derived by a sequence of rule applications from a given start symbol. An example RTG is shown in Fig. 2; its start symbol is $\{1;7\}$, and it describes exactly the five trees in

$$
\begin{array}{l|l}
\{1;7\} \to \mathit{Cond}(\{1\},\{3;7\})\ [1] & \{5;7\} \to \mathit{Contr}(\{5\},\{7\})\ [1] \\
\{3;7\} \to \mathit{Contr}(\{3;5\},\{7\})\ [1] & \{3;5\} \to \mathit{Cause}(\{3\},\{5\})\ [1] \\
\{1;7\} \to \mathit{Contr}(\{1;5\},\{7\})\ [1] & \{1;3\} \to \mathit{Cond}(\{1\},\{3\})\ [5] \\
\{1;7\} \to \mathit{Cause}(\{1;3\},\{5;7\})\ [1] & \{1;5\} \to \mathit{Cond}(\{1\},\{3;5\})\ [3] \\
\{1;5\} \to \mathit{Cause}(\{1;3\},\{5\})\ [1] & \{3;7\} \to \mathit{Cause}(\{3\},\{5;7\})\ [1] \\
\{1\} \to C_1\ [1] & \{5\} \to C_3\ [1] \quad \{7\} \to C_4\ [1] \\
\{3\} \to C_2\ [1] &
\end{array}
$$

Figure 2: A wRTG modelling Fig. 1

Fig. (1b-f). For example, Fig. (1e) is derived by expanding the start symbol with the first rule in Fig. 2. This determines that the tree root is labelled with *Condition*; we then derive the left subtree from the NT $\{1\}$ and the right subtree from the NT $\{3;7\}$.

The NTs in the grammar correspond to subgraphs in the dominance graph: The NT $\{1;7\}$ represents the subgraph $\{1,2,3,4,5,6,7\}$ (i.e. the whole graph); the NT $\{1\}$ represents the subgraph containing only the fragment 1; and so forth. The trees that can be derived from each nonterminal correspond exactly to the configurations of the subgraph.

Koller and Thater (2005b) presented an algorithm for generating, from a very general class of dominance graphs, an RTG that describes exactly the same trees. For each subgraph S that is to be the LHS of a rule, the algorithm determines the *free* fragments of S, i.e. the fragments that may serve as the root of one of its configurations, by a certain graph algorithm. For every free fragment in S with n holes and a root label f, the algorithm generates a new rule of the form $S \to f(S_1,\ldots,S_n)$, where each S_i corresponds to the remaining subgraph under the i-th hole. The procedure calls itself recursively on the subgraphs until it reaches singleton subgraphs.

While this algorithm works well with underspecified semantic representations in semantics, it is too slow for the larger discourse graphs, as we will see in Section 5. However, we will now optimise it for the special case of constrained chains. First, we observe that all subgraphs ever visited by the algorithm are connected subchains. A subchain is uniquely identifiable by the positions of the first and last fragment in the left-to-right order of the chain; we can thus read the nonterminal $\{i;j\}$ simply as a pair of integers that identifies the subchain from the i-th to the

Algorithm 1: GenerateRules($\{i; j\}$,G,C)

```
1  if G contains rules for {i; j} then return
2  if i=j then G.add({ {i; j} → Label(i) } ) else
   /* Loop over upper fragments                    */
3  for k = i+1 to j-1 step 2 do
4      if ¬∃ edge=(s,t) ∈ C s.t. (i ≤ s < k ≤ t ≤ j) ∨ (i ≤ t
       ≤ k < s ≤ j) then
5          lSub ← {i;k-1}, rSub ← {k+1; j}
6          G.add({i; j} → Label(i)(lSub, rSub))
7          GenerateRules(lSub, G, C)
8          GenerateRules(rSub, G, C)
```

j-th fragment (rather than an abbreviation for a set of fragments). i and j will generally represent lower fragments. In the grammar in Fig. 2, $\{i\}$ is an abbreviation of $\{i;i\}$.

We can now rephrase the Koller & Thater algorithm in our terms (Algorithm 1). The most important change is that we can now test whether an upper fragment k in a subgraph $\{i; j\}$ is free simply by checking whether there is no dominance edge from some upper fragment l to some upper fragment r such that $i \leq l < k \leq r \leq j$, and no dominance edge from r to l such that $i \leq l \leq k < r \leq j$. For instance, if there was a dominance edge from the right hole of 2 to the root of 6 in Fig. (1a), then 4 and 6 would not be free, but 2 would be; and indeed, all configurations of this graph would have to have 2 as their roots. Hence we can replace the graph algorithm for freeness by a simple comparison of integers. The general structure of the algorithm remains the same as in (Koller and Thater, 2005b): It takes a dominance graph C as its input, and recursively calls itself on pairs $\{i; j\}$ representing subgraphs while adding rules and NTs to an RTG G.

4 Soft Discourse Constraints

RTGs can be extended to *weighted* regular tree grammars (Knight and Graehl, 2005, wRTGs) by adding numeric weights to the rules. WRTG derivations assign weights to each tree: The weight of a tree is the product of the weights of all rules that were used in its derivation.

Egg and Regneri (2008) motivate the use of wRTGs in discourse processing. They assign rule weights based on corpus-extracted constraints which express the interdependencies between discourse relations and their surrounding tree structure. One such constraint states that the right subtree of a *Condition* node should be of minimal size, which ranks the readings of Fig. 1 (a): (b), (d) > (c) > (e), (f).

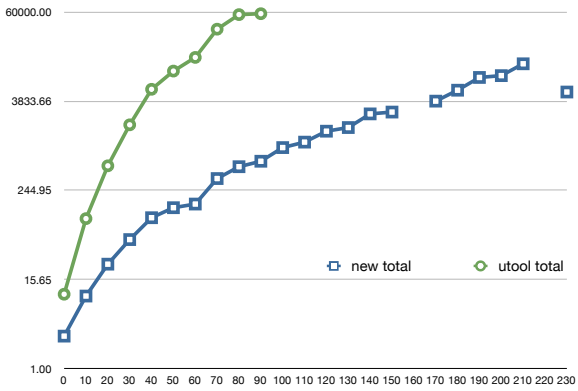


Figure 3: Runtime Comparison

In order to state this constraint in a wRTG, we annotate the grammar in Fig. 2 with the weights shown in brackets. The *Condition* rules get higher weights if the second NT on the RHS represents a smaller subgraph. The grammar assigns the maximum weight of 5 to (b) and (d) (fragment 2 has a leaf as right child), the medium weight 3 to (c) (the right subgraph of fragment 2 contains two EDUs), and the minimum weight 1 to (e) and (f). i.e. it ranks the readings as intended.

Based on our implementation of nonterminals as integer pairs, we can efficiently compute a configuration with maximal weight using a version of Knight and Graehl's (2005) algorithm for computing the best derivation of a wRTG that is specialised to the grammars we use.

5 Evaluation

We compare our runtimes with those of Utool (Koller and Thater, 2005a), the fasted known solver for general dominance graphs; it implements the Koller & Thater algorithm. Utool runs very fast for underspecified representations in semantics, but the representations for discourse parsing are considerably larger: The largest underspecified semantic representation found in the Rondane treebank analysed with the English Resource Grammar (Copestake and Flickinger, 2000, ERG) has 4.5×10^{12} structural scope readings, but for 59% of the discourses in the RST Discourse Treebank (Carlson et al., 2002, RST-DT), there are more ways of configuring all EDUs into a binary tree than that.

We evaluate the efficiency of our algorithm on 364 texts from the RST-DT, by converting each discourse

into a chain with one lower fragment for each EDU and one upper fragment labelled with each annotated discourse relation. We use our algorithm and Utool to generate the RTG from the chain, assign all soft constraints of Egg and Regneri (2008) to the grammar, and finally compute the best configuration according to this model. The evaluation results are shown in Fig. 3. The horizontal axis shows the chain length (= number of EDUs minus 1), rounded down to multiples of ten; the (logarithmic) vertical axis shows the average runtime in milliseconds for discourses of that length. Both algorithms spend a bit over half the runtime on computing the RTGs.

As the diagram shows, our algorithm is up to 100 times faster than Utool for the same discourses. It is capable of computing the best configuration for every tested discourse – in less than one second for 86% of the texts. Utool exceeded the OS memory limit on 77 discourses, and generally couldn't process any text with more than 100 EDUs. The longest text in the RST-DT has 304 EDUs, so the UDR has about 2.8×10^{178} different configurations. Our algorithm computes the best configuration for this UDR in about three minutes.

6 Conclusion

We presented the first solver for underspecified discourse representations that is efficient enough to compute the globally best configurations of every discourse in the RST discourse treebank, by exploiting the fact that UDRs are very large but obey very strong structural restrictions. Our solver converts a dominance graph into an RTG, adds weights to the RTG to represent discourse constraints, and then computes the globally optimal configuration.

It takes about three minutes to compute a best configuration with a given probability model for the longest discourse in the treebank, out of 10^{178} possible configurations. For comparison, an algorithm that enumerates a billion configurations per second to find the best one could have inspected only about 10^{26} within the estimated age of the universe. So our algorithm is useful and necessary to process real-world underspecified discourse representations.

We have thus demonstrated that discourse processing based on underspecification is computationally feasible. Nothing in our algorithm hinges on using RST in particular; it is compatible with any approach that uses binary trees. In future research, it would be interesting to complete our system into a full-blown discourse parser by adding a module that computes an UDR for a given text, and evaluate whether its ability to delay decisions about discourse structure would improve accuracy.

References

E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48:194–219.

L. Carlson, D. Marcu, and M. E. Okurowski. 2002. RST Discourse Treebank. LDC.

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree Automata Techniques and Applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. Release 12-10-2007.

A. Copestake and D. Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Conference on Language Resources and Evaluation*.

M. Egg and G. Redeker. 2007. Underspecified discourse representation. In A. Benz and P. Kühnlein, editors, *Constraints in Discourse*, Amsterdam. Benjamins.

M. Egg and M. Regneri. 2008. Underspecified Modelling of Complex Discourse Constraints. Submitted.

C. Gardent and B. Webber. 1998. Describing Discourse Semantics. In *Proceedings of the 4th TAG+ Workshop*, University of Pennsylvania, Philadelphia.

K. Knight and J. Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Computational linguistics and intelligent text processing*, pages 1–24. Springer.

A. Koller and S. Thater. 2005a. Efficient solving and exploration of scope ambiguities. Proceedings of the ACL-05 Demo Session.

A. Koller and S. Thater. 2005b. The evolution of dominance constraint solvers. In *Proceedings of the ACL-05 Workshop on Software*, Ann Arbor.

A. Koller, M. Regneri, and S. Thater. 2008. Regular tree grammars as a formalism for scope underspecification. In *Proceedings of ACL-08: HLT*.

F. Schilder. 2002. Robust discourse parsing via discourse markers, topicality and position. *Natural Language Engineering*, 8:235–255.

M. Stede. 2004. The Potsdam Commentary Corpus. In B. Webber and D. Byron, editors, *ACL-04 Workshop on Discourse Annotation*.

M. Taboada and W. Mann. 2006. Applications of Rhetorical Structure Theory. *Discourse Studies*, 8:567–588.

Choosing Sense Distinctions for WSD: Psycholinguistic Evidence

Susan Windisch Brown
Department of Linguistics
Institute of Cognitive Science
University of Colorado
Hellems 295 UCB
Boulder, CO 80309
susan.brown@colorado.edu

Abstract

Supervised word sense disambiguation requires training corpora that have been tagged with word senses, which begs the question of which word senses to tag with. The default choice has been WordNet, with its broad coverage and easy accessibility. However, concerns have been raised about the appropriateness of its fine-grained word senses for WSD. WSD systems have been far more successful in distinguishing coarse-grained senses than fine-grained ones (Navigli, 2006), but does that approach neglect necessary meaning differences? Recent psycholinguistic evidence seems to indicate that closely related word senses may be represented in the mental lexicon much like a single sense, whereas distantly related senses may be represented more like discrete entities. These results suggest that, for the purposes of WSD, closely related word senses can be clustered together into a more general sense with little meaning loss. The current paper will describe this psycholinguistic research and its implications for automatic word sense disambiguation.

1 Introduction*

The problem of creating a successful word sense disambiguation system begins, or should begin, well before methods or algorithms are considered. The first question should be, "Which senses do we want to be able to distinguish?" Dictionaries encourage us to consider words as having a discrete set of senses, yet any comparison between dictionaries quickly reveals how differently a word's meaning can be divided into separate senses. Rather than having a finite list of senses, many words seem to have senses that shade from one into another.

One could assume that dictionaries make broadly similar divisions and the exact point of division is only a minor detail. Simply picking one resource and sticking with it should solve the problem. In fact, WordNet, with its broad coverage and easy accessibility, has become the resource of choice for WSD. However, some have questioned whether WordNet's fine-grained sense distinctions are appropriate for the task (Ide & Wilks, 2007; Palmer et al., 2007). Some are concerned about feasibility: Is WSD at this level an unattainable goal? Others with practicality: Is this level of detail really needed for most NLP tasks, such as machine translation or question-answering? Finally, some wonder whether such fine-grained distinctions even reflect how human beings represent word meaning.

Human annotators have trouble distinguishing such fine-grained senses reliably. Interannotator agreement with WordNet senses is around 70% (Snyder & Palmer, 2004; Chklovski & Mihalcea, 2002), and it's understandable that WSD systems would have difficulty surpassing this upper bound.

Researchers have responded to these concerns by developing various ways to cluster WordNet senses. Mihalcea & Moldovan (2001) created an unsupervised approach that uses rules to cluster senses. Navigli (2006) has induced clusters by mapping WordNet senses to a more coarse-grained lexical resource. OntoNotes (Hovy et al., 2006) is manually grouping WordNet senses and creating a corpus tagged with these sense groups. Using On-

* I gratefully acknowledge the support of the National Science Foundation Grant NSF-0415923, Word Sense Disambiguation.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 249–252,
Columbus, Ohio, USA, June 2008.

toNotes and another set of manually tagged data, Snow et al. (2007) have developed a supervised method of clustering WordNet senses.

Although ITA rates and system performance both significantly improve with coarse-grained senses (Duffield et al., 2007; Navigli, 2006), the question about what level of granularity is needed remains. Palmer et al. (2007) state, "If too much information is being lost by failing to make the more fine-grained distinctions, the [sense] groups will avail us little."

Ides and Wilks (2007) drew on psycholinguistic research to help establish an appropriate level of sense granularity. However, there is no consensus in the psycholinguistics field on how lexical meaning is represented in the mind (Klein & Murphy, 2001; Pylkkänen et al., 2006; Rodd et al., 2002), and, as the Ide and Wilks (2007) state, "research in this area has been focused on developing psychological models of language processing and has not directly addressed the problem of identifying senses that are distinct enough to warrant, in psychological terms, a separate representation in the mental lexicon."

Our experiment looked directly at sense distinctions of varying degrees of meaning relatedness and found indications that the mental lexicon does not consist of separate representations of discrete senses for each word. Rather, word senses may share a greater or smaller portion of a semantic representation depending on the how closely related the senses are. Because closely related senses may share a large portion of their semantic representation, clustering such senses together would result in very little meaning loss. The remainder of this paper will describe the experiment and its implications for WSD in more detail.

2 Experiment

The goal of this experiment was to determine whether each sense of a word has a completely separate mental representation or not. If so, we also hoped to discover what types of sense distinctions seem to have separate mental representations.

2.1 Materials

Four groups of materials were prepared using the fine-grained sense distinctions found in WordNet 2.1. Each group consisted of 11 pairs of phrases. The groups comprised (1) homonymy, (2) distantly related senses, (3) closely related senses, and (4) same senses (see Table 1 for examples). Placement in these groups depended both on the classification of the usages by WordNet and the Oxford English Dictionary and on the ratings given to pairs of phrases by a group of undergraduates. They rated the relatedness of the verb in each pair on a scale of 0 to 3, with 0 being completely unrelated and 3 being the same sense.

A pair was considered to represent the same sense if the usage of the verb in both phrases was categorized by WordNet as the same and if the pair received a rating greater than 2.7. Closely related senses were listed as separate senses by WordNet and received a rating between 1.8 and 2.5. Distantly related senses were listed as separate senses by WordNet and received ratings between 0.7 and 1.3. Because WordNet makes no distinction between related and unrelated senses, the Oxford English Dictionary was used to classify homonyms. Homonyms were listed as such by the OED and received ratings under 0.3.

	Prime	Target
Unrelated	banked the plane	banked the money
Distantly related	ran the track	ran the shop
Closely related	broke the glass	broke the radio
Same sense	cleaned the shirt	cleaned the cup

Table 1. Stimuli.

2.2 Method

The experiment used a semantic decision task (Klein & Murphy, 2001; Pylkkänen et al., 2006), in which people were asked to judge whether short phrases "made sense" or not. Subjects saw a phrase, such as "posted the guard," and would decide whether the phrase made sense as quickly and as accurately as possible. They would then see another phrase with the same verb, such as "posted the letter," and respond to that phrase as well. The response time and accuracy were recorded for the second phrase of each pair.

2.3 Results and Discussion

When comparing response times between same sense pairs and different sense pairs (a combina-

tion of closely related, distantly related, and unrelated senses), we found a reliable difference (same sense mean: 1056ms, different sense mean: 1272ms; t_{32} =6.33; p<.0001). We also found better accuracy for same sense pairs (same sense: 95.6% correct vs. different sense: 78% correct; t_{32}=7.49; p<.0001). When moving from one phrase to another with the same meaning, subjects were faster and more accurate than when moving to a phrase with a different sense of the verb.

By itself, this result would fit with the theory that every sense of a word has a separate semantic representation. One would expect people to access the meaning of a verb quickly if they had just seen the verb used with that same meaning. One could think of the meaning as already having been "activated" by the first phrase. Accessing a completely different semantic representation when moving from one sense to another should be slower.

If all senses have separate representations, access to meaning should proceed in the same way for all. For example, if one is primed with the phrase "fixed the radio," response time and accuracy should be the same whether the target is "fixed the vase" or "fixed the date." Instead, we found a significant difference between these two groups, with closely related pairs accessed, on average, 173ms more quickly than the mean of the distantly and unrelated pairs (t_{32}=5.85; p<.0005), and accuracy was higher (91% vs. 72%; t_{32}=8.65; p<.0001).

A distinction between distantly related pairs and homonyms was found as well. Response times for distantly related pairs was faster than for homonyms (distantly related mean: 1253ms, homonym mean: 1406ms; t_{32}=2.38; p<.0001). Accuracy was enhanced as well for this group (distantly related mean: 81%, unrelated mean: 62%; t_{32}=5.66; p<.0001). Related meanings, even distantly related, seem to be easier to access than unrelated meanings.

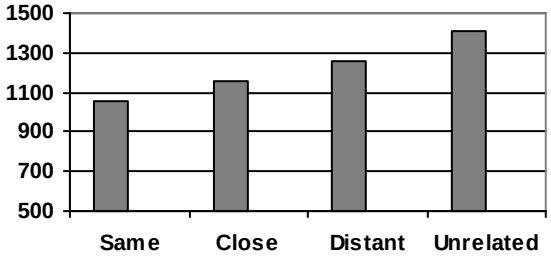


Figure 1. Mean response time (ms).

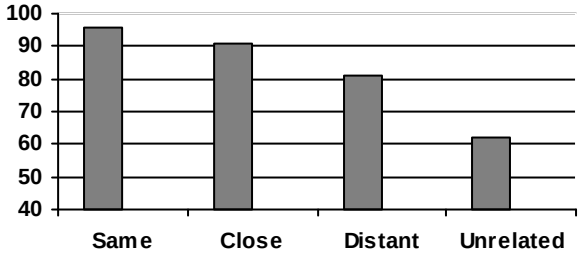


Figure 2. Mean accuracy (% correct).

A final planned comparison tested for a linear progression through the test conditions. Although somewhat redundant with the other comparisons, this test did reveal a highly significant linear progression for response time ($F_{1,32}$=95.8; p<.0001) and for accuracy ($F_{1,32}$=100.1; p<.0001).

People have an increasingly difficult time accessing the meaning of a word as the relatedness of the meaning in the first phrase grows more distant. They respond more slowly and their accuracy declines. However, closely related senses are almost as easy to access as same sense phrases. These results suggest that closely related word senses may be represented in the mental lexicon much like a single sense, perhaps sharing a core semantic representation.

The linear progression through meaning relatedness is also compatible with a theory in which the semantic representations of related senses overlap. Rather than being discrete entities attached to a main "entry", they could share a general semantic space. Various portions of the space could be activated depending on the context in which the word occurs. This structure allows for more coarse-grained or more fine-grained distinctions to be made, depending on the needs of the moment.

A structure in which the semantic representations overlap allows for the apparently smooth progression from same sense usages to more and more distantly related usages. It also provides a simple explanation for semantically underdetermined usages of a word. Although separate senses of a word can be identified in different contexts, in some contexts, both senses (or a vague meaning indeterminate between the two) seem to be represented by the same word. For example, "newspaper" can refer to a physical object: "He tore the newspaper in half", or to the content of a publication: "The newspaper made me mad today, suggesting that our committee is corrupt." The sen-

tence “I really like this newspaper” makes no commitment to either sense.

.

3 Conclusions

What does this mean for WSD? Most would agree that NLP applications would benefit from the ability to distinguish homonym-level meaning differences. Similarly, most would agree that it is not necessary to make very fine distinctions, even if we can describe them. For example, the process of cleaning a cup is discernibly different from the process of cleaning a shirt, yet we would not want to have a WSD system try to distinguish between every minor variation on cleaning. The problem comes with deciding when meanings can be considered the same sense, and when they should be considered different.

The results of this study suggest that some word usages considered different by WordNet provoke similar responses as those to same sense usages. If these usages activate the same or largely overlapping meaning representations, it seems safe to assume that little meaning loss would result from clustering these closely related senses into one more general sense. Conversely, people reacted to distantly related senses much as they did to homonyms, suggesting that making distinctions between these usages would be useful in a WSD system.

A closer analysis of the study materials reveals differences between the types of distinctions made in the closely related senses and the types made in the distantly related senses. Most of the closely related senses distinguished between different concrete usages, whereas the distantly related senses distinguished between a concrete usage and a figurative or metaphorical usage. This suggests that grouping concrete usages together may result in little, if any, meaning loss. It may be more important to keep concrete senses distinct from figurative or metaphorical senses. The present study, however, divided senses only on degree of relatedness rather than type of relatedness. It would be useful in future studies to address more directly the question of distinctions based on concreteness, animacy, agency, and so on.

References

Chklovski, Tim, and Rada Mihalcea. 2002. Building a sense tagged corpus with open mind word expert. *Proc. of ACL 2002 Workshop on WSD: Recent Successes and Future Directions*. Philadelphia, PA.

Duffield, Cecily Jill, Jena D. Hwang, Susan Windisch Brown, Dmitriy Dligach, Sarah E.Vieweg, Jenny Davis, Martha Palmer. 2007. Criteria for the manual grouping of verb senses. *Linguistics Annotation Workshop, ACL-2007*. Prague, Czech Republic.

Hovy, Eduard, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. *Proc. of HLT-NAACL 2006*. New York, NY.

Ide, Nancy, and Yorick Wilks. 2007. Making sense about sense. In *Word Sense Disambiguation: Algorithms and Applications*, E. Agirre and P. Edmonds (eds.). Dordrecht, The Netherlands: Springer.

Klein, D., and Murphy, G. (2001). The representation of polysemous words. *J of Memory and Language 45*, 259-282.

Mihalcea, Rada, and Dan I. Moldovan. 2001. Automatic generation of a coarse-grained WordNet. In *Proc. of NAACL Workshop on WordNet and Other Lexical Resources*. Pittsburg, PA.

Navigli, Roberto. 2006. Meaningful clustering of word senses helps boost word sense disambiguation performance. *Proc. of the 21st International Conference on Computational Linguistics.* Sydney, Australia.

Palmer, Martha, Hwee Tou Ng, and Hoa Trang Dang. 2007. Evaluation of WSD systems. In *Word Sense Disambiguation: Algorithms and Applications*, E. Agirre and P. Edmonds (eds.). Dordrecht, The Netherlands: Springer.

Pylkkänen, L., Llinás, R., and Murphy, G. L. (2006). The representation of polysemy: MEG evidence. *J of Cognitive Neuroscience 18*, 97-109.

Rodd, J., Gaskell, G., and Marslen-Wilson, W. (2002). Making sense of semantic ambiguity: Semantic competition in lexical access. *J. of Memory and Language, 46*, 245-266.

Snow, Rion, Sushant Prakash, Dan Jurafsky and Andrew Y. Ng. 2007. Learning to merge word senses. *Proc. of EMNLP 2007*. Prague, Czech Republic.

Snyder, Benjamin, and Martha Palmer. 2004. The English all-words task. *Proc. of ACL 2004 SENSEVAL-3 Workshop*. Barcelona, Spain.

Decompounding query keywords from compounding languages

Enrique Alfonseca
Google Inc.
ealfonseca@google.com

Slaven Bilac
Google Inc.
slaven@google.com

Stefan Pharies
Google Inc.
stefanp@google.com

Abstract

Splitting compound words has proved to be useful in areas such as Machine Translation, Speech Recognition or Information Retrieval (IR). Furthermore, real-time IR systems (such as search engines) need to cope with noisy data, as user queries are sometimes written quickly and submitted without review. In this paper we apply a state-of-the-art procedure for German decompounding to other compounding languages, and we show that it is possible to have a single decompounding model that is applicable across languages.

1 Introduction

Compounding languages (Krott, 1999), such as German, Dutch, Danish, Norwegian, Swedish, Greek or Finnish, allow the generation of complex words by merging together simpler ones. So, for instance, the *flower bouquet* can be expressed in German as *Blumensträuße*, made up of *Blumen* (flower) and *sträuße* (bouquet), and in Finnish as *kukkakimppu*, from *kukka* (flower) and *kimppu* (bunch, collection). For many language processing tools that rely on lexicons or language models it is very useful to be able to decompose compounds to increase their coverage and reduce out-of-vocabulary terms. Decompounders have been used successfully in Information Retrieval (Braschler and Ripplinger, 2004), Machine Translation (Brown, 2002; Koehn and Knight, 2003) and Speech Recognition (Adda-Decker et al., 2000). The Cross Language Evaluation Forum (CLEF) competitions have shown that very simple approaches can produce big gains in Cross Language Information Retrieval (CLIR) for German and Dutch (Monz and de Rijke, 2001) and for Finnish (Adafre et al., 2004).

When working with web data, which has not necessarily been reviewed for correctness, many of the words are more difficult to analyze than when working with standard texts. There are more words with spelling mistakes, and many texts mix words from different languages. This problem exists to a larger degree when handling user queries: they are written quickly, not paying attention to mistakes. However, being able to identify that *achzigerjahre* should be decompounded as *achzig+jahre* (where *achzig* is a misspelled variation of *achtzig*) is still useful in obtaining some meaning from the user query and in helping the spelling correction system. This paper evaluates a state-of-the-art procedure for German splitting (Alfonseca et al., 2008), robust enough to handle query data, on different languages, and shows that it is possible to have a single decompounding model that can be applied to all the languages under study.

2 Problem definition and evaluation settings

Any set of query keywords contains a large amount of noisy data, such as words in foreign languages or misspelled words. In order to be robust enough to handle this kind of corpus, we require the following for a decompounder: first, obviously, compounds should be split, and non-compounds should be left untouched. This also applies if they are misspelled. Unknown words or words involving a part

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 253–256,
Columbus, Ohio, USA, June 2008.

in a foreign language are split if there is a plausible interpretation of them being a compound word. An example is *Turingmaschine* (Turing machine) in German, where Turing is an English word. Finally, words that are not really grammatical compounds, but due to the user forgetting to input the blankspace between the words (like *desktopcomputer*) are split.

For the evaluation, we have built and manually annotated gold standard sets for German, Dutch, Danish, Norwegian, Swedish and Finnish from fully anonymized search query logs. Because people do not use capitalization consistently when writing queries, all the query logs are lowercased. By randomly sampling keywords we would get few compounds (as their frequency is small compared to that of non-compounds), so we have proceeded in the following way to ensure that the gold-standards contain a substantial amount of compounds: we started by building a very naive decompounder that splits a word in several parts using a frequency-based compound splitting method (Koehn and Knight, 2003). Using this procedure, we obtain two random samples with possibly repeated words: one with words that are considered non-compounds, and the other with words that are considered compounds by this naive approach. Next, we removed all the duplicates from the previous list, and we had them annotated manually as compounds or non-compounds, including the correct splittings. The sizes of the final training sets vary between 2,000 and 3,600 words depending on the language. Each compound was annotated by two human judges who had received the previous instructions on when to consider that a keyword is a compound. For all the languages considered, exactly one of the two judges was a native speaker living in a country where it is the official language[1]. Table 1 shows the percentage of agreement in classifying words as compounds or non-compounds (Compound Classification Agreement, CCA) for each language and the Kappa score (Carletta, 1996) obtained from it, and the percentage of words for which also the decomposition provided was identical (Decompounding Agreement, DA). The most common source of disagreement were long words that could be split into two or more parts.

[1]This requisite is important because many queries contain novel or fashionable words.

Language	CCA	Kappa	DA
German	93%	0.86	88%
Dutch	96%	0.92	96%
Danish	89%	0.78	89%
Norwegian	93%	0.86	81%
Swedish	96%	0.92	95%
Finnish	92%	0.84	89%

Table 1: Inter-judge agreement metrics.

Language	Morphemes
German	∅,-e,+s,+e,+en,+nen,+ens,+es,+ns,+er
Dutch	∅,-e,+s,+e,+en
Danish	∅,+e,+s
Norwegian	∅,+e,+s
Swedish	∅,+o,+u,+e,+s
Finnish	∅

Table 2: Linking morphemes used in this work.

The evaluation is done using the metrics precision, recall and accuracy, defined in the following way (Koehn and Knight, 2003):

- Correct splits: no. of compounds that are split correctly.
- Correct non-splits: no. non-compounds that are not split.
- Wrong non-splits: no. of compounds and are not split.
- Wrong faulty splits: no. of compounds that are incorrectly split.
- Wrong splits: no. of non-compounds that are split.

$$Precision = \frac{\text{correct splits}}{\text{correct splits} + \text{wrong faulty splits} + \text{wrong splits}}$$

$$Recall = \frac{\text{correct splits}}{\text{correct splits} + \text{wrong faulty splits} + \text{wrong non-splits}}$$

$$Accuracy = \frac{\text{correct splits}}{\text{correct splits} + \text{wrong splits}}$$

3 Combining corpus-based features

Most approaches for decompounding can be considered as having this general structure: given a word w, calculate every possible way of splitting w in one or more parts, and score those parts according to some weighting function. If the highest scoring splitting contains just one part, it means that w is not a compound.

For the first step (calculating every possible splitting), it is common to take into account that modifiers inside compound words sometimes need *linking morphemes*. Table 2 lists the ones used in our system (Langer, 1998; Marek, 2006; Krott, 1999).

Method	Precision	Recall	Accuracy
Never split	-	0.00%	64.09%
Geometric mean of frequencies	39.77%	54.06%	65.58%
Compound probability	60.41%	**80.68%**	76.23%
Mutual Information	**82.00%**	48.29%	80.52%
Support-Vector Machine	**83.56%**	**79.48%**	**87.21%**

Table 3: Results of the several configurations.

Language	P	R	A
German	83.56%	79.48%	87.21%
Dutch	78.99%	76.18%	83.45%
Danish	81.97%	87.12%	85.36%
Norwegian	88.13%	93.05%	90.40%
Swedish	83.34%	92.98%	87.79%
Finnish	90.79%	91.21%	91.62%

Table 4: Results in all the different languages.

Concerning the second step, there is some work that uses, for scoring, additional information such as rules for cognate recognition (Brown, 2002) or sentence-aligned parallel corpora and a translation model, as in the full system described by Koehn and Knight (2003). When those resources are not available, the most common methods used for compound splitting are using features such as the geometric mean of the frequencies of compound parts in a corpus, as in Koehn and Knight (2003)'s back-off method, or learning a language model from a corpus and estimating the probability of each sequence of possible compound parts (Schiller, 2005; Marek, 2006). While these methods are useful for several applications, such as CLIR and MT, they have known weaknesses, such as preferring a decomposition if a compound part happens to be very frequent by chance, in the case of the frequency-based method, or the preference of decompositions with the least possible number of parts, in the case of the probability-based method.

Alfonseca et al. (2008) describe an integration of the previous methods, together with the Mutual Information and additional features obtained from web anchor texts to train a supervised German decompounder that outperforms the previous methods used as standalone. The geometric mean of the frequencies of compound parts and the probability estimated from the language model usually attain a high recall, given they are based on unigram features which are easy to collect, but they have some weaknesses, as mentioned above. On the other hand, while Mutual Information is a much more precise metric, it is less likely to have evidence about every single possible pair of compound parts from a corpus, so it suffers from low recall. A combination of all these metrics into a learning model is able to attain a high recall. An ablation study, reported in that paper, indicated that the contribution of the web anchor texts is minimal, so in this study we have just kept the other three metrics. Table 3 shows the results reported for German, training (i.e. counting frequencies and learning the language model) on the query keywords, and running a 10-fold cross validation of a SVM with a polynomial kernel using the German gold-standard. The supervised system improves over the single unsupervised metrics, attaining simultaneously good recall and precision metrics.

4 Experiments and evaluation

The first motivation of this work is to test whether the results reported for German are easy to reproduce in other languages. The results, shown in Table 4, are very similar across languages, having precision and recall values over 80% for most languages. A notable exception is Dutch, for which the inter-judge agreement was the highest, so we expected the set of words to be easier to classify. An analysis of the errors reported in the 10-fold cross-validation indicates that most errors in Dutch were wrong non-splits (in 147 cases) and wrong splits (in 139 cases), with wrong faulty splits happening only in 20 occasions. Many of the wrong splits are location names and trademarks, like *youtube*, *piratebay* or *smallville*.

While the supervised model gives much better results than the unsupervised ones, it still requires the construction of a goldstandard from which to train, which is usually costly. Therefore, we ran another experiment to check whether the models trained from some languages are applicable to other languages. Table 5 shows the results obtained in this case, the last column indicating the results when the model is trained from the training instances from all the other languages together. For each row, the highest value and those which are inside its 95% confidence interval are highlighted. Interestingly, apart from a few exceptions, the results are rather good for all the pairs of training and test language.

	Language for training						
	de	**nl**	**da**	**no**	**sv**	**fi**	**others**
de	P:83.56	P:78.69	P:74.96	P:**88.93**	P:82.72	P:**89.69**	P:80.89
	R:79.48	R:75.48	R:**92.77**	R:89.26	R:**90.79**	R:89.96	R:76.07
	A:87.21	A:82.76	A:83.53	A:**90.31**	A:86.53	A:**90.82**	A:88.15
nl	P:79.52	P:78.99	P:76.93	P:**92.81**	P:85.67	P:**90.98**	P:77.53
	R:75.74	R:76.18	R:**89.02**	R:55.08	R:**87.15**	R:86.73	R:76.54
	A:87.77	A:83.45	A:83.21	A:**91.00**	A:86.47	A:88.95	A:82.32
da	P:82.21	P:**90.86**	P:81.97	P:90.61	P:85.52	P:**92.65**	P:76.28
	R:45.01	R:42.94	R:87.12	R:80.25	R:81.41	R:82.46	R:**94.84**
	A:78.95	A:74.78	A:85.36	A:**89.30**	A:83.70	A:87.55	A:84.60
no	P:68.23	P:70.18	P:74.85	P:88.13	P:82.25	P:**90.08**	P:88.78
	R:83.33	R:87.18	R:**96.67**	R:93.05	R:94.21	R:91.84	R:90.88
	A:83.77	A:80.67	A:84.18	A:90.40	A:87.24	A:**91.41**	A:89.85
sv	P:76.57	P:77.33	P:76.31	P:89.00	P:83.34	P:**90.81**	P:83.89
	R:79.76	R:81.79	R:**94.66**	R:90.41	R:92.98	R:90.86	R:92.05
	A:87.18	A:83.38	A:84.57	A:89.67	A:87.79	A:**91.38**	A:87.69
fi	P:74.12	P:74.50	P:75.93	P:88.71	P:83.54	P:**90.79**	P:**90.70**
	R:80.12	R:81.67	R:**95.39**	R:91.46	R:92.70	R:91.21	R:90.62
	A:85.93	A:81.98	A:84.51	A:90.07	A:87.52	A:**91.62**	A:**91.18**

Table 5: Result training and testing in different languages.

Thus, the use of features like frequencies, probabilities or mutual information of compound parts is truly language-independent and the models learned from one language can safely be applied for decompounding a different language without the need of annotating a gold-standard for it.

Still, some trends in the results can be observed: training with the Danish corpus produced the best results in terms of recall for all the languages, but recall for Danish still improved when we trained on data from all languages. We believe that this indicates that the Danish dataset contains items with a more varied sets of feature combinations, so that the models trained from it have a good coverage on different kinds of compounds, but models trained in other languages are not able to identify many of the compounds in the Danish dataset. Concerning precision, training with either the Norwegian or the Finnish data produced very good results for most languages. This is consistent with the monolingual experiments (see Table 4) in which these languages had the best results. We believe these trends are probably due to the quality of the training data. Interestingly, the size of the training data is not so relevant, as most of the best results are not located at the last column in the table.

5 Conclusions

This paper shows that a combination of several corpus-based metrics for decompounding, previously applied to German, with big improvements with respect to other state-of-the-art systems, is also useful for other compounding languages. More interestingly, models learned from a goldstandard created for some language can be applied to other languages, sometimes producing better results than when a model is trained and tested in the same language. This should alleviate the fact that the proposed system is supervised, as there should just be the need of creating a goldstandard in one language in order to train a generic decompounder, thus facilitating the availability of decompounders for smaller languages like Faroese. For future work, we plan to investigate more deeply how the quality of the data affects the results, with a more detailed error analysis. Other open lines include exploring the addition of new features to the trained models.

References

S.F. Adafre, W.R. van Hage, J. Kamps, G.L. de Melo, and M. de Rijke. 2004. The University of Amsterdam at CLEF 2004. *CLEF 2004 Workshop*, pages 91–98.

M. Adda-Decker, G. Adda, and L. Lamel. 2000. Investigating text normalization and pronunciation variants for German broadcast transcription. In *ICSLP-2000*.

E. Alfonseca, S. Bilac, and S. Pharies. 2008. German decompounding in a difficult corpus. In *CICLING*.

M. Braschler and B. Ripplinger. 2004. How effective is stemming and decompounding for german text retrieval? *Information Retrieval*, 7:291–316.

R.D. Brown. 2002. Corpus-driven splitting of compound words. In *TMI-2002*.

J. Carletta. 1996. Assessing agreement on classification tasks: the Kappa statistics. *Computational Linguistics*, 22(2):249–254.

P. Koehn and K. Knight. 2003. Empirical methods for compound splitting. In *ACL-2003*.

A. Krott. 1999. Linking elements in compounds. LINGUIST, 7 Oct 1999. http://listserv.linguistlist.org/cgi-bin/wa?A2=ind9910a&L=linguist&P=6009.

S. Langer. 1998. Zur Morphologie und Semantik von Nominalkomposita. *Tagungsband der 4. Konferenz zur Verarbeitung naturlicher Sprache (KONVENS)*.

T. Marek. 2006. Analysis of german compounds using weighted finite state transducers. Technical report, BA Thesis, Universität Tbingen.

C. Monz and M. de Rijke. 2001. Shallow morphological analysis in monolingual information retrieval for Dutch, German and Italian. In *CLEF-2001*.

A. Schiller. 2005. German compound analysis with wfsc. In *Finite State Methods and NLP 2005*.

Multi-domain Sentiment Classification

Shoushan Li and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
{sshanli,cqzong}@nlpr.ia.ac.cn

Abstract

This paper addresses a new task in sentiment classification, called multi-domain sentiment classification, that aims to improve performance through fusing training data from multiple domains. To achieve this, we propose two approaches of fusion, feature-level and classifier-level, to use training data from multiple domains simultaneously. Experimental studies show that multi-domain sentiment classification using the classifier-level approach performs much better than single domain classification (using the training data individually).

1 Introduction

Sentiment classification is a special task of text categorization that aims to classify documents according to their opinion of, or sentiment toward a given subject (e.g., if an opinion is supported or not) (Pang et al., 2002). This task has created a considerable interest due to its wide applications.

Sentiment classification is a very domain-specific problem; training a classifier using the data from one domain may fail when testing against data from another. As a result, real application systems usually require some labeled data from multiple domains, guaranteeing an acceptable performance for different domains. However, each domain has a very limited amount of training data due to the fact that creating large-scale high-quality labeled corpora is difficult and time-consuming. Given the limited multi-domain training data, an interesting task arises, how to best make full use of all training data to improve sentiment classification performance. We name this new task, 'multi-domain sentiment classification'.

In this paper, we propose two approaches to multi-domain sentiment classification. In the first, called feature-level fusion, we combine the feature sets from all the domains into one feature set. Using the unified feature set, we train a classifier using all the training data regardless of domain. In the second approach, classifier-level fusion, we train a base classifier using the training data from each domain and then apply combination methods to combine the base classifiers.

2 Related Work

Sentiment classification has become a hot topic since the publication work that discusses classification of movie reviews by Pang et al. (2002). This was followed by a great many studies into sentiment classification focusing on many domains besides that of movie.

Research into sentiment classification over multiple domains remains sparse. It is worth noting that Blitzer et al. (2007) deal with the domain adaptation problem for sentiment classification where labeled data from one domain is used to train a classifier for classifying data from a different domain. Our work focuses on the problem of how to make multiple domains 'help each other' when all contain some labeled samples. These two problems are both important for real applications of sentiment classification.

3 Our Approaches

3.1 Problem Statement

In a standard supervised classification problem, we seek a predictor f (also called a classifier) that

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 257–260,
Columbus, Ohio, USA, June 2008.

maps an input vector x to the corresponding class label y. The predictor is trained on a finite set of labeled examples $\{ (X_i, Y_i) \}$ (i=1,…,n) and its objective is to minimize expected error, i.e.,

$$\hat{f} = \arg\min_{f \in \mathbf{H}} \sum_{i}^{n} L(f(X_i), Y_i)$$

Where L is a prescribed loss function and **H** is a set of functions called the hypothesis space, which consists of functions from x to y. In sentiment classification, the input vector of one document is constructed from weights of terms. The terms $(t_1,...,t_N)$ are possibly words, word n-grams, or even phrases extracted from the training data, with N being the number of terms. The output label y has a value of 1 or -1 representing a positive or negative sentiment classification.

In multi-domain classification, m different domains are indexed by k={1,…,m}, each with n_k training samples (X_{i_k}, Y_{i_k}) $i_k = \{1,...,n_k\}$. A straightforward approach is to train a predictor f_k for the k-th domain only using the training data $\{(X_{i_k}, Y_{i_k})\}$. We call this approach single domain classification and show its architecture in Figure 1.

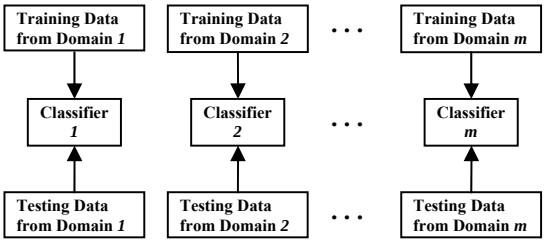


Figure 1: The architecture of single domain classification.

3.2 Feature-level Fusion Approach

Although terms are extracted from multiple domains, some occur in all domains and convey the same sentiment (this can be called global sentiment information). For example, some terms like 'excellent' and 'perfect' express positive sentiment information independent of domain. To learn the global sentiment information more correctly, we can pool the training data from all domains for training. Our first approach is using a common set of terms $(t'_1,...,t'_{N_{all}})$ to construct a uniform feature vector x' and then train a predictor using all training data:

$$\widehat{f_{all}} = \arg\min_{f_{all} \in \mathbf{H}_{all}} \sum_{k=1}^{m} \sum_{i_k=1}^{n_k} L(f(X'_{i_k}), Y_{i_k})$$

We call this approach feature-level fusion and show its architecture in Figure 2. The common set of terms is the union of the term sets from multiple domains.

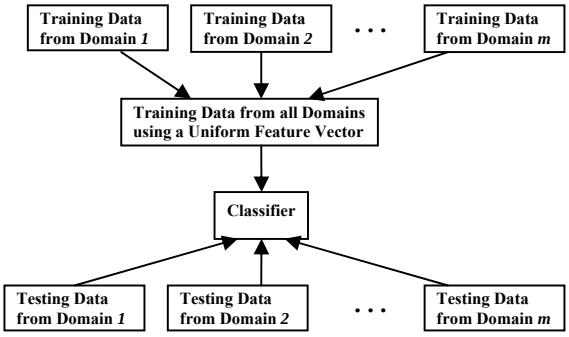


Figure 2: The architecture of the feature-level fusion approach

Feature-level fusion approach is simple to implement and needs no extra labeled data. Note that training data from different domains contribute differently to the learning process for a specific domain. For example, given data from three domains, books, DVDs and kitchen, we decide to train a classifier for classifying reviews from books. As the training data from DVDs is much more similar to books than that from kitchen (Blitzer et al., 2007), we should give the data from DVDs a higher weight. Unfortunately, the feature-level fusion approach lacks the capacity to do this. A more qualified approach is required to deal with the differences among the classification abilities of training data from different domains.

3.3 Classifier-level Fusion Approach

As mentioned in sub-Section 2.1, single domain classification is used to train a single classifier for each domain using the training data in the corresponding domain. As all these single classifiers aim to determine the sentiment orientation of a document, a single classifier can certainly be used to classify documents from other domains. Given multiple single classifiers, our second approach is to combine them to be a multiple classifier system for sentiment classification. We call this approach classifier-level fusion and show its architecture in Figure 3. This approach consists of two main steps:

(1) train multiple base classifiers (2) combine the base classifiers. In the first step, the base classifiers are multiple single classifiers f_k (k=1,…,m) from all domains. In the second step, many combination methods can be applied to combine the base classifiers. A well-known method called meta-learning (ML) has been shown to be very effective (Vilalta and Drissi, 2002). The key idea behind this method is to train a meta-classifier with input attributes that are the output of the base classifiers.

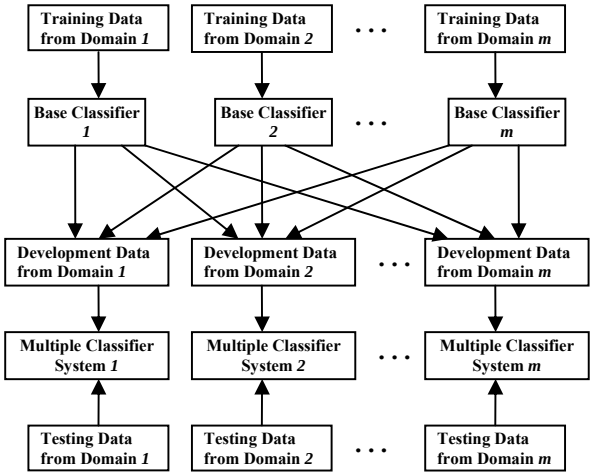


Figure 3: The architecture of the classifier-level fusion approach

Formally, let $X_{k'}$ denote a feature vector of a sample from the development data of the k'-th domain ($k'=1,...,m$) . The output of the k-th base classifier f_k on this sample is the probability distribution over the set of classes $\{c_1, c_2, ..., c_n\}$, i.e.,

$$p_k(X_{k'}) = < p_k(c_1 \mid X_{k'}), ..., p_k(c_n \mid X_{k'}) >$$

For the k'-th domain, we train a meta-classifier $f_{k'}$ ($k'=1,...,m$) using the development data from the k'-th domain with the meta-level feature vector $X_{k'}^{meta} \in R^{m \cdot n}$

$$X_{k'}^{meta} = < p_1(X_{k'}), ..., p_k(X_{k'}), ..., p_m(X_{k'}) >$$

Each meta-classifier is then used to test the testing data from the same domain.

Different from the feature-level approach, the classifier-level approach treats the training data from different domains individually and thus has the ability to take the differences in classification abilities into account.

4 Experiments

Data Set: We carry out our experiments on the labeled product reviews from four domains: books, DVDs, electronics, and kitchen appliances[1]. Each domain contains 1,000 positive and 1,000 negative reviews.

Experiment Implementation: We apply SVM algorithm to construct our classifiers which has been shown to perform better than many other classification algorithms (Pang et al., 2002). Here, we use LIBSVM[2] with a linear kernel function for training and testing. In our experiments, the data in each domain are partitioned randomly into training data, development data and testing data with the proportion of 70%, 20% and 10% respectively. The development data are used to train the meta-classifier.

Baseline: The baseline uses the single domain classification approach mentioned in sub-Section 2.1. We test four different feature sets to construct our feature vector. First, we use unigrams (e.g., 'happy') as features and perform the standard feature selection process to find the optimal feature set of unigrams (1Gram). The selection method is Bi-Normal Separation (BNS) that is reported to be excellent in many text categorization tasks (Forman, 2003). The criterion of the optimization is to find the set of unigrams with the best performance on the development data through selecting the features with high BNS scores. Then, we get the optimal word bi-gram (e.g., 'very happy') (2Gram) and mixed feature set (1+2Gram) in the same way. The fourth feature set (1Gram+2Gram) also consists of unigrams and bi-grams just like the third one. The difference between them lies in their selection strategy. The third feature set is obtained through selecting the unigrams and bi-grams with high BNS scores while the fourth one is obtained through simply uniting the two optimal sets of 1Gram and 2Gram.

From Table 1, we see that 1Gram+2Gram features perform much better than other types of features, which implies that we need to select good unigram and bi-gram features separately before combine them. Although the size of our training data are smaller than that reported in Blitzer et al.

[1] This data set is collected by Blitzer et al. (2007): http://www.seas.upenn.edu/~mdredze/datasets/sentiment/

[2] LIBSVM is an integrated software for SVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

(2007) (70% vs. 80%), the classification performance is comparative to theirs.

Features	Books	DVDs	Electronic	Kitchen
1Gram	0.75	0.84	0.8	0.825
2Gram	0.75	0.73	0.815	0.785
1+2Gram	0.765	0.81	0.825	0.80
1Gram+2Gram	**0.79**	**0.845**	**0.85**	**0.845**

Table 1: Accuracy results on the testing data of single domain classification using different feature sets.

We implement the fusion using 1+2Gram and 1Gram+2Gram respectively. From Figure 4, we see that both the two fusion approaches generally outperform single domain classification when using 1+2Gram features. They increase the average accuracy from 0.8 to 0.82375 and 0.83875, a significant relative error reduction of 11.87% and 19.38% over baseline.

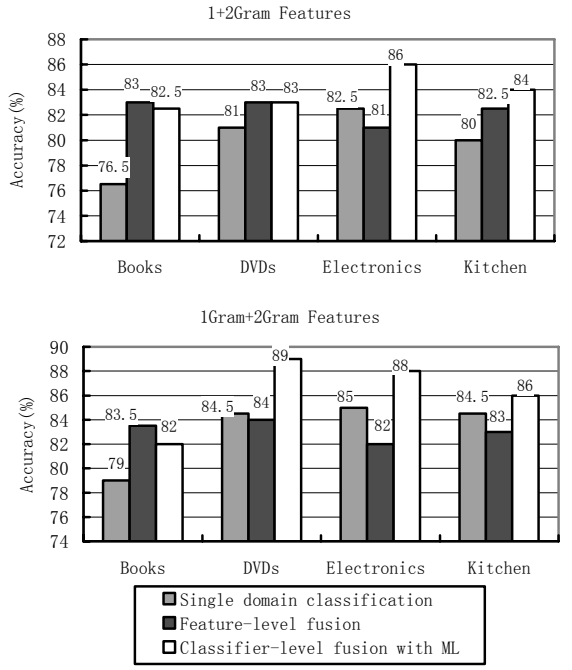


Figure 4: Accuracy results on the testing data using multi-domain classification with different approaches.

However, when the performance of baseline increases, the feature level approach fails to help the performance improvement in three domains. This is mainly because the base classifiers perform extremely unbalanced on the testing data of these domains. For example, the four base classifiers from Books, DVDs, Electronics, and Kitchen achieve the accuracies of 0.675, 0.62, 0.85, and 0.79 on the testing data from Electronics respectively. Dealing with such an unbalanced performance, we definitely need to put enough high weight on the training data from Electronics. However, the feature-level fusion approach simply pools all training data from different domains and treats them equally. Thus it can not capture the unbalanced information. In contrast, meta-learning is able to learn the unbalance automatically through training the meta-classifier using the development data. Therefore, it can still increase the average accuracy from 0.8325 to 0.8625, an impressive relative error reduction of 17.91% over baseline.

5 Conclusion

In this paper, we propose two approaches to multi-domain classification task on sentiment classification. Empirical studies show that the classifier-level approach generally outperforms the feature approach. Compared to single domain classification, multi-domain classification with the classifier-level approach can consistently achieve much better results.

Acknowledgments

The research work described in this paper has been partially supported by the Natural Science Foundation of China under Grant No. 60575043, and 60121302, National High-Tech Research and Development Program of China under Grant No. 2006AA01Z194, National Key Technologies R&D Program of China under Grant No. 2006BAH03B02, and Nokia (China) Co. Ltd as well.

References

J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*.

G. Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3: 1533-7928.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

R. Vilalta and Y. Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2): 77–95.

Evaluating Word Prediction: Framing Keystroke Savings

Keith Trnka and Kathleen F. McCoy
University of Delaware
Newark, DE 19716
trnka@cis.udel.edu

Abstract

Researchers typically evaluate word prediction using keystroke savings, however, this measure is not straightforward. We present several complications in computing keystroke savings which may affect interpretation and comparison of results. We address this problem by developing two gold standards as a frame for interpretation. These gold standards measure the maximum keystroke savings under two different approximations of an ideal language model. The gold standards additionally narrow the scope of deficiencies in a word prediction system.

1 Introduction

Word prediction is an application of language modeling to speeding up text entry, especially to entering utterances to be spoken by an Augmentative and Alternative Communication (AAC) device. AAC devices seek to address the dual problem of speech and motor impairment by attempting to optimize text input. Even still, communication rates with AAC devices are often below 10 words per minute (Newell et al., 1998), compared to the common 130-200 words per minute speech rate of speaking people. Word prediction addresses these issues by reducing the number of keystrokes required to produce a message, which has been shown to improve communication rate (Trnka et al., 2007). The reduction in keystrokes also translates into a lower degree of fatigue from typing all day (Carlberger et al., 1997).

Word prediction systems present multiple completions of the current word to the user. Systems generate a list of W predictions on the basis of the word being typed and a language model. The vocabulary is filtered to match the prefix of the current word and the language model ranks the words according to their likelihood. In the case that no letters of the current word have been entered, the language model is the sole factor in generating predictions. Systems often use a touchscreen or function/number keys to select any of the predicted words.

Because the goal of word prediction systems is to reduce the number of keystrokes, the primary evaluation for word prediction is keystroke savings (Garay-Vitoria and Abascal, 2006; Newell et al., 1998; Li and Hirst, 2005; Trnka and McCoy, 2007; Carlberger et al., 1997). Keystroke savings (KS) measures the percentage reduction in keys pressed compared to letter-by-letter text entry.

$$KS = \frac{keys_{normal} - keys_{with\ prediction}}{keys_{normal}} \times 100\%$$

A word prediction system that offers higher savings will benefit a user more in practice.

However, the equation for keystroke savings has two major deficiencies. Firstly, the equation alone is not enough to compute keystroke savings — actually computing keystroke savings requires a precise definition of a keystroke and also requires a method for determining how many keystrokes are used when predictions are available, discussed in Section 2. Beyond simply computing keystroke savings, the equation alone does not provide much in the way of interpretation — is 60% keystroke savings good? Can we do better? Section 3 will present two gold standards to allow better interpretation of keystroke savings.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 261–264,
Columbus, Ohio, USA, June 2008.

2 Computing Keystroke Savings

We must have a way to determine how many keystrokes a user would take under both letter-by-letter entry and word prediction to compute keystroke savings. The common trend in research is to simulate a "perfect" user that will never make typing mistakes and will select a word from the predictions as soon as it appears.

Implementation of perfect utilization of the predictions is not always straightforward. For example, consider the predictive interface in Microsoft Word™: a single prediction is offered as an inline completion. If the prediction is selected, the user may backspace and edit the word. However, this freedom makes finding the minimum sequence of keys more difficult — now the user may select a prediction with the incorrect suffix and correct the suffix as the optimal action. We feel that a more intuitive interface would allow a user to undo the prediction selection by pressing backspace, an interface which does not support backspace-editing. In addition to backspacing, future research in multi-word prediction will face a similar problem, analogous to the garden-path problem in parsing, where a greedy approach does not always give the optimal result.

The keystrokes used for training and testing word prediction systems can affect the results. We attempt to evaluate word prediction as realistically as possible. Firstly, many corpora have punctuation marks, but an AAC user in a conversational setting is unlikely to use punctuation due to the high cost of each key press. Therefore, we remove punctuation on the outside of words, such as commas and periods, but leave word-internal punctuation intact. Also, we treat capital letters as a single key press, reflecting the trend of many AAC users to avoid capitalization. Another problem occurs for a newline or "speak key", which the user would press after completing an utterance. In pilot studies, including the simulation of a speak key lowered keystroke savings by 0.8–1.0% for window sizes 1–10, because newlines are not able to be predicted in the system. However, we feel that the simulation of a speak key will produce an evaluation metric that is closer to the actual user's experience, therefore we include a speak key in our evaluations.

An evaluation of word prediction must address these issues, if only implicitly. The effect of these potentially implicit decisions on keystroke savings can make comparison of results difficult. However, if results are presented in reference to a gold standard under the same assumptions, we can draw more reliable conclusions from results.

3 Towards a Gold Standard

In trying to improve the state of word prediction, several researchers have noted that it seems extremely difficult to improve keystroke savings beyond a certain point. Copestake (1997) discussed the entropy of English to conclude that 50–60% keystroke savings may be the most we can expect in practice. Lesher et al. (2002) replaced the language model in a word prediction system with a human to try and estimate the limit of keystroke savings. They found that humans could achieve 59% keystroke savings with access to their advanced language model and that their advanced language model alone achieved 54% keystroke savings. They noted that one subject achieved nearly 70% keystroke savings on one particular text, and concluded that further improvements on current methods are possible. Garay-Vitoria and Abascal (2006) survey many prediction systems, showing a wide spectrum of savings, but no system offers more than 70% keystroke savings.

We investigated the problem of the limitations of keystroke savings first from a theoretical perspective, seeking a clearly defined upper boundary. Keystroke savings can never reach 100% — it would mean that the system divined the entire text they intended without a single key.

3.1 Theoretical keystroke savings limit

The minimum amount of input required corresponds to a perfect system — one that predicts every word as soon as possible. In a word *completion* system, the predictions are delayed until after the first character of the word is entered. In such a system, the minimum amount of input using a perfect language model is two keystrokes per word — one for the first letter and one to select the prediction. The system would also require one keystroke per sentence. In a word *prediction* system, the predictions are available immediately, so the minimal in-

put for a perfect system is one keystroke per word (to select the prediction) and one keystroke per sentence. We added the ability to measure the minimum number of keystrokes and maximum savings to our simulation software, which we call the *theoretical keystroke savings limit*.

We evaluated a baseline trigram model under two conditions with different keystroke requirements on the Switchboard corpus. The simulation software was modified to output the theoretical limit in addition to actual keystroke savings at various window sizes. To demonstrate the effect of the theoretical keystroke savings limit on actual savings, we evaluated the trigram model under conditions with two different limits — word prediction and word completion. The evaluation of the trigram model using word *completion* is shown in Figure 1. The actual keystroke savings is graphed by window size in reference to the theoretical limit. As noted by other researchers, keystroke savings increases with window size, but with diminishing returns (this is the effect of placing the most probable words first). One of

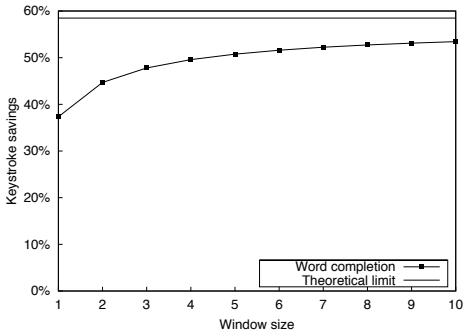


Figure 1: Keystroke savings and the limit vs. window size for word completion.

the problems with word completion is that the theoretical limit is so close to actual performance — around 58.5% keystroke savings compared to 50.8% keystroke savings with five predictions. At only five predictions, the system has already realized 87% of the possible keystroke savings. Under these circumstances, it would take a drastic change in the language model to impact keystroke savings.

We repeated this analysis for word *prediction*, shown in Figure 2 alongside word completion. Word prediction is much higher than completion, both theoretically (the limit) and in actual keystroke savings.

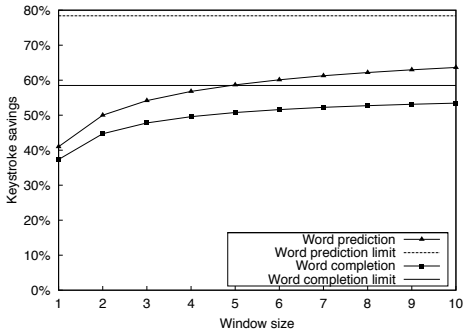


Figure 2: Keystroke savings and the limit vs. window size for word prediction compared to word completion.

Word prediction offers much more headroom in terms of improvements in keystroke savings. Therefore our ongoing research will focus on word prediction over word completion.

This analysis demonstrates a limit to keystroke savings, but this limit is slightly different than Copestake (1997) and Lesher et al. (2002) seek to describe — beyond the limitations of the user interface, there seems to be a limitation on the predictability of English. Ideally, we would like to have a gold standard that is a closer estimate of an ideal language model.

3.2 Vocabulary limit

We can derive a more practical limit by simulating word prediction using a perfect model of all words that occur in the training data. This gold standard will predict the correct word immediately so long as it occurs in the training corpus. Words that never occurred in training require letter-by-letter entry. We call this measure the *vocabulary limit* and apply it to evaluate whether the difference between training and testing vocabulary is significant. Previous research has focused on the percentage of out-of-vocabulary (OOV) terms to explain changes in keystroke savings (Trnka and McCoy, 2007; Wandmacher and Antoine, 2006). In contrast, the vocabulary limit gives more guidance for research by translating the problem of OOVs into keystroke savings.

Expanding the results from the theoretical limit, the vocabulary limit is 77.6% savings, compared to 78.4% savings for the theoretical limit and 58.7% actual keystroke savings with 5 predictions. The practical limit is very close to the theoretical limit

in the case of Switchboard. Therefore, the remaining gap between the practical limit and actual performance must be due to other differences between testing and training data, limitations of the model, and limitations of language modeling.

3.3 Application to corpus studies

We applied the gold standards to our corpus study, in which a trigram model was individually trained and tested on several different corpora (Trnka and McCoy, 2007). In contrast to the actual trigram model

Corpus	Trigram	Vocab. limit	Theor. limit
AAC Email	48.92%	61.94%	84.83%
Callhome	43.76%	54.62%	81.38%
Charlotte	48.30%	65.69%	83.74%
SBCSAE	42.30%	60.81%	79.86%
Micase	49.00%	69.18%	84.08%
Switchboard	60.35%	80.33%	82.57%
Slate	53.13%	81.61%	85.88%

Table 1: A trigram model compared to the limits.

performance, the theoretical limits all fall within a relatively narrow range, suggesting that the achievable keystroke savings may be similar even across different domains. The more technical and formal corpora (Micase, Slate, AAC) show higher limits, as the theoretical limit is based on the length of words and sentences in each corpus. The practical limit exhibits much greater variation. Unlike the Switchboard analysis, many other corpora have a substantial gap between the theoretical and practical limits. Although the practical measure seems to match the actual savings similarly to OOVs testing with cross-validation (Trnka and McCoy, 2007), this measure more concretely illustrates the effect of OOVs on actual keystroke savings — 60% keystroke savings when training and testing on AAC Email would be extraordinary.

4 Conclusions

Although keystroke savings is the predominant evaluation for word prediction, this evaluation is not straightforward, exacerbating the problem of interpreting and comparing results. We have presented a novel solution — interpreting results alongside gold standards which capture the difficulty of the evaluation. These gold standards are also applicable to drive future research — if actual performance is very close to the theoretical limit, then relaxing the minimum keystroke requirements should be the most beneficial (e.g., multi-word prediction). Similarly, if actual performance is very close to the vocabulary limit, then the vocabulary of the language model must be improved (e.g., cache modeling, adding general-purpose training data). In the case that keystroke savings is far from either limit, then research into improving the language model is likely to be the most beneficial.

Acknowledgments

This work was supported by US Department of Education grant H113G040051.

References

Alice Carlberger, John Carlberger, Tina Magnuson, M. Sharon Hunnicutt, Sira Palazuelos-Cagigas, and Santiago Aguilera Navarro. 1997. Profet, a new generation of word prediction: An evaluation study. In *ACL-97 workshop on Natural Language Processing for Communication Aids*.

Ann Copestake. 1997. Augmented and alternative NLP techniques for augmentative and alternative communication. In *ACL-97 workshop on Natural Language Processing for Communication Aids*, pages 37–42.

Nestor Garay-Vitoria and Julio Abascal. 2006. Text prediction systems: a survey. *Univ Access Inf Soc*, 4:183–203.

Gregory W. Lesher, Bryan J. Moulton, D Jeffery Higginbotham, and Brenna Alsofrom. 2002. Limits of human word prediction performance. In *CSUN*.

Jianhua Li and Graeme Hirst. 2005. Semantic knowledge in word completion. In *ASSETS*, pages 121–128.

Alan Newell, Stefan Langer, and Marianne Hickey. 1998. The rôle of natural language processing in alternative and augmentative communication. *Natural Language Engineering*, 4(1):1–16.

Keith Trnka and Kathleen F. McCoy. 2007. Corpus Studies in Word Prediction. In *ASSETS*, pages 195–202.

Keith Trnka, Debra Yarrington, John McCaw, Kathleen F. McCoy, and Christopher Pennington. 2007. The Effects of Word Prediction on Communication Rate for AAC. In *NAACL-HLT; Companion Volume: Short Papers*, pages 173–176.

Tonio Wandmacher and Jean-Yves Antoine. 2006. Training Language Models without Appropriate Language Resources: Experiments with an AAC System for Disabled People. In *Eurospeech*.

Pairwise Document Similarity in Large Collections with MapReduce

Tamer Elsayed,* Jimmy Lin,† and Douglas W. Oard†
Human Language Technology Center of Excellence and
UMIACS Laboratory for Computational Linguistics and Information Processing
University of Maryland, College Park, MD 20742
{telsayed,jimmylin,oard}@umd.edu

Abstract

This paper presents a MapReduce algorithm for computing pairwise document similarity in large document collections. MapReduce is an attractive framework because it allows us to decompose the inner products involved in computing document similarity into separate multiplication and summation stages in a way that is well matched to efficient disk access patterns across several machines. On a collection consisting of approximately 900,000 newswire articles, our algorithm exhibits linear growth in running time and space in terms of the number of documents.

1 Introduction

Computing pairwise similarity on large document collections is a task common to a variety of problems such as clustering and cross-document coreference resolution. For example, in the PubMed search engine,[1] which provides access to the life sciences literature, a "more like this" browsing feature is implemented as a simple lookup of document-document similarity scores, computed offline. This paper considers a large class of similarity functions that can be expressed as an inner product of term weight vectors.

For document collections that fit into random-access memory, the solution is straightforward. As collection size grows, however, it ultimately becomes necessary to resort to disk storage, at which point aligning computation order with disk access patterns becomes a challenge. Further growth in the document collection will ultimately make it desirable to spread the computation over several processors, at which point interprocess communication becomes a second potential bottleneck for which the computation order must be optimized. Although tailored implementations can be designed for specific parallel processing architectures, the MapReduce framework (Dean and Ghemawat, 2004) offers an attractive solution to these challenges. In this paper, we describe how pairwise similarity computation for large collections can be efficiently implemented with MapReduce. We empirically demonstrate that removing high frequency (and therefore low entropy) terms results in approximately linear growth in required disk space and running time with increasing collection size for collections containing several hundred thousand documents.

2 MapReduce Framework

MapReduce builds on the observation that many tasks have the same structure: a computation is applied over a large number of records (e.g., documents) to generate partial results, which are then aggregated in some fashion. Naturally, the per-record computation and aggregation vary by task, but the basic structure remains fixed. Taking inspiration from higher-order functions in functional programming, MapReduce provides an abstraction that involves the programmer defining a "mapper" and a "reducer", with the following signatures:

$$\text{map: } (k_1, v_1) \rightarrow [(k_2, v_2)]$$
$$\text{reduce: } (k_2, [v_2]) \rightarrow [(k_3, v_3)]$$

Key/value pairs form the basic data structure in MapReduce. The "mapper" is applied to every input

*Department of Computer Science
†The iSchool, College of Information Studies
[1]http://www.ncbi.nlm.nih.gov/PubMed

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 265–268,
Columbus, Ohio, USA, June 2008.

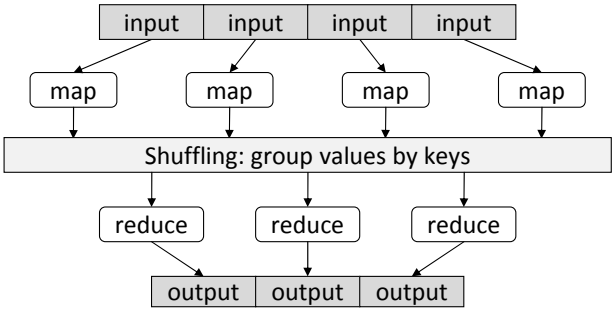


Figure 1: Illustration of the MapReduce framework: the "mapper" is applied to all input records, which generates results that are aggregated by the "reducer".

key/value pair to generate an arbitrary number of intermediate key/value pairs. The "reducer" is applied to all values associated with the same intermediate key to generate output key/value pairs (see Figure 1).

On top of a distributed file system (Ghemawat et al., 2003), the runtime transparently handles all other aspects of execution (e.g., scheduling and fault tolerance), on clusters ranging from a few to a few thousand nodes. MapReduce is an attractive framework because it shields the programmer from distributed processing issues such as synchronization, data exchange, and load balancing.

3 Pairwise Document Similarity

Our work focuses on a large class of document similarity metrics that can be expressed as an inner product of term weights. A document d is represented as a vector W_d of term weights $w_{t,d}$, which indicate the importance of each term t in the document, ignoring the relative ordering of terms ("bag of words" model). We consider symmetric similarity measures defined as follows:

$$sim(d_i, d_j) = \sum_{t \in V} w_{t,d_i} \cdot w_{t,d_j} \qquad (1)$$

where $sim(d_i, d_j)$ is the similarity between documents d_i and d_j and V is the vocabulary set. In this type of similarity measure, a term will contribute to the similarity between two documents only if it has non-zero weights in both. Therefore, $t \in V$ can be replaced with $t \in d_i \cap d_j$ in equation 1.

Generalizing this to the problem of computing similarity between *all* pairs of documents, we note that a term contributes to *each* pair that contains it.[2] For example, if a term appears in documents x, y, and z, it contributes *only* to the similarity scores between (x, y), (x, z), and (y, z). The list of documents that contain a particular term is exactly what is contained in the postings of an inverted index. Thus, by processing all postings, we can compute the entire pairwise similarity matrix by summing term contributions.

Algorithm 1 Compute Pairwise Similarity Matrix

1: $\forall i, j : sim[i, j] \Leftarrow 0$
2: **for all** $t \in V$ **do**
3: $\quad p_t \Leftarrow postings(t)$
4: $\quad$ **for all** $d_i, d_j \in p_t$ **do**
5: $\quad\quad sim[i, j] \Leftarrow sim[i, j] + w_{t,d_i} \cdot w_{t,d_j}$

Algorithm 1 formalizes this idea: $postings(t)$ denotes the list of documents that contain term t. For simplicity, we assume that term weights are also stored in the postings. For small collections, this algorithm can be run efficiently to compute the entire similarity matrix in memory. For larger collections, disk access optimization is needed—which is provided by the MapReduce runtime, without requiring explicit coordination.

We propose an efficient solution to the pairwise document similarity problem, expressed as two separate MapReduce jobs (illustrated in Figure 2):

1) Indexing: We build a standard inverted index (Frakes and Baeza-Yates, 1992), where each term is associated with a list of docid's for documents that contain it and the associated term weight. Mapping over all documents, the mapper, for each term in the document, emits the term as the key, and a tuple consisting of the docid and term weight as the value. The MapReduce runtime automatically handles the grouping of these tuples, which the reducer then writes out to disk, thus generating the *postings*.

2) Pairwise Similarity: Mapping over each posting, the mapper generates key tuples corresponding to pairs of docids in the postings: in total, $\frac{1}{2}m(m-1)$ pairs where m is the posting length. These key tuples are associated with the product of the corresponding term weights—they represent the individ-

[2] Actually, since we focus on symmetric similarity functions, we only need to compute half the pairs.

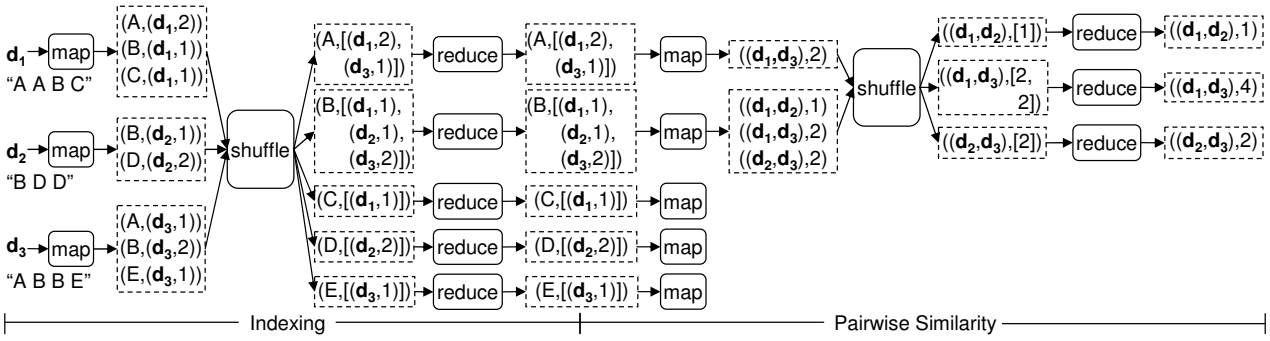


Figure 2: Computing pairwise similarity of a toy collection of 3 documents. A simple term weighting scheme ($w_{t,d} = tf_{t,d}$) is chosen for illustration.

ual term contributions to the final inner product. The MapReduce runtime sorts the tuples and then the reducer sums all the individual score contributions for a pair to generate the final similarity score.

4 Experimental Evaluation

In our experiments, we used Hadoop version 0.16.0,[3] an open-source Java implementation of MapReduce, running on a cluster with 20 machines (1 master, 19 slave). Each machine has two single-core processors (running at either 2.4GHz or 2.8GHz), 4GB memory, and 100GB disk.

We implemented the symmetric variant of Okapi-BM25 (Olsson and Oard, 2007) as the similarity function. We used the AQUAINT-2 collection of newswire text, containing 906k documents, totaling approximately 2.5 gigabytes. Terms were stemmed. To test the scalability of our technique, we sampled the collection into subsets of 10, 20, 25, 50, 67, 75, 80, 90, and 100 percent of the documents.

After stopword removal (using Lucene's stopword list), we implemented a *df*-cut, where a fraction of the terms with the highest document frequencies is eliminated.[4] This has the effect of removing non-discriminative terms. In our experiments, we adopt a 99% cut, which means that the most frequent 1% of terms were discarded (9,093 terms out of a total vocabulary size of 909,326). This technique greatly increases the efficiency of our algorithm, since the number of tuples emitted by the mappers in the pairwise similarity phase is dominated by the length of the longest posting (in the worst case, if a term appears in all documents, it would generate approximately 10^{12} tuples).

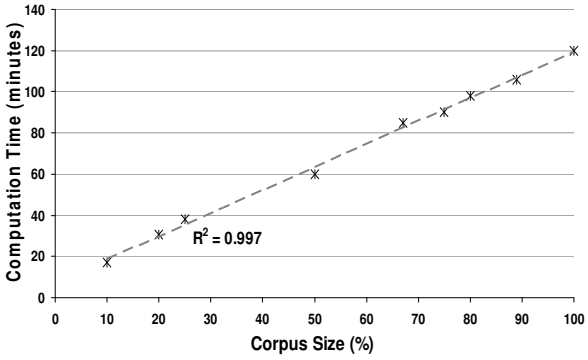


Figure 3: Running time of pairwise similarity comparisons, for subsets of AQUAINT-2.

Figure 3 shows the running time of the pairwise similarity phase for different collection sizes.[5] The computation for the entire collection finishes in approximately two hours. Empirically, we find that running time increases linearly with collection size, which is an extremely desirable property. To get a sense of the space complexity, we compute the number of intermediate document pairs that are emitted by the mappers. The space savings are large (3.7 billion rather than 8.1 trillion intermediate pairs for the entire collection), and space requirements grow linearly with collection size over this region ($R^2 = 0.9975$).

[3] http://hadoop.apache.org/

[4] In text classification, removal of rare terms is more common. Here we use *df*-cut to remove common terms.

[5] The entire collection was indexed in about 3.5 minutes.

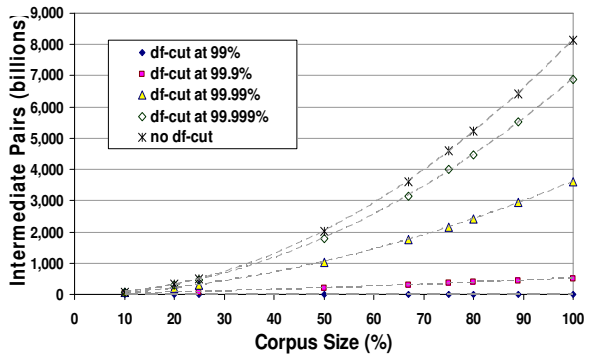


Figure 4: Effect of changing *df*-cut thresholds on the number of intermediate document-pairs emitted, for subsets of AQUAINT-2.

5 Discussion and Future Work

In addition to empirical results, it would be desirable to derive an analytical model of our algorithm's complexity. Here we present a preliminary sketch of such an analysis and discuss its implications. The complexity of our pairwise similarity algorithm is tied to the number of document pairs that are emitted by the mapper, which equals the total number of products required in $O(N^2)$ inner products, where N is the collection size. This is equal to:

$$\frac{1}{2}\sum_{t \in V} df_t(df_t - 1) \qquad (2)$$

where df_t is the document frequency, or equivalently the length of the postings for term t. Given that tokens in natural language generally obey Zipf's Law, and vocabulary size and collection size can be related via Heap's Law, it may be possible to develop a closed form approximation to the above series.

Given the necessity of computing $O(N^2)$ inner products, it may come as a surprise that empirically our algorithm scales linearly (at least for the collection sizes we explored). We believe that the key to this behavior is our *df*-cut technique, which eliminates the head of the *df* distribution. In our case, eliminating the top 1% of terms reduces the number of document pairs by several orders of magnitude. However, the impact of this technique on effectiveness (e.g., in a query-by-example experiment) has not yet been characterized. Indeed, a *df*-cut threshold of 99% might seem rather aggressive, removing meaning-bearing terms such as "arthritis" and "Cornell" in addition to perhaps less problematic terms such as "sleek" and "frail." But redundant use of related terms is common in news stories, which we would expect to reduce the adverse effect on many applications of removing these low entropy terms.

Moreover, as Figure 4 illustrates, relaxing the *df*-cut to a 99.9% threshold still results in approximately linear growth in the requirement for intermediate storage (at least over this region).[6] In essence, optimizing the *df*-cut is an efficiency vs. effectiveness tradeoff that is best made in the context of a specific application. Finally, we note that alternative approaches to similar problems based on locality-sensitive hashing (Andoni and Indyk, 2008) face similar tradeoffs in tuning for a particular false positive rate; cf. (Bayardo et al., 2007).

6 Conclusion

We present a MapReduce algorithm for efficiently computing pairwise document similarity in large document collections. In addition to offering specific benefits for a number of real-world tasks, we also believe that our work provides an example of a programming paradigm that could be useful for a broad range of text analysis problems.

Acknowledgments

This work was supported in part by the Intramural Research Program of the NIH/NLM/NCBI.

References

A. Andoni and P. Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *CACM*, 51(1):117–122.

R. Bayardo, Y. Ma, and R. Srikant. 2007. Scaling up all pairs similarity search. In *WWW '07*.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *OSDI '04*.

W. Frakes and R. Baeza-Yates. 1992. *Information Retrieval: Data Structures and Algorithms*.

S. Ghemawat, H. Gobioff, and S. Leung. 2003. The Google File System. In *SOSP '03*.

J. Olsson and D. Oard. 2007. Improving text classification for oral history archives with temporal domain knowledge. In *SIGIR '07*.

[6] More recent experiments suggest that a *df*-cut of 99.9% results in almost no loss of effectiveness on a query-by-example task, compared to no *df*-cut.

Text Segmentation with LDA-Based Fisher Kernel

Qi Sun, Runxin Li, Dingsheng Luo and Xihong Wu
Speech and Hearing Research Center, and
Key Laboratory of Machine Perception (Ministry of Education)
Peking University
100871, Beijing, China
{sunq,lirx,dsluo,wxh}@cis.pku.edu.cn

Abstract

In this paper we propose a domain-independent text segmentation method, which consists of three components. Latent Dirichlet allocation (LDA) is employed to compute words semantic distribution, and we measure semantic similarity by the Fisher kernel. Finally global best segmentation is achieved by dynamic programming. Experiments on Chinese data sets with the technique show it can be effective. Introducing latent semantic information, our algorithm is robust on irregular-sized segments.

1 Introduction

The aim of text segmentation is to partition a document into a set of segments, each of which is coherent about a specific topic. This task is inspired by problems in information retrieval, summarization, and language modeling, in which the ability to provide access to smaller, coherent segments in a document is desired.

A lot of research has been done on text segmentation. Some of them utilize linguistic criteria (Beeferman et al., 1999; Mochizuki et al., 1998), while others use statistical similarity measures to uncover lexical cohesion. Lexical cohesion methods believe a coherent topic segment contains parts with similar vocabularies. For example, the TextTiling algorithm, introduced by (Hearst, 1994), assumes that the local minima of the word similarity curve are the points of low lexical cohesion and thus the natural boundary candidates. (Reynar, 1998) has proposed a method called dotplotting depending on the distribution of word repetitions to find tight regions of topic similarity graphically. One of the problems with those works is that they treat terms uncorrelated, assigning them orthogonal directions in the feature space. But in reality words are correlated, and sometimes even synonymous, so that texts with very few common terms can potentially be on closely related topics. So (Choi et al., 2001; Brants et al., 2002) utilize semantic similarity to identify cohesion. Unsupervised models of texts that capture semantic information would be useful, particularly if they could be achieved with a "semantic kernel" (Cristianini et al., 2001) , which computes the similarity between texts by also considering relations between different terms. A Fisher kernel is a function that measures the similarity between two data items not in isolation, but rather in the context provided by a probability distribution. In this paper, we use the Fisher kernel to describe semantic information similarity. In addition, (Fragkou et al., 2004; Ji and Zha, 2004) has treated this task as an optimization problem with global cost function and used dynamic programming for segments selection.

The remainder of the paper is organized as follows. In section 2, after a brief overview of our method, some key aspects of the algorithm are described. In section 3, some experiments are presented. Finally conclusion and future research directions are drawn in section 4.

2 Methodology

This paper considers the sentence to be the smallest unit, and a block b is the segment candidate which consists of one or more sentences. We employ LDA

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 269–272,
Columbus, Ohio, USA, June 2008.

model (Blei et al., 2003) in order to find out latent semantic topics in blocks, and LDA-based Fisher kernel is used to measure the similarity of adjacent blocks. Each block is then given a final score based on its length and semantic similarity with its previous block. Finally the segmentation points are decided by dynamic programming.

2.1 LDA Model

We adopt LDA framework, which regards the corpus as mixture of latent topics and uses document as the unit of topic mixtures. In our method, the blocks defined in previous paragraph are regarded as "documents" in LDA model.

The LDA model defines two corpus-level parameters α and β. In its generative process, the marginal distribution of a document $p(d|\alpha, \beta)$ is given by the following formula:

$$\int p(\theta|\alpha)(\prod_{n=1}^{N}\sum_{k} p(z_k|\theta_d)p(w_n|z_k,\beta))d\theta$$

where d is a word sequence $(w_1, w_2, ...w_N)$ of length N. α parameterizes a Dirichlet distribution and derives the document-related random variable θ_d, then we choose a topic $z_k,\ k \in \{1...K\}$ from the multinomial distribution of θ_d. Word probabilities are parameterized by a $k \times V$ matrix β with V being the size of vocabulary and $\beta_{vk} = P(w = v|z_k)$. We use variational EM (Blei et al., 2003) to estimate the parameters.

2.2 LDA-Based Fisher Kernel

In general, a kernel function $k(x, y)$ is a way of measuring the resemblance between two data items x and y. The Fisher kernel's key idea is to derive a kernel function from a generative probability model. In this paper we follow (Hofmann, 2000) to consider the average log-probability of a block, utilizing the LDA model. The likelihood of b is given by:

$$l(b) = \sum_{i=1}^{N} \widehat{P}(w_i|b) \log \sum_{k=1}^{K} \beta_{w_i k}\theta_b^{(k)}$$

where the empirical distribution of words in the block $\widehat{P}(w_i|b)$ can be obtained from the number of word-block co-occurrence $n(b, w_i)$, normalized by the length of the block.

The Fisher kernel is defined as

$$K(b_1, b_2) = \bigtriangledown_\theta^T l(b_1) I^{-1} \bigtriangledown_\theta l(b_2)$$

which engenders a measure of similarity between any two blocks b_1 and b_2. The derivation of the kernel is quite straightforward and following (Hofmann, 2000) we finally have the result:

$$K(b_1, b_2) = K_1(b_1, b_2) + K_2(b_1, b_2), \quad with$$
$$K_1(b_1, b_2) = \sum_k \theta_{b_1}^{(k)}\theta_{b_2}^{(k)}/\theta_{corpus}^{(k)}$$
$$K_2(b_1, b_2) =$$
$$\sum_i \widehat{P}(w_i|b_1)\widehat{P}(w_i|b_2) \sum_k \frac{P(z_k|b_1,w_i)P(z_k|b_2,w_i)}{P(w_i|z_k)}$$

where $K_1(b_1, b_2)$ is a measure of how much b_1 and b_2 share the same latent topic, taking synonymy into account. And $K_2(b_1, b_2)$ is the traditional inner product of common term frequencies, but weighted by the degree to which these terms belong to the same latent topic, taking polysemy into account.

2.3 Cost Function and Dynamic Programming

The local minima of LDA-based Fisher kernel similarities indicate low semantic cohesion and segmentation candidates, which is not enough to get reasonably-sized segments. The lengths of segmentation candidates have to be considered, thus we build a cost function including two parts of information. Segmentation points can be given in terms of a vector $\vec{t} = (t_0, ..., t_m, ..., t_M)$, where t_m is the sentence label with m indicating the mth block. We define a cost function as follows:

$$\begin{aligned} J(\vec{t};\lambda) &= \sum_{m=1}^{M} \lambda F(l_{t_m+1,t_{m+1}}) \\ &+ K(b_{t_{m-1}+1,t_m}, b_{t_m+1,t_{m+1}}) \end{aligned}$$

where $F(l_{t_m+1,t_{m+1}})$ is equal to $\frac{(l_{t_m+1,t_{m+1}}-\mu)^2}{2\sigma^2}$ and $l_{t_m+1,t_{m+1}}$ is equal to $t_{m+1} - t_m$ indicating the number of sentences in block m. The LDA-based kernel function measures similarity of block $m-1$ and block m, where block $m-1$ spans sentence $t_{m-1}+1$ to t_m and block m spans sentence $t_m + 1$ to t_{m+1}

The cost function is the sum of the costs of assumed unknown M segments, each of which is made up of the length probability of block m and the similarity score of block m with its previous block $m-1$. The optimal segmentation $\vec{t}$ gives a global minimum of $J(\vec{t};\lambda)$.

3 Experiments

3.1 Preparation

In our experiments, we evaluate the performance of our algorithms on Chinese corpus. With news documents from Chinese websites, collected from 10 different categories, we design an artificial test corpus in the similar way of (Choi, 2000), in which we take each n-sentence document as a coherent topic segment, randomly choose ten such segments and concatenate them as a sample. Three data sets, Set 3-5, Set 13-15 and Set 5-20, are prepared in our experiments, each of which contains 100 samples. The data sets' names are represented by a range number n of sentences in a segment.

Due to generality, we take three indices to evaluate our algorithm: precision, recall and error rate metric (Beeferman et al., 1999) . And all experimental results are averaged scores generated from the individual results of different samples. In order to determine appropriate parameters, some hold-out data are used.

We compare the performance of our methods with the algorithm in (Fragkou et al., 2004) on our test set. In particular, the similarity representation is a main difference between those two methods. While we pay attention to latent topic information behind words of adjacent blocks, (Fragkou et al., 2004) calculates word density as the similarity score function.

3.2 Results

In order to demonstrate the improvement of LDA-based Fisher kernel technique in text similarity evaluation, we omit the length probability part in the cost function and compare the LDA-based Fisher kernel and the word-frequency cosine similarity by the error rate P_k of segmenting texts. Figure 1 shows the error rates for different sets of data. On average, the error rates are reduced by as much as about 30% over word-frequency cosine similarity with our methods, which shows Fisher kernel similarity measure,with latent topic information added by LDA, outperforms traditional word similarity measure. The performance comparisons drawn from Set 3-5 and Set 13-15 indicates that our similarity algorithm can uncover more descriptive statistics than traditional one especially for segments with less sentences due to its prediction on latent topics.

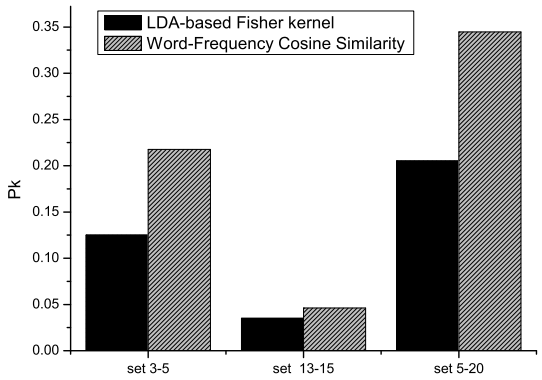


Figure 1: *Error Rate P_k on different data sets with different similarity metrics.*

In the cost function, there are three parameters μ , σ and λ. We determine appropriate μ and σ with hold-out data. For the value of λ, we take it between 0 and 1 because the length part is less important than the similarity part according to our preliminary experiments. We design the experiment to study λ's impact on segmentation by varying it over a certain range. Experimental results in Figure 2 show that the reduce of error rate achieved by our algorithm is in a range from 14.71% to 53.93%. Set 13-15 achieves best segmentation performance, which indicates the importance of text structure: it is easier to segment the topic with regular length and more sentences. The performance on Set 5-20 obtains the best improvement with our methods, which illustrates that LDA-based Fisher kernel can express text similarity more exactly than word density similarity on irregular-sized segments.

Table 1: *Evaluation against different algorithms on Set 5-20.*

Algo.	P_k	Recall	Precision
TextTiling	0.226	66.00%	60.72 %
P. Fragkou Algo.	0.344	69.00%	37.92 %
Our Algo.	0.205	59.00%	62.27 %

While most experiments of other authors were taken on short regular-sized segments which was firstly presented by (Choi, 2000), we use comparatively long range of segments, Set 5-20, to evaluate different algorithms. Table 1 shows that, in terms of

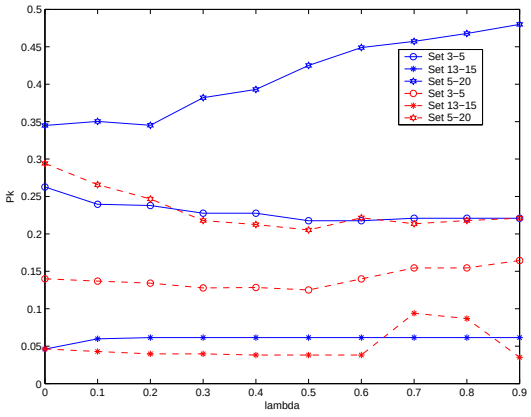


Figure 2: *Error Rate P_k when the λ changes. There are two groups of lines, the solid lines representing algorithm of (Fragkou et al., 2004) while the dash ones indicate performance of our algorithm, and each line in a group shows error rates in different data sets.*

P_k, our algorithm employing dynamic programming as P. Fragkou Algo. achieves the best performance among those three. As for long irregular-sized text segmentation, although local even-sized blocks similarity provides more exact information than the similarity between global irregular-sized texts, with the consideration of latent topic information, the latter will perform better in the task of text segmentation. Though the performance of the proposed method is not superior to TextTiling method, it avoids thresholds selection, which makes it robust in applications.

4 Conclusions and Future Work

We present a new method for topic-based text segmentation that yields better results than previously methods. The method introduces a LDA-based Fisher kernel to exploit text semantic similarities and employs dynamic programming to obtain global optimization. Our algorithm is robust and insensitive to the variation of segment length. In the future, we plan to investigate more other similarity measures based on semantic information and to deal with more complicated segmentation tasks. Also, we want to exam the factor importance of similarity and length in this text segmentation task.

Acknowledgments

The authors would like to thank Jiazhong Nie for his help and constructive suggestions. The work was supported in part by the National Natural Science Foundation of China (60435010; 60535030; 60605016), the National High Technology Research and Development Program of China (2006AA01Z196; 2006AA010103), the National Key Basic Research Program of China (2004CB318005), and the New-Century Training Program Foundation for the Talents by the Ministry of Education of China.

References

Doug Beeferman, Adam Berger and John D. Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning*, 34(1-3):177–210.

David M. Blei and Andrew Y. Ng and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of machine Learning Research* 3: 993–1022.

Thorsten Brants, Francine Chen and Ioannis Tsochantaridis. 2002. Topic-Based Document Segmentation with Probabilistic Latent Semantic Analysis. *CIKM '02*211–218

Freddy Choi, Peter Wiemer-Hastings and Johanna Moore. 2001. Latent Semantic Analysis for Text Segmentation. *Proceedings of 6th EMNLP*, 109–117.

Freddy Y. Y. Choi. 2000. Advances in Domain Independent Linear Text Segmentation. *Proceedings of NAACL-00*.

Nello Cristianini, John Shawe-Taylor and Huma Lodhi. 2001. Latent Semantic Kernels. *Proceedings of ICML-01, 18th International Conference on Machine Learning* 66–73.

Pavlina Fragkou, Petridis Vassilios and Kehagias Athanasios. 2004. A Dynamic Programming Algorithm for Linear Text Segmentation. *J. Intell. Inf. Syst.*, 23(2): 179–197.

Marti Hearst. 1994. Multi-Paragraph Segmentation of Expository Text. *Proceedings of the 32nd. Annual Meeting of the ACL*, 9–16.

Thomas Hofmann. 2000. Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization. *Advances in Neural Information Processing Systems* 12: 914–920.

Xiang Ji and Hongyuan Zha. 2003. Domain-Independent Text Segmentation Using Anisotropic Diffusion and Dynamic Programming. *Proceedings of the 26th annual international ACM SIGIR Conference on Research and Development in Informaion Retrieval,* 322–329.

Hajime Mochizuki, Takeo Honda and Manabu Okumura. 1998. Text Segmentation with Multiple Surface Linguistic Cues. *Proceedings of the COLING-ACL'98*, 881-885.

Jeffrey C. Reynar. 1998. Topic Segmentation: Algorithms and Applications. PhD thesis. University of Pennsylvania.

Author Index

Adriaans, Pieter, 185
Agichtein, Eugene, 97
Ahmad, Faisal, 17
Alfonseca, Enrique, 253
Aw, Aiti, 149, 157

Bach, Nguyen, 77
Badr, Ibrahim, 153
Bangalore, Srinivas, 225
Batista, Fernando, 1
Belz, Anja, 197
Bethard, Steven, 177
Bikel, Daniel M., 145
Bilac, Slaven, 253
Boston, Marisa Ferrara, 5
Brown, Susan Windisch, 249

Callan, Jamie, 85
Carroll, James, 65
Castelli, Vittorio, 145
Chali, Yllias, 9
Charniak, Eugene, 41, 101
Chen, Boxing, 157
Chen, Wei, 13
Chitturi, Rahul, 21
Crammer, Koby, 233

Dalbelo Bašić, Bojana, 181
de la Chica, Sebastian, 17
DeNero, John, 25
Diab, Mona, 117
Dligach, Dmitriy, 29
Dredze, Mark, 33, 233
Dreyer, Markus, 81
Duh, Kevin, 37
Dupoux, Emmanuel, 165

Egg, Markus, 245
Eisner, Jason, 81

Elhadad, Michael, 237
Elsayed, Tamer, 265
Elsner, Micha, 41

Finkel, Jenny Rose, 45
Foster, Jennifer, 221

Gabbard, Ryan, 209
Gao, Quin, 77
Gatt, Albert, 197
Georgila, Kallirroi, 49
Gilbert, Juan, 173
Glass, James, 153
Goldberg, Yoav, 237
Goldwasser, Dan, 53
Gravano, Agustín, 169

Habash, Nizar, 57, 117
HaCohen-Kerner, Yaakov, 61
Haertel, Robbie, 65
Hale, John T., 5
Hansen, John, 21
Hashimoto, Chikara, 69
He, Zhongjun, 161
Henderson, James, 73
Hewavitharana, Sanjika, 77
Hida, Shinya, 125
Hildebrand, Almut Silja, 77
Hirschberg, Julia, 169
Hou, Yuexian, 89

Joty, Shafiq, 9

Karakos, Damianos, 81
Kass, Ariel, 61
Katrenko, Sophia, 185
Khudanpur, Sanjeev, 81, 165
Kim, Donghyun, 229
Kim, Harksoo, 229

Kirchhoff, Katrin, 37
Klein, Dan, 25
Kliegl, Reinhold, 5
Koller, Alexander, 245
Kondadadi, Ravikumar, 205
Kordoni, Valia, 189
Krahmer, Emiel, 193
Kulick, Seth, 209
Kulkarni, Anagha, 85
Kurohashi, Sadao, 69

Lee, Hyunjung, 229
Lemon, Oliver, 73
Li, Haizhou, 149, 157
Li, Runxin, 269
Li, Shoushan, 257
Li, Wenjie, 89
Lin, Feng, 217
Lin, Jen-Hsiang, 93
Lin, Jimmy, 265
Lin, Shouxun, 161
Liu, Chao-Lin, 93
Liu, Feifan, 201
Liu, Qun, 161
Liu, Yandong, 97
Liu, Yang, 201
Lu, Qin, 89
Luo, Dingsheng, 269

Mamede, Nuno, 1
Manning, Christopher D., 45
Marsi, Erwin, 193
Martin, James H., 17, 177, 241
McClosky, David, 101
McCoy, Kathleen, 261
McMillian, Yolanda, 173
Merlo, Paola, 213
Miller, Tim, 105
Moilanen, Karo, 109
Moore, Johanna, 49
Moschitti, Alessandro, 113
Musillo, Gabriele Antonio, 213

Nakagawa, Seiichi, 125
Narayanan, Shrikanth, 225
Nelken, Rani, 137
Nenkova, Ani, 169
Nielsen, Rodney D., 241
Noamany, Mohamed, 77

Oard, Douglas, 265

Palmer, Martha, 29, 241
Paris, Cécile, 129
Peretz, Ariel, 61
Peter, McClanahan, 65
Petrović, Saša, 181
Pharies, Stefan, 253
Pulman, Stephen, 109

Quarteroni, Silvia, 113

Rambow, Owen, 117
Rangarajan Sridhar, Vivek Kumar, 225
Regneri, Michaela, 245
Ringger, Eric, 65
Roa, Sergio, 189
Roth, Dan, 53
Roth, Ryan, 117
Rottmann, Kay, 77
Rudin, Cynthia, 117

Schilder, Frank, 205
Schuler, William, 105
Seo, Jungyun, 229
Seon, Choong-Nyoung, 229
Seppi, Kevin, 65
Sikirić, Ivan, 181
Šnajder, Jan, 181
Sumner, Tamara, 17
Sun, Qi, 269
Syed, Umar, 121

Trancoso, Isabel, 1
Trnka, Keith, 261
Tsuchiya, Masatoshi, 125

van Genabith, Josef, 221
van Pelt, Paul, 193
Varadarajan, Balakrishnan, 165
Vasishth, Shravan, 5
Vogel, Stephan, 77

Wagner, Joachim, 221
Wallenberg, Joel, 33

Wan, Stephen, 129
Wang, Linlin, 133
Ward, Wayne, 241
Wei, Furu, 89
Weng, Fuliang, 217
Williams, Jason, 121
Wolters, Maria, 49
Wong, Kam-Fai, 133
Wu, Xihong, 269

Xia, Yunqing, 133
Xiong, Deyi, 149
Xu, Mingxing, 133

Yamangil, Elif, 137
Yuret, Deniz, 141

Zbib, Rabih, 153
Zhang, Min, 149, 157
Zhang, Peng, 89
Zhang, Yi, 189
Zong, Chengqing, 257